UNIVERSITY OF TRENTO

INdAM

**Doctorate of National Interest**
**on**
**Space Science and Technology**

# Mathematical and Computational Methods for the $n$-Body Problem


## Margaux Introna

**Supervisor**
Prof. Susanna Terracini
Affiliation: Department of Mathematics, University of Turin, Italy

**Co-Supervisor**
Prof. Vivina Laura Barutello
Affiliation: Department of Mathematics, University of Turin, Italy

**Co-Supervisor**
Prof. Massimiliano Vasile
Affiliation: Department of Mechanical & Aerospace Engineering, University of Strathclyde, United Kingdom



Janurary 2026

*Success is not final, failure is not fatal:*
*it is the courage to continue that counts.*
Winston S. Churchill

# Abstract

The gravitational $n$-body problem, considered one of the most classical and enduring challenges in celestial mechanics, seeks to describe the motion of $n$ masses interacting through Newtonian gravity. Despite its deceptively simple formulation, the problem exhibits a profound interplay between order and chaos, giving rise to an extraordinary variety of dynamical behaviours. This Thesis develops an integrated framework that combines variational analysis, advanced numerical optimisation, and rigorous computer-assisted proofs (CAPs) to detect, classify, and validate periodic solutions of the $n$-body problem.

The work begins by reformulating the search for periodic trajectories as a variational problem, where periodic orbits correspond to critical points of an action functional defined over the space of periodic loops. A discretised Fourier representation is then introduced, reducing the infinite-dimensional variational problem to a finite-dimensional optimisation task. Within this setting, novel hybrid algorithms are implemented, most notably the Multi-Population Adaptive Inflationary Differential Evolution Algorithm and the Lattice domain-partitioning scheme, which jointly enable efficient exploration and refinement of the action landscape under prescribed symmetry constraints.

Once candidate periodic orbits are identified, their structure is analysed through a combination of the discrete Morse index, intra- and trans-level distance metrics, and topological invariants such as the winding numbers. These diagnostics reveal the local and global organisation of the action landscape, distinguishing between minimum points, saddle-type solutions, and families of related orbits. The Thesis further extends classical variational methods by implementing a constructive Mountain Pass algorithm, based on the Ambrosetti–Rabinowitz and Barutello–Terracini frameworks, to uncover non-minimising critical points corresponding to new periodic trajectories.

The final part of the work establishes a rigorous computer-assisted validation framework. Using interval arithmetic, validated Fourier-series operations, and fixed-point theorems, numerically computed orbits are enclosed within certified bounds, thereby providing formal proofs of existence for true periodic solutions of Newton's equations. This CAPs methodology transforms high-precision computations into mathematically guaranteed results, uniting analytical rigour with computational discovery.

Overall, the Thesis advances the state of the art by integrating variational theory, numerical optimisation, and validated computation into a single, coherent methodology. The resulting framework not only expands the catalogue of known periodic orbits in the gravitational $n$-body problem but also establishes a reproducible and extensible foundation for rigorous exploration of nonlinear dynamics in celestial mechanics and beyond.

**Keywords.** $n$-body problem, numerical optimisation, computer-assisted proofs, celestial mechanics, variational method

# Contents

# Chapter 1

# Introduction

This Chapter serves a dual purpose. Sections 1.1 and 1.2 provide a self-contained introduction to the classical gravitational $n$-body problem and a review of the existing analytical, numerical, variational, and computer-assisted methods that form the current state of the art. These sections are primarily expository and are intended to place the present work within its broader historical and scientific context.
The original contributions of this Thesis begin in Section 1.3, where the main theoretical, algorithmic, and computational advances developed by the author are summarised. In particular, this Chapter delineates the novel variational–computational framework proposed in this work, clarifies how it extends and unifies existing approaches, and outlines the specific results obtained by the author that are developed in detail in the subsequent Chapters.

## 1.1 The $n$-body problem

Among the classical problems of celestial mechanics, the gravitational $n$-body problem stands as one of the most emblematic and challenging. Its enduring fascination stems from the coexistence of two features: on the one hand, a formulation of striking simplicity, dictated directly by Newton's laws; on the other, an extraordinary richness of dynamical behaviour, ranging from predictable regular motion to highly chaotic trajectories. In its most general form, the problem asks us to describe the evolution of $n$ massive point particles, interacting only through their mutual gravitational attraction, as they move in Euclidean space.
Formally, if $x_1, \dots, x_n \in \mathbb{R}^3$ denote the positions of the bodies and $m_1, \dots, m_n > 0$ their masses, the equations of motion are given by

$$m_k \ddot{x}_k(t) = -\sum_{j \neq k} m_k m_j \frac{x_k(t) - x_j(t)}{\|x_k(t) - x_j(t)\|^3} \qquad k \in \{1, \dots, n\}, \tag{1.1}$$

where $\|\cdot\|$ is the Euclidean norm. A solution of the $n$-body problem is thus the set of trajectories $x_k(t)_{k=1}^n$ satisfying this coupled system of second-order differential equations.
At first glance, the singular nature of the problem appears daunting: the force law diverges whenever two or more bodies collide, so the equations are undefined at such events. These singularities, coupled with the nonlinear interaction of all the particles, endow the problem with extraordinary complexity.
The simplest nontrivial case, $n = 2$, admits a complete description. After reducing to relative two-body motion, each trajectory traces a conic section, ellipse, parabola, or hyperbola, classified by the sign of the mechanical energy. This integrable system forms the cornerstone of classical celestial mechanics and underlies Kepler's laws of planetary motion. Yet the moment we turn to $n = 3$, the picture changes dramatically: the system ceases to be integrable in general, and small perturbations in initial conditions can produce large deviations in trajectories, an early manifestation of what we now call sensitive dependence on initial data.

This transition from simplicity to unpredictability was famously highlighted by Henri Poincaré [1] in his pioneering studies of the three-body problem.

> *D'ailleurs, ce qui nous rend ces solutions périodiques si précieuses, c'est qu'elles sont, pour ainsi dire, la seule brèche par où nous puissions essayer de pénétrer dans une place jusqu'ici réputée inabordable...*

Poincaré revealed that, although the governing equations are deterministic, their solutions can behave in ways that are effectively unpredictable. Crucially, he emphasised the importance of periodic solutions: far from being mathematical curiosities, they provide a framework for understanding the global structure of dynamical systems.

> *...voici un fait que je n'ai pu démontrer rigoureusement, mais qui me paraît pourtant très vraisemblable. Étant données des équations de la forme définie dans le n° 131 et une solution particulière quelconque de ces équations, on peut toujours trouver une solution périodique (dont la période peut, il est vrai, être très longue), telle que la différence entre les deux solutions soit aussi petite qu'on le veut, pendant un temps aussi long qu'on le veut.*

In Poincaré's words, periodic orbits are "*the only breach through which we might try to penetrate a fortress so far deemed unassailable.*" He even conjectured that arbitrary solutions could be approximated by periodic ones, a view that continues to shape modern research.
The search for periodic trajectories is central not only from a dynamical standpoint but also across other disciplines. In semiclassical physics, for instance, families of periodic orbits underpin trace formulae that relate the spectra of quantum systems to their classical counterparts, forging deep and unexpected links between celestial mechanics and quantum theory [2].
Another fundamental aspect of the $n$-body problem is its deep and enduring connection with chaos. Even in the three-body case, the coexistence of regular and chaotic regimes produces an intricate phase-space structure, where stable periodic motions are often surrounded by regions of irregular wandering. In Devaney's formulation [3], a dynamical system is said to be chaotic if it satisfies three defining properties:

(i) *sensitivity to initial conditions* (nearby trajectories diverge exponentially),

(ii) *topological transitivity* (the trajectory of one open set eventually overlaps any other), and

(iii) *density of periodic orbits* in phase space.

These conditions capture the paradoxical nature of chaos: a system governed by deterministic laws, yet effectively unpredictable in practice. While popular accounts often emphasise sensitivity to initial conditions, the so-called "butterfly effect", the full mathematical structure of chaos is richer, relying on the interplay between topological transitivity and the abundance of periodic orbits.
For the gravitational three-body problem, a complete and rigorous proof of chaos in the strict Devaney sense remains elusive. Nonetheless, analytical studies and extensive numerical simulations overwhelmingly indicate that chaotic behaviour is typical rather than exceptional: for most initial conditions, trajectories exhibit exponential sensitivity and irregular motion instead of integrable regularity (see, e.g., [4–11]). The three-body problem thus exemplifies how deterministic Newtonian laws can generate complex, seemingly unpredictable dynamics, precisely the phenomenon Poincaré foresaw and that continues to drive modern research.
Despite centuries of progress, the general $n$-body problem still resists complete resolution. Collisional singularities, the absence of global integrability, and the immense dimensionality of phase space all conspire to preclude explicit solutions. Yet these challenges have spurred the development of increasingly sophisticated analytical and computational tools: regularisation methods to handle

singularities, variational approaches to uncover new orbit families, and, more recently, *computer-assisted proofs* that bridge numerical computation and mathematical rigour.

Computer-assisted proofs represent a major paradigm shift in the study of dynamical systems. By combining interval arithmetic, rigorous error control, and topological methods, they convert high-precision numerical data into mathematically certified statements. Such techniques have been successfully applied to verify the existence and stability of periodic orbits, establish symbolic dynamics, and rigorously confirm chaotic scattering in restricted three-body models. Notably, the pioneering works of Simó, Zgliczyński, and collaborators have demonstrated how validated numerics can turn computational explorations into formal proofs (see e.g. [12–15]). Modern frameworks, such as the CAPD library [16] and interval Newton methods, now enable the systematic verification of dynamical structures with fully controlled numerical uncertainty. In this way, computation has evolved from an exploratory tool into an integral part of rigorous mathematical analysis.

Even so, many fundamental questions remain unresolved. Among these are whether the set of central configurations is finite, whether stable periodic orbits exist beyond the integrable cases, whether every topological braid type can be realised by an $n$-body trajectory, and whether scattering solutions can densely fill configuration space. As Montgomery discusses in [17], these questions reveal the deep interplay of geometry, topology, and dynamics that characterises current research. They also highlight the diversity of mathematical approaches, from algebraic geometry and KAM theory to variational and topological methods, that continue to illuminate this classical yet ever-evolving field.

In this light, the gravitational $n$-body problem is far more than a technical challenge of celestial mechanics. It stands as a paradigmatic model of nonlinear dynamics, embodying the subtle balance between order and chaos, determinism and unpredictability. Today, it serves as both a testing ground for new mathematical theories and a bridge between analysis, computation, and geometry, continuing to inspire theoretical insight and computational innovation alike.

## 1.2 State of the art

The modern study of the gravitational $n$-body problem can be approached through three complementary yet deeply interconnected perspectives: *(i)* direct numerical integration of Newton's equations (1.1), *(ii)* analytical perturbation theory within the Hamiltonian framework, and *(iii)* variational methods, which reformulate the search for periodic orbits as the identification of critical points of an action functional.

The first approach focuses on the *direct propagation* of the $n$-body equations of motion, with the goal of computing trajectories and exploring the dynamical behaviour of systems over extended timescales. From a computational standpoint, such integration demands numerical schemes that maintain accuracy while preserving key invariants of motion. Traditional high-order integrators, though highly accurate over short intervals, tend to accumulate secular energy errors that distort long-term dynamics. To overcome this limitation, *symplectic* and *variational integrators* have become indispensable. These structure-preserving algorithms respect the underlying geometric properties of Hamiltonian flows, most notably the symplectic two-form and phase-space volume, yielding excellent long-term energy behaviour and qualitative fidelity to the true dynamics (see, e.g., [18–20]). Classic examples include the Störmer–Verlet scheme and the Wisdom–Holman method, which exploit the separability of the Hamiltonian into integrable and perturbative parts to achieve efficient and accurate integration of planetary systems [21, 22].

Regularisation techniques, such as the Levi–Civita [23] and Kustaanheimo–Stiefel transformations [24], are also essential in handling close encounters and binary collisions, transforming singular configurations into smooth dynamical variables. Combined with adaptive time-stepping and parallel implementations, these advances have made it possible to simulate complex systems, ranging from resonant multi-planet interactions to dense stellar clusters, with unprecedented precision and temporal depth (see, e.g. [25, 26]).

The second approach, rooted in the classical tradition of celestial mechanics, aims to understand the dynamics analytically through *perturbative and asymptotic methods*. Following the works of Laplace, Lagrange, and Hill, the motion of each body is decomposed into Keplerian elements (semi-major axis, eccentricity, inclination, and related angular parameters) that, in the absence of perturbations, remain constant. Mutual gravitational interactions induce slow variations, both *oscillatory* (short-period) and *secular* (long-period), in these elements. The development of systematic averaging techniques, beginning with Lagrange's planetary equations and Laplace's expansions of the disturbing function, made it possible to construct perturbation series that describe the slow evolution of orbital configurations over astronomical timescales (see, e.g., [9, 27–29]).
Within the Hamiltonian formalism, such perturbative analyses are expressed through canonical transformations that bring the Hamiltonian into progressively simpler or "normal" forms. The unperturbed system, composed of independent Keplerian motions, is integrable, while mutual interactions introduce a nearly integrable perturbation. In this setting, the celebrated Kolmogorov–Arnold–Moser (KAM) theorem ensures the persistence of quasi-periodic invariant tori under small perturbations, guaranteeing the stability of a large measure of initial conditions. Complementarily, Nekhoroshev theory provides exponential bounds on the stability time of actions in analytic nearly integrable systems, even where KAM tori are destroyed [30, 31]. Conversely, in resonant domains, Arnold diffusion can occur, leading to slow chaotic transport through phase space. Together, these results form the modern mathematical framework for understanding stability and chaos in Hamiltonian celestial systems.
Analytical perturbation theory and high-fidelity numerical propagation thus form the twin pillars of modern dynamical astronomy: the former elucidating qualitative mechanisms such as resonances, stability, and chaos, and the latter providing the quantitative tools to explore them in practice.

Parallel to these developments, variational approaches have emerged, particularly since the 1990s, as one of the most powerful tools for studying singular dynamical systems, including the gravitational $n$-body problem. Building upon classical ideas from the calculus of variations, these methods exploit the principle that periodic solutions of Newton's equations correspond to critical points of an appropriate action functional. This perspective not only provides a unifying conceptual framework linking dynamics and geometry but also offers constructive numerical techniques for identifying periodic, quasi-periodic, and choreographic solutions of the $n$-body problem. In this context, a central object of study is the Lagrangian action functional

$$\mathcal{A} = \int_0^T \left[ \frac{1}{2} \sum_{i=1}^{n} m_i \|\dot{x}_i(t)\|^2 + \sum_{i<j} \frac{m_i m_j}{\|x_i(t) - x_j(t)\|} \right] \mathrm{d}t \tag{1.2}$$

defined over the space of $T$-periodic loops. Searching for collision-free periodic orbits thus reduces to identifying critical points of $\mathcal{A}$ in a suitable function space (see, e.g., the monograph [32]). Among these, minimisers are of particular interest, provided coercivity can be ensured and compactness issues controlled.
A typical roadmap for variational techniques involves three main steps:

1. defining a differentiable functional on a candidate space of loops, typically a Sobolev space such as $H^1$, that reflects the dynamical or geometric structure of the equations;

2. establishing a variational principle (e.g. Maupertuis' or the least action principle) which guarantees that nontrivial critical points correspond to genuine, collision-free solutions of Newton's equations (1.1);

3. employing direct methods or minimax arguments to prove the existence of such critical points, often relying on coercivity or compactness of sublevel sets.

The main difficulty in applying this scheme to the $n$-body problem lies in the absence of compactness and coercivity: the action functional $\mathcal{A}$ can remain bounded even for sequences of loops that "drift

apart" in space with diverging $H^1$-norm. To overcome this obstacle, researchers have considered reductions of the problem that impose symmetry or other structural constraints. In the case of the symmetric $n$-body problem, group actions play a key role: by restricting attention to paths invariant under a finite group $G$, it becomes possible to recover coercivity and ensure the minimisation process is well-posed (see, for example, [33–42]).
A landmark result in this direction is due to [35], where the authors demonstrated that minimisers of $\mathcal{A}$ restricted to $G$-equivariant loop spaces are necessarily collision-free, thereby yielding genuine symmetric solutions. This theoretical breakthrough has been reinforced by computational tools. Algorithms for constrained optimisation, such as those implemented in SymOrb [43], allow one to construct $G$-equivariant candidate spaces explicitly and to search numerically for minimising orbits. Such numerical evidence not only guides the identification of new families of solutions but also helps classify them by symmetry and variational properties (see also [44–48]).
More recently, a further development has enriched the state of the art: the rise of computer-assisted proofs (CAPs) in celestial mechanics and dynamical systems. While variational and numerical methods offer strong analytical and computational evidence for the existence of periodic orbits, CAPs transform this evidence into fully rigorous mathematical proofs. Using techniques such as interval arithmetic, validated integration, and rigorous continuation methods, one can enclose numerical approximations in mathematically guaranteed error bounds. In the context of the $n$-body problem, CAPs have been successfully applied to confirm the existence of new periodic orbits, to verify bifurcations, and to rigorously control chaotic dynamics (see [13, 49–57]).

In this Thesis, we build upon these developments by uniting variational principles, modern optimisation algorithms, and rigorous computer-assisted methods into a coherent framework for the detection, classification, and validation of periodic solutions in the gravitational $n$-body problem. Our analysis relies primarily on the variational formulation, which provides both the conceptual foundation and the computational strategy for constructing candidate orbits through critical points of the action functional (1.2). The differential formulation of Newton's equations (1.1) is employed chiefly within the computer-assisted component of the work, where validated numerical integration and rigorous error control are used to confirm the accuracy and stability of the obtained solutions. Through this combined methodology, the Thesis aims not only to expand the catalogue of known periodic orbits but also to deepen our understanding of their geometric organisation and dynamical behaviour within the broader phase-space landscape.

## 1.3 Main results

The research presented in this Thesis has produced a set of original contributions at the intersection of variational analysis, numerical optimisation, and computer-assisted proofs applied to the classical $n$-body problem. These results not only extend the theoretical understanding of periodic solutions but also provide concrete computational tools that can be used in practice. The main achievements are as follows:

1. **Detection of critical points.** The first major contribution of this work concerns the systematic detection of critical points of the action functional associated with the gravitational $n$-body problem. Within the variational framework, we establish a general methodology for locating both minimising and non-minimising stationary configurations, corresponding respectively to stable and saddle-type periodic orbits. The procedure begins with the discretisation of the action functional through a truncated Fourier representation of the periodic trajectories. This reduction transforms the infinite-dimensional variational problem into a finite-dimensional optimisation problem, where each coordinate represents a Fourier coefficient encoding the temporal evolution of a body. Symmetry and boundary constraints imposed by the Bolza problem are enforced by projecting each candidate solution onto the corresponding invariant configuration space, ensuring the consistency of the search within

the prescribed symmetry class. The resulting discrete action functional is then analysed using a combination of global and local optimisation techniques. By evaluating the gradient and Hessian structures numerically, we are able to identify stationary configurations where the first variation of the action vanishes. The numerical framework thus enables the detection of multiple classes of critical points, both local minimum points and higher-index saddle points, which often mark the onset of bifurcations and transitions between distinct dynamical regimes. This detection strategy forms the backbone of the computational approach developed throughout the Thesis, providing a unified means of exploring the landscape of periodic solutions in the $n$-body problem.

2. **Algorithmic innovations.** A second key contribution of this work lies in the development of a hybrid optimisation framework that integrates deterministic and stochastic search techniques to efficiently locate critical points of the action functional. Classical gradient-based methods, such as trust-region algorithms, exhibit strong local convergence but are prone to stagnation in the presence of multiple minima or saddle structures. To overcome these limitations, we introduce a multi-stage strategy that couples local refinement with global exploration, thereby enhancing robustness and coverage of the solution space. At the core of this strategy is the *Multi-Population Adaptive Inflationary Differential Evolution Algorithm* (MP-AIDEA), a metaheuristic method designed to balance exploration and exploitation across dynamically interacting subpopulations. The algorithm adaptively regulates population diversity and mutation amplitude based on convergence diagnostics, allowing it to escape shallow local minima and sample widely separated attraction basins. This global search phase is complemented by a local optimisation step based on trust-region refinement, which ensures high-precision convergence once a promising region has been identified. In addition to MP-AIDEA, a complementary algorithm termed *Lattice* is introduced to improve domain exploration through systematic partitioning. The Lattice approach decomposes the parameter space into a hierarchical grid of subdomains, each treated as an independent optimisation problem, enabling fine-grained exploration of the action landscape and facilitating the discovery of multiple, symmetry-distinct periodic orbits. Together, these algorithmic components constitute a novel and flexible computational framework that bridges numerical optimisation, variational discretisation, and symmetry reduction. This hybrid methodology not only enhances the efficiency and reliability of critical-point detection but also provides a scalable foundation for future extensions involving high-dimensional or strongly nonlinear systems within celestial mechanics.

3. **Qualitative and dynamical characterisation.** The third main contribution of this Thesis concerns the investigation of the local and global structure of the periodic trajectories. Once the candidate solutions are established as critical points of the action functional, attention is directed toward assessing their qualitative nature, distinguishing between local minimum points, saddle points, and more complex stationary configurations, and analysing how these structures are organised in the action landscape. We introduce two complementary tools for this purpose: the *discrete Morse index* and the *intra- and trans-level distance analysis*. The first provides a rigorous measure of the functional stability of each critical point, while the second offers a global view of the geometrical and energetic organisation of the minimum points within the configuration space. The discrete Morse index, adapted from the classical notion of the Morse index in infinite-dimensional Hilbert spaces, quantifies the number of independent directions in which the action decreases locally around a critical configuration. Operationally, it is computed as the number of negative eigenvalues of the discretised Hessian matrix of the action functional. Two numerical formulations are implemented, both on the fundamental symmetry domain and on the full orbit, using either the Fourier discretisation of the trajectories or the discrete-time formulation of the action. Each is complemented by a rigorous algorithmic procedure that ensures full consistency between the analytical definition

and its numerical realisation. Complementing this local analysis, we develop a global structural characterisation of the action landscape through the computation of the *intra-level* and *trans-level* distances introduced in [58]. Local minima are grouped into "levels" according to the proximity of their action values, and pairwise distances among and across levels are computed to quantify both the diversity within a level and the accessibility between adjacent levels. A small trans-level distance combined with a large intra-level spread indicates a funnel-like structure, reflecting a smooth transition toward lower-action configurations. This statistical characterisation of the minima offers valuable insight into the organisation of the energy landscape, the connectivity among attraction basins, and the potential for global optimisation strategies. Finally, for selected planar periodic orbits, we introduce a topological invariant, the *winding number*, to characterise their geometric morphology. Computed via the real-coordinate formulation of the contour integral, the winding number provides an integer measure of the number of revolutions each trajectory performs around a reference point (typically the centre of mass). Its numerical evaluation, based on the Fourier representation of the orbits, serves both as a validation tool and as a geometric descriptor of the family of solutions. Together, these analyses furnish a comprehensive understanding of the qualitative stability and geometric organisation of the periodic orbits found through the variational–computational framework. The discrete Morse index reveals the local structure of the action functional near critical points, while the intra- and trans-level metrics and winding numbers illuminate the global topology and connectivity of the solution space, linking stability theory, landscape geometry, and topological classification into a unified diagnostic framework.

4. **Constructive variational methods.** The fourth methodological contribution of this work concerns the constructive identification of non-minimising critical points of the action functional through the implementation of variational algorithms inspired by the Mountain Pass Theorem. Building upon the classical Ambrosetti–Rabinowitz framework and the algorithmic construction proposed by Barutello and Terracini [59], we develop an improved and fully operational scheme capable of locating saddle-type periodic orbits that escape detection by standard minimisation techniques. The algorithm is grounded in the concept of the steepest descent flow associated with the action functional, and in the topological structure of its disconnected sublevels. By iteratively applying a bisection process along paths connecting distinct basins of attraction, the method converges towards a point on the boundary separating these basins, corresponding to a non-minimal critical point. The improvement introduced in this Thesis extends the original Barutello–Terracini scheme by integrating adaptive gradient–descent control, flow-based evolution, and an automated discovery mechanism that records all previously unknown critical points encountered during the iterative search. This enhancement not only increases numerical robustness but also systematically enriches the catalogue of stationary configurations detected. To guarantee theoretical soundness, the action functional is regularised through a smooth modification which ensures differentiability and satisfies the Palais–Smale condition at all levels. This regularisation permits the rigorous application of the Mountain Pass Theorem to the symmetric $n$-body problem, establishing the existence of non-minimising periodic solutions distinct from the minimisers found. The resulting *Mountain Pass Algorithm* thus provides a constructive bridge between abstract variational theory and computational realisation. It delivers an effective mechanism for discovering saddle-type critical orbits within the regularised action landscape, expanding the scope of variational methods beyond minimisation and opening the way to a systematic, algorithmic exploration of higher-index periodic trajectories in the $n$-body problem.

5. **Rigorous validation via CAPs.** The final contribution of this Thesis concerns the development of a comprehensive framework for *computer-assisted proofs* (CAPs) aimed at the rigorous validation of periodic orbits obtained through the variational and optimisation procedures

of the preceding chapters. The goal is to convert high-precision numerical approximations into mathematically certified solutions of the $n$-body equations, complete with explicit error bounds and reproducible computational evidence. The approach is based on the principles of *validated numerics*, combining interval arithmetic, functional analysis, and fixed-point methods. All computations are carried out in spaces of scalar and vector-valued Fourier series, equipped with analytically controlled norms. Interval arithmetic with directed rounding is employed to ensure that every operation produces an enclosure guaranteed to contain the true value, thereby eliminating rounding or truncation uncertainties. This rigorous numerical layer forms the backbone of the validation process. A central component of the framework is the representation and manipulation of analytic functions and operators within these series spaces. The implementation provides certified arithmetic for Fourier series, including addition, multiplication, division, and composition, together with algorithms for antiderivatives, norms, and the estimation of truncation errors. The method also extends naturally to vector-valued series, enabling the treatment of multi-body configurations with symmetry constraints. At the operator level, we introduce validated routines for differentiation and the computation of bounds on linear and compact operators, which are essential for applying fixed-point theorems in a rigorous setting. Given a numerically computed periodic orbit, the CAPs procedure constructs an approximate inverse of the differential operator governing the system and verifies, by interval bounds, that a true solution lies within a small neighbourhood of the numerical one. When this condition is met, the existence and uniqueness of an exact periodic solution are established within explicitly known error margins. This validation does not merely confirm numerical accuracy but provides a formal mathematical proof of existence in the functional sense. In this way, computer-assisted proofs complete the methodological cycle of the Thesis. Starting from variational formulations and algorithmic detection of critical points, and proceeding through dynamical characterisation and Mountain-Pass analyses, the CAPs framework provides the final rigorous step: transforming computational discoveries into certified mathematical results. The implementation establishes a reproducible and extensible foundation for the validated study of periodic solutions and dynamical structures in the gravitational $n$-body problem.

### 1.3.1 Conclusion and perspective

This work presents a comprehensive framework that unites variational methods, algorithmic optimisation, and rigorous computer-assisted proofs to detect, classify, and validate periodic solutions in the gravitational $n$-body problem. Beyond the methodological results, the research opens several promising avenues for further development, both on the theoretical and applied fronts.

*Extension of CAPs to complex critical points.* The first natural step forward is to extend the current computer-assisted proof framework to handle higher-index critical points and more intricate dynamical structures. Thus far, the validation pipeline has been successfully applied to minimum points and simple saddle points of the action functional. Generalising these techniques to mountain-pass and multi-saddle configurations would enable a rigorous exploration of the global topology of the variational landscape. This will likely require refined functional settings, adaptive norm control, and hybrid interval–spectral methods to manage the higher-dimensional geometry of unstable manifolds. Such developments would further strengthen the bridge between analytical theory and numerical computation, enabling the certified detection of new, genuinely complex families of periodic orbits.

*Astrophysical exploration.* A second major direction concerns the astrophysical interpretation of periodic orbits and their role in shaping dynamical systems on planetary and stellar scales. Periodic configurations offer valuable insight into resonant structures, long-term stability, and the processes of orbital migration and energy dissipation. In planetary systems, the Laplace resonance among Jupiter's Galilean moons provides a classical example of a stable, periodic four-body configuration. Similar architectures have been identified in exoplanetary systems such as K2-138, where chains

of near-resonant planets display synchronised motion patterns likely produced by tidal dissipation and convergent migration. Extending the present methods to these contexts could elucidate how periodic orbits arise, persist, or disappear under realistic astrophysical perturbations.
In the stellar-dynamical regime, dense environments such as globular or nuclear clusters provide a natural laboratory for the formation and disruption of few-body systems. Here, the interplay between gravitational encounters, relativistic corrections, and chaotic scattering determines the evolution of binaries and higher-order bound configurations. The proposed approach, combining variational discovery with computer-assisted validation, offers a pathway to identify stable or recurrent periodic structures within these highly nonlinear systems, potentially revealing dynamical channels relevant to the formation of compact-object binaries and black-hole mergers.
*Astrodynamical applications.* Finally, the same mathematical machinery holds great potential for applied celestial mechanics. Variational optimisation and CAP-based validation can be adapted to mission design, trajectory optimisation, and the study of multi-satellite formations. Periodic orbits and their stable manifolds serve as natural highways in phase space, allowing for low-energy transfers and station-keeping strategies. A rigorously certified framework would ensure not only numerical accuracy but also formal guarantees of dynamical feasibility, an essential step toward trustworthy, high-precision astrodynamical computations.

In conclusion, the convergence of rigorous mathematics, algorithmic innovation, and astrophysical modelling offers a powerful paradigm for advancing both the theoretical understanding and the practical utilisation of the $n$-body problem. By extending computer-assisted proofs to richer dynamical regimes and applying them to realistic astrophysical and astrodynamical contexts, future research can transform periodic-orbit theory from a classical topic of celestial mechanics into a predictive, quantitative tool for modern gravitational science.

## 1.4 Structure of the Thesis

The Thesis is organised into eight chapters, following a logical progression from the theoretical formulation of the problem to the implementation of numerical and rigorous validation techniques, and finally to the presentation of concrete results and examples. Each chapter contributes a specific methodological or conceptual element to the overall framework.

**Chapter 1 – Introduction.** This opening Chapter introduces the $n$-body problem, its mathematical and physical background, and its significance within celestial mechanics and dynamical systems theory. It reviews the historical development and state of the art, highlighting the open challenges that motivate this work. The main results of the Thesis are then summarised, and future perspectives are outlined. The chapter concludes with an overview of the Thesis structure.

**Chapter 2 – Mathematical framework.** This Chapter establishes the theoretical and notational foundations of the study. It defines the configuration space of the $n$-body system and introduces the necessary elements of group theory, including dihedral, cyclic, and symmetric groups, and their actions on the configuration space. The chapter also discusses how symmetry constraints can be encoded through group actions and presents the concept of $G$-equivariant minimisers of the action functional, setting the stage for the subsequent variational analysis.

**Chapter 3 – Finding critical points.** The third Chapter focuses on the numerical detection of stationary points of the action functional. Various optimisation strategies are presented and compared, ranging from classical trust-region methods to constrained optimisation and metaheuristic algorithms. The MP-AIDEA algorithm is introduced as a hybrid framework combining these approaches to efficiently explore the variational landscape and locate periodic orbits consistent with given symmetry constraints. The chapter concludes with the first set of computed solutions for the symmetric $n$-body problem.

**Chapter 4 – Analysing critical points.** Once candidate orbits are identified, their qualitative nature and stability are analysed in this Chapter. The discrete Morse index is introduced as a numerical tool to distinguish minimum points from saddle-type solutions. Additional geometric measures, such as intra- and trans-level distances, provide insight into the organisation of the action landscape and the connectivity among local minimum points. Finally, the winding number is computed to classify planar periodic trajectories according to their topological properties.

**Chapter 5 – Mountain Pass solutions.** This Chapter extends the variational approach beyond minimising solutions by implementing a constructive version of the Mountain Pass Theorem. An improved algorithm based on the Barutello–Terracini framework is developed to identify non-minimal critical points of the action functional, corresponding to saddle-type periodic orbits. The method is then applied to the symmetric $n$-body problem through a regularised formulation satisfying the Palais–Smale condition, ensuring both theoretical soundness and computational applicability.

**Chapter 6 – Examples.** This Chapter illustrates the full methodology developed in the preceding chapters through explicit case studies involving specific symmetry groups, such as $G = \mathbb{Z}_2$ and $G = D_6$. For each symmetry, both minimising and mountain-pass periodic solutions are computed using the variational and optimisation framework. The examples are intended to demonstrate the applicability and effectiveness of the proposed approach prior to the introduction of rigorous computer-assisted validation.

**Chapter 7 – Computer-assisted proofs.** The seventh Chapter presents the rigorous validation framework based on computer-assisted proofs. It introduces the principles of interval arithmetic and describes the representation of analytic functions and operators in spaces of scalar and vector-valued Fourier series. The Chapter then details the implementation of validated operations, operator bounds, and fixed-point methods that provide formal mathematical proofs of existence for numerically obtained periodic orbits. This framework bridges the gap between numerical computation and analytical rigour.

**Chapter 8 – Results and conclusions.** This final Chapter presents the main numerical and computer-assisted results obtained with the proposed framework. It reports the discovery of several families of symmetric periodic orbits in the $n$-body problem, computed under various symmetry constraints. Each orbit is analysed in terms of its action value, stability properties, and discrete Morse index. The Chapter also provides the corresponding rigorous validations performed through the computer-assisted proof framework, confirming the existence of these orbits within explicitly certified error bounds. These results collectively demonstrate the accuracy, robustness, and reliability of the integrated variational–numerical–rigorous approach developed throughout the Thesis, and are complemented by a set of general conclusions that synthesise the contributions of this work and outline perspectives for future research.

Altogether, the structure of the Thesis reflects a coherent narrative: from the theoretical formulation of symmetric variational problems, through numerical exploration and stability analysis, to the rigorous validation of periodic orbits. Each component, variational, algorithmic, and analytical, contributes to building a complete, reliable, and extensible methodology for the study of periodic solutions in the gravitational $n$-body problem.

# Chapter 2

# Mathematical framework

This Chapter establishes the mathematical setting and notation used throughout the Thesis. Its purpose is primarily expository: it collects standard definitions and known results from the Newtonian $n$-body problem, finite group actions, and symmetric variational principles in order to make the subsequent chapters self-contained and to fix a common language and set of conventions. We begin by recalling classical definitions related to the Newtonian $n$-body problem and then introduce basic notions from group theory that will be instrumental in the construction of symmetric solutions. Further background and historical motivation can be found in Section 1.1 and Section 1.2. Our main references for the present material are [35] and [60]. In particular, the definitions of the loop space, the group action on $\Lambda$, and the reduction to a fundamental domain will be used repeatedly when formulating and implementing the optimisation and validation procedures in later chapters.

## 2.1 Configuration space and notation

We consider the motion of $n$ point masses interacting through Newtonian gravitation in a $d$-dimensional Euclidean space $\mathbb{R}^d$. Let $m_1, \dots, m_n > 0$ be the masses (with $n \geq 2$) and define the configuration space with vanishing centre of mass as

$$\chi := \left\{ x = (x_1, \dots, x_n) \in (\mathbb{R}^d)^n : \sum_{i=1}^{n} m_i x_i = 0 \right\}.$$

Collisions correspond to singular configurations where two or more bodies coincide. Denoting

$$\Delta_{ij} := \{ x \in \chi : x_i = x_j \}, \quad \Delta := \bigcup_{i,j} \Delta_{ij},$$

the set of collision-free configurations is given by $\hat{\chi} := \chi \setminus \Delta$. Given $T_x\chi$ the tangent space to $\chi$, the Newtonian potential and kinetic energy functions are respectively defined by

$$U(x) := \sum_{i<j} \frac{m_i m_j}{\|x_i - x_j\|}, \qquad K(\dot{x}) := \frac{1}{2} \sum_{i=1}^{n} m_i \|\dot{x}_i\|^2, \qquad x \in \chi,\ \dot{x} \in T_x\chi,$$

so that the associated Lagrangian is

$$L(x, \dot{x}) := K(\dot{x}) + U(x), \quad x \in \chi,\ \dot{x} \in T_x\chi.$$

The equations of motion then read

$$m_i \ddot{x}_i = \frac{\partial U}{\partial x_i}, \quad i = 1, \dots, n, \tag{2.1}$$

which describe the evolution of $n$ interacting bodies in $\mathbb{R}^d$.
As outlined in the Chapter 1, a central objective of this work is to identify periodic, collision-free solutions of Equation (2.1). In the variational framework, such solutions are characterised as critical points of an action functional defined on a suitable space of $T$-periodic loops. The general methodology, already discussed in Section 1.2, may be briefly summarised as follows:

- define an appropriate function space of loops (typically with $H^1$ regularity) satisfying symmetry or boundary constraints;
- introduce a differentiable functional, arising from the weak formulation of Equation (2.1), whose critical points correspond to periodic trajectories;
- apply variational principles (such as Maupertuis' or the least-action principle) to guarantee that non-trivial critical points yield genuine solutions;
- exploit compactness and coercivity properties to prove the existence of minimisers or saddle-type critical points.

As is well known, the lack of coercivity in the general $n$-body problem presents the main technical difficulty. Following the approach discussed in Chapter 1, we overcome this by restricting the search to spaces of loops possessing prescribed symmetries in space, time, and particle indices, which restore compactness and make the variational analysis tractable.
Let us concentrate on some basic notations first. For $T > 0$, set the time circle $\mathbb{T} \simeq \mathbb{R}/T\mathbb{Z}$ so that we can define the space of $H^1$-periodic loops in $\chi$ and non collision ones as

$$\Lambda := H^1(\mathbb{T};\chi) = \{x \in H^1(\mathbb{T};\chi) : x(0) = x(T)\}$$
$$\hat{\Lambda} := H^1(\mathbb{T};\hat{\chi}) = \{x \in H^1(\mathbb{T};\hat{\chi}) : x(0) = x(T)\}.$$

We can write the Lagrange action functional $\mathcal{A} : \Lambda \longrightarrow \mathbb{R} \cup \{+\infty\}$ as

$$\mathcal{A}(x) := \int_0^T L(x(t), \dot{x}(t))\,\mathrm{d}t, \quad \text{for } x \in \Lambda. \tag{2.2}$$

As one has that $\mathcal{A} \in \mathcal{C}^1(\hat{\Lambda})$, one can deduce that if $x \in \hat{\Lambda}$ is a critical point of $\mathcal{A}$, then $x$ is a $T$-periodic solution of (2.2). For $x = x(t) \in \Lambda$, the *collision times of* $x$ can be defined as the set $x^{-1}\Delta \subset \mathbb{T}$.

## 2.2 Basic definitions in group theory

Let us recall some known definitions from finite group theory.

**Definition 2.1** (Group)**.** A *group* $(G, \cdot)$ is a set $G$ and a binary operation "$\cdot$" on the elements of $G$ which satisfy the following properties:

- the operation is *associative*, i.e. for any $g_1, g_2, g_3 \in G$ one has
$$(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3);$$
- there is an element $1_G$ called the *unit* of $G$ such that for any $g \in G$ one has
$$1_G \cdot g = g = g \cdot 1_G;$$
- for any element $g \in G$ there exists its *inverse* element $g^{-1}$ such that
$$g \cdot g^{-1} = g^{-1} \cdot g = 1_G.$$

From now on, $G$ denotes a finite group with its binary operation, assuming that, for $g_1, g_2 \in G$, $g_1 g_2$ denotes the group operation $g_1 \cdot g_2$. Moreover, $1_G$ will be replaced simply by 1 when its meaning is clear from the context.

**Definition 2.2** (Order)**.** The *order of a group* $G$, denoted as usual with $|G|$, represents the number of elements of $G$.
The *order of an element* $g \in G$ is the smallest $n \in \mathbb{N}$ such that $g^n = 1$.

**Definition 2.3** (Subgroups, cyclic groups)**.** A *subgroup* $H$ of a group $G$ is a subset $H$ of $G$ which is itself a group under the binary operation defined in $G$.
If $S$ is a subset of $G$, the *subgroup generated by* $S$, usually denoted by $\langle S \rangle$, is the smallest subgroup of $G$ which contains $S$. If $\langle S \rangle = G$, we say that $S$ is a set of *generators of* $G$.
If there exists $g \in G$ such that $\langle g \rangle = G$ we say that $G$ is a *cyclic group*.

**Definition 2.4** (Cosets, index)**.** If $G$ is a finite group, then

- if $H$ is a subgroup of $G$ and $g \in G$, then we define the *left cosets* of $H$ in $G$ as the sets

  $$gH = \{gh : h \in H\}, \quad \text{for } g \in G.$$

  and an analogous definition follows for the *right cosets* $Hg$ for $g \in G$;

- the *index* of a subgroup $H$ in $G$ is denoted by $|G : H|$ and represents the number of left (or, equivalently, right) cosets of $H$ in $G$. We also recall the following useful formula

  $$|G : H| = \frac{|G|}{|H|}.$$

In the following definition we recall the properties of *normal subgroups*, which are very useful since they allow to consider *quotients* of $G$.

**Definition 2.5** (Normal subgroups, quotients)**.** We say that a subgroup $N$ of $G$ is a *normal subgroup*, and we will write $N \lhd G$, if $gng^{-1} \in N$, for any $g \in G$, $n \in N$ or, equivalently, if $gN = Ng$ for any $g \in G$.
If $N \lhd G$, the *quotient group* $G/N$ is the group of all cosets of $N$ in $G$, i.e.,

$$G/N := \{gN : g \in G\}.$$

In this case, we have $|G : N| = |G/N|$.

In general, not every subgroup H is normal, therefore one looks for those groups which normalise $H$ and define a quotient.

**Definition 2.6** (Normalisers, Weyl groups)**.** If $H$ is a subgroup of $G$, we define the *normaliser* of $H$ in $G$ as the smallest subgroup of $G$ in which $H$ is normal, and we denote it by $N_G H$. It follows that

$$N_G H := \{g \in G : gHg^{-1} = H\}$$

and clearly $H \lhd N_G H$.
If $H$ is a subgroup of $G$, we define the *Weyl group* of $H$ in $G$ as the quotient group

$$W_G H := N_G H/H = \{wH : w \in N_G H\}.$$

## 2.3 Dihedral groups, symmetric group, subgroups of $O(2)$

Some particular finite groups can be very useful in describing the geometrical symmetries of our dynamical system. In the following pages, the concept of group action is introduced, which is a

very smart way of describing the symmetry of an object and allows to store in a group all the necessary attributes to describe these symmetries. As a matter of fact, there are some particular classes of groups whose action on time and space reduce by symmetry the $n$-body problem.

**Definition 2.7** (Dihedral groups)**.** A *dihedral group* is the group of symmetries of a regular polygon, i.e., the group of *reflections* and *rotations* which fixes the polygon. If we consider a polygon with $n$ edges, also named *$n$-gon*, we define $D_{2n}$ as its dihedral group. The elements of $D_{2n}$ are rotational symmetries, reflectional symmetries and their compositions. Note that rotations which fix a polygon with $n$ edges are those of $\frac{2\pi}{n}$ and their multiples. The order of the dihedral group $D_{2n}$ is exactly $2n$, since it contains $n$ rotational symmetries and $n$ reflectional symmetries.

*Remark.* Note that the dihedral group $D_{2n}$ can be generated by a unique rotation $r$ and a unique reflection $s$, so that it is usually defined through the following *presentation*

$$\langle s, r : s^2 = r^n = (sr)^2 = 1 \rangle.$$

**Definition 2.8** (Symmetric groups)**.** If $X$ is a set, we say that a *permutation* of $X$ is any bijection from $X$ to itself. The family of all permutations of $X$ is a group under composition and we denote it by $\Sigma_n$.
If $n \in \mathbb{N}$ and $X = \{1, 2, \dots, n\}$, we say the permutation group $\Sigma_n$ is the *symmetric group* of degree $n$ and its order is exactly $n!$.

**Definition 2.9** (Cycles)**.** Given $n_1, \dots, n_k \in \{1, 2, \dots, n\}$ the permutation that maps $n_1$ to $n_2$, $n_2$ to $n_3$, …, $n_{k-1}$ to $n_k$ and finally $n_k$ to $n_1$ and leaves the other elements unchanged is called a *$k$-cycle* or a *cyclic permutation of order $k$*. The usual notation for this permutation is simply $(n_1, \dots, n_k)$.

We are very interested in obtaining information about the two-dimensional *orthogonal group* $O(2)$. To briefly recap, $O(2)$ is the group of $2 \times 2$ real orthogonal matrices. It is well known that an orthogonal matrix has a determinant of either 1 or $-1$, and the elements of $O(2)$ with a determinant of 1 form a subgroup known as the *special orthogonal group* $SO(2)$. The elements of $O(2)$ are $2 \times 2$ matrices that represent either a rotation of $\mathbb{R}^2$ about the origin or a reflection with respect to a line passing through the origin. Consider $R_\theta \in O(2)$ and assume that it represents a rotation by an angle $\theta$ about the origin. We can represent $R_\theta$ in the following way:

$$R_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

which is clearly in $SO(2)$ since its determinant is equal to 1. In the same way, an element $S_\theta \in O(2) \setminus SO(2)$ which is a reflection with respect to a line which forms an angle of $\theta$ with the $x$-axis can be represented in this form

$$S_\theta = \begin{pmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{pmatrix}$$

and its determinant is equal to $-1$. Since all the elements of $O(2)$ can be represented as products between these two kinds of matrices, the classification of the finite subgroups of $O(2)$ is quite precise.

**Proposition 2.10** (Finite subgroups of $O(2)$)**.** *Any finite subgroup $H$ of $O(2)$ is either cyclic or dihedral. In particular:*

- *if $H \subseteq SO(2)$, then $H$ is cyclic and only contains rotations;*
- *if $H = \{1, S\}$, with $S \in O(2) \setminus SO(2)$ a reflection, then $H$ is cyclic of order 2;*
- *if $H$ contains both elements of $SO(2)$ and $O(2) \setminus SO(2)$, then it is dihedral.*

## 2.4 Group actions

Representation theory offers a different point of view from which groups can be thought of as structures that *act* on mathematical objects, such as sets or spaces. We recall that our first aim is to endow the space of $H^1$-periodic loops $\Lambda$ with a prescribed symmetry constraint, in order to *compactify* the configuration space $\chi$. A powerful way to describe such a symmetry is through the *action* of a finite group $G$ on $\Lambda$. The rigorous definition of group action is given in the following:

**Definition 2.11** (Group action, Definition 1.1 [43])**.** We say that a finite group $G$ *acts* (on the left) on a set $X$ if there exists a map $\phi : G \times X \to X$ such that

- $\phi(1_G, x) = x$ for any $x \in X$;
- $\phi(g, \phi(h, x)) = \phi(gh, x)$ for any $g, h \in G$ and $x \in X$,

i.e., the map is compatible with the group operation. Since in this case there is no ambiguity between the notations involved, we will write $gx$ instead of $\phi(g, x)$.

Therefore, a group action is essentially a way in which $G$ interacts with the set $X$. Given $x \in X$, it could be useful to regroup those elements of $G$ which fix $x$; on the other hand, we might need to gather those elements of $X$ which are invariant under the action of the whole $G$ or of a subgroup $H$ of $G$. For instance, if $X = \mathbb{R}^2$ and $g \in G$ is a reflection with respect to the line $y = x$, we see that all the points $(x, x)$ are fixed by $g$.

**Definition 2.12** (Isotropy groups and projectors, Definition 1.2 [43])**.** For an element $x \in X$, we define the *isotropy group* of $x$ in $G$ as the group

$$G_x := \{g \in G : gx = x\},$$

which is a subgroup of $G$. Moreover, if $H$ is a subgroup of $G$, we define the set of *points fixed by* $H$ as the set

$$X^H := \{x \in X : H \subseteq G_x\}.$$

Note that $X^H := \{x \in X : hx = x, \forall h \in H\}$.
Moreover, we define the *projector onto* $X^H$ as the projection map

$$\begin{aligned} \pi_H &: X \to X \\ x &\mapsto \pi_H x := \frac{1}{|H|} \sum_{h \in H} \phi(h, x). \end{aligned}$$

**Definition 2.13** (Transitive group action)**.** We say that the action of $G$ on a set $X$ is *transitive* if for any $x, y \in X$ there exists $g \in G$ such that $gx = y$.

*Remark* 2.14. It is easy to check that, if $G$ acts on $X$ and $H$ is a subgroup of $G$, then the Weyl group $W_G H$ acts on $X^H$. Indeed, recalling that

$$W_G H = N_G H \setminus H = \{wH : w \in N_G H\},$$

it is enough to define the map $\psi : W_G H \times X^H \longrightarrow X^H$ such that

$$\psi(W, x) = wx, \quad \text{whenever } W = wH \in W_G H,$$

which easily fulfils the properties of Definition 2.11. Note that, if $H$ is normal in $G$, then the above property guarantees the existence of a natural action of the quotient group $G/H$ over $X^H$.

**Definition 2.15** ($G$-equivariant map)**.** Given two $G$-equivariant spaces $X, Y$, we say that a map $f : X \to Y$ is a *$G$-equivariant map* if

$$f(gx) = gf(x) \quad \text{for any } g \in G, x \in X.$$

*Remark.* If we are given a $G$-equivariant map $f : X \to Y$ and $H$ is a subgroup of $G$, then we immediately obtain a $G$-equivariant map on the fixed points

$$f^H : X^H \longrightarrow Y^H,$$

which is nothing but the restriction of $f$ to $X^H$.

## 2.5 Symmetry constraints

In this Section, we show how imposing a symmetry constraint on a space of loops within the configuration space can be achieved by introducing a group action on that space. As a preliminary remark, it is evident that since the space of loops is infinite-dimensional, we must seek finite-dimensional objects on which a group can effectively act.

### 2.5.1 Symmetries as group actions

Building on the concepts presented in Definition 2.11, from a geometrical perspective, groups can be viewed as sets of symmetries of an object or space that are closed under composition and the inverse operation.
From a different perspective, we can consider a space $X$ and seek groups $G$ that act on $X$ in a suitable manner. To this end, it is useful to introduce another way to describe a group action. For a fixed $g \in G$, we define the map

$$a(g) : X \longrightarrow X, \quad a(g)x = gx,$$

where, with an abuse of notation, $gx$ represents the action of $g$ on $x$ as in Definition 2.11. For any $g \in G$, the group action implies that the map $a(g)$ is a bijection with inverse $a(g^{-1})$; consequently, the set $\{a(g) : g \in G\}$ forms a group under composition and is a subgroup of $\Sigma_X$, the family of all permutations of elements of $X$ (see Definition 2.8). As a result, the action of $G$ on $X$ induces a group homomorphism between $G$ and $\Sigma_X$,

$$a : G \longrightarrow \Sigma_X.$$

This homomorphism $a$ is an example of a representation of $G$, derived from its action on $X$. In general, a representation of $G$ should be seen as a description of $G$ as a subgroup of the automorphisms group of a mathematical object.

### 2.5.2 Action of a finite group on $\Lambda$

Our main goal is to study how a finite group $G$ can act on the loop space $\Lambda = H^1(\mathbb{T}; \chi)$. We have that $\mathbb{T} = \mathbb{R}/T\mathbb{Z}$ and that for any $x = x(t) \in \Lambda$ one can write

$$x(t) = (x_1(t), \dots, x_n(t)) \in \chi, \quad \text{for any } t \in \mathbb{T}.$$

It follows that the configuration space $\chi$ can be seen as a subset of $(\mathbb{R}^d)^n$, as every component $x_i(t) \in \mathbb{R}^d$. Moreover, it is natural to have that the group $G$ acts in the same way on each component. Let us see three finite dimensional objects underlying in $\Lambda$:

- the *space* $\mathbb{R}^d$ on which every component $x_i(t)$ of $x(t)$ lies;
- the *time circle* $\mathbb{T} \subset \mathbb{R}^2$ which represents the period of a trajectory;
- the *index set* $\{1, \dots, n\}$ on which the $n$-bodies are labelled.

One can write the action of $G$ on $\Lambda$ using three different representations of $G$ as a group action respectively on $\mathbb{R}^d$, $\mathbb{R}^2$ and $\{1,\dots,n\}$. In detail, $G$ is represented as a subgroup of $O(d)$, of $O(2)$ and of the symmetric group $\Sigma_n$ through the following homomorphisms

$$\rho : G \to O(d), \quad \tau : G \to O(2), \quad \sigma : G \to \Sigma_n.$$

Firstly, let us analyse the representations $\rho$ and $\sigma$ through which we can define the action of $G$ on the configuration space $\chi$

$$\phi : G \times \chi \to \chi, \quad \phi(g,x) = gx,$$

where

$$gx = g(x_1,\dots,x_n) = (\rho(g)x_{\sigma(g^{-1})1},\dots,\rho(g)x_{\sigma(g^{-1})n}),$$

or shortly

$$(gx)_i = gx_{g^{-1}i}.$$

Since $\rho(g) \in O(d)$ and the permutation $\sigma(g^{-1})$ do not alter the inner product of $(\mathbb{R}^d)^n$, it follows that this action preserves the inner product of $(\mathbb{R}^d)^n$, therefore it is orthogonal. As a matter of fact, let $x, y \in \chi$ and $g \in G$, one has

$$\langle gx, gy\rangle = \sum_{j=1}^{n} \langle \rho(g)x_{\sigma(g^{-1})j}, \rho(g)y_{\sigma(g^{-1})j}\rangle = \sum_{j=1}^{n} \langle x_{\sigma(g^{-1})j}, y_{\sigma(g^{-1})j}\rangle,$$

where the same symbol for the inner product in different dimensions has been used.
As it will be important in the following pages, we require that the homomorphism $\sigma : G \to \Sigma_n$ satisfies the following property

$$\forall g \in G : \{\sigma(g^{-1})i = j \Rightarrow m_i = m_j\}, \tag{2.3}$$

which means that $\sigma(g^{-1})$ can swap only bodies with equal masses. With this property, and recalling Definition 2.13, if the action of $G$ on $\{1,\dots,n\}$ is transitive, it follows that one has $n$ equal masses. Now let us consider an element of $\Lambda$, which is an $H^1$-periodic loop $x : T \to X$, meaning it consistently resides in the configuration space $X$. In Section 2.1, we observed that the periodicity condition is introduced by defining such loops on the time circle $\mathbb{T}$, a one-dimensional object embedded in $\mathbb{R}^2$. The action of $G$ on $\mathbb{T}$ must then be defined through the two-dimensional representation $\tau : G \to O(2)$ as follows:

$$gt = \tau(g^{-1})\, t, \quad \text{for } t \in \mathbb{T},\ g \in G,$$

clearly using an abuse of notation by identifying the time instant $t$ with an element of $\mathbb{S}^1 \subset \mathbb{R}^2$.

*Remark* 2.16. As one has that $\tau : G \to O(2)$ is a homomorphism, thanks to the first isomorphims theorem, it follows that

- $\ker\tau \lhd G$;
- the image $\tau(G)$ is a finite subgroup of $O(2)$;
- $G/\ker\tau \simeq \tau(G)$, therefore it is a finite group as well.

Using the previous Remark and Proposition 2.10, one can conclude that the image $\tau(G)$ is either a cyclic or dihedral subgroup of $O(2)$. Moreover, as $\tau$ represents an action of $G$ in the time circle $\mathbb{T}$ and $\tau(g)$ is either a rotation of a reflection, we use the following notation:

- if $\tau(g)$ is a counter-clockwise rotation of $\theta$, we let $\tau(g)$ act on $\mathbb{T}$ as follows:

$$\tau(g)t = \theta + t;$$

- if $\tau(g)$ is a reflection with respect to a line through the origin which forms an angle of $\theta/2$ with the $x$-axis, we let $\tau(g)$ act on $\mathbb{T}$ as follows:

$$\tau(g)t = \theta - t.$$

Now we can write a way to represent the action of $G$ on the loop space $\Lambda$ using the homomorphisms $\rho, \sigma$ and $\tau$, which is the following

$$(gx)(t) = \left(\rho(g)x_{\sigma(g^{-1})1}(\tau(g^{-1})t), \dots, \rho(g)x_{\sigma(g^{-1})n}(\tau(g^{-1})t)\right),$$

or shortly

$$(gx)_i(t) = (gx_{g^{-1}i}(g^{-1}t))_i,$$

for any $g \in G, t \in \mathbb{T}$ and $x \in \Lambda$. One can also define the set of loops in $\Lambda$ fixed by $G$, or the *$G$-equivariant loops*, as the set

$$\Lambda^G := \{x \in \Lambda : (gx)(t) = x(t),\ \forall t \in \mathbb{T}, g \in G\}$$

and its collision-less subset

$$\hat{\Lambda}^G := \{x \in \hat{\Lambda} : (gx)(t) = x(t),\ \forall t \in \mathbb{T}, g \in G\}.$$

The following result ensures that $\Lambda^G$ is a loop space suitable for applying a variational approach to identify periodic solutions as critical points of the action functional.

**Proposition 2.17** (Proposition 2.5, [60])**.** *Let $G$ be a finite group, with orthogonal representations $\rho : G \to O(d)$, $\tau : G \to O(2)$, $\sigma : G \to \Sigma_n$ satisfying* (2.3)*. Then the following assertions hold true:*

*(i) $\Lambda^G$ is a closed linear subspace of $\Lambda$;*

*(ii) the Lagrange action functional $\mathcal{A}$ is $G$-equivariant, i.e.,*

$$\mathcal{A}(gx) = \mathcal{A}(x),\ \forall g \in G, x \in \Lambda;$$

*(iii) the colliding set $\Lambda$ is $G$-equivariant, i.e.,*

$$x \in \Lambda \Longrightarrow gx \in \Lambda,\ \forall g \in G.$$

From the previous properties, one can deduce a version of the *Palais principle of symmetric criticality* (for more details, see [61]):

**Lemma 2.18** (Lemma 2.6, [60])**.** *Let $\mathcal{A}|_{\Lambda^G}$ be the restriction of the Lagrange action functional $\mathcal{A}$ to the $G$-equivariant loop space $\Lambda^G$.*
*Then, a collision-less critical point $\bar{x} \in \hat{\Lambda}^G$ of $\mathcal{A}|_{\Lambda^G}$ is also a (collision-less) critical point of $\mathcal{A}$ over the whole $\Lambda$.*

**Proposition 2.19** (Proposition 2.7, [60])**.** *Assume that $\mathcal{A}|_{\Lambda^G}$ is not identical $+\infty$. Then*

$$\mathcal{A}|_{\Lambda^G} \text{ is coercive} \iff \chi^G = \{0\}.$$

*As a consequence, if $\chi^G = \{0\}$, there exists at least a minimiser of $\mathcal{A}$ in $\Lambda^G$.*

### 2.5.3 Non reducible group action

In the previous Section, we showed how to equip $\Lambda$ with the group action of a finite group $G$. It is also advantageous to refine the selection of a finite group $G$ based on its action on the periodic

loops in $\Lambda$ and to exclude representations of $G$ that result in *reducible* group actions. This process, outlined in the current section, provides a definition of an *admissible* group action on $\Lambda$.

**Definition 2.20** (Non-reducible actions, Definition 2.6 [43])**.** Let $G$ be a finite group, with orthogonal representations $\rho : G \to O(d)$, $\tau : G \to O(2)$, $\sigma : G \to \Sigma_n$ satisfying (2.3). We say that the action of $G$ on $\Lambda$ is *non-reducible* if the following assumptions hold true:

(i) $\ker\tau \bigcap \ker\rho \bigcap \ker\sigma = 1$;

(ii) there is no proper linear subspace $V$ of $\mathbb{R}^d$ such that

$$x_i(t) \in V \subsetneq \mathbb{R}^d, \ \forall i \in \{1,\dots,n\}, \forall x \in \Lambda^G, \forall t \in \mathbb{T};$$

(iii) there is no integer $k \neq \pm 1$ such that

$$\forall x \in \Lambda^G \ \exists y \in \Lambda \text{ such that } x(t) = y(kt), \ \forall t \in \mathbb{T}.$$

There are some situations in which the action of a group $G$ is reducible.

**Proposition 2.21** (Proposition 2.10, [60])**.** *The action of a finite group $G$ on $\Lambda$ is reducible if one of the following holds:*

*(i)* $\ker\tau \bigcap \ker\sigma \neq 1$*;*

*(ii)* $|\ker\rho \bigcap \ker\sigma| > 2$*.*

*Remark* 2.22. If $\ker\rho \bigcap \ker\sigma = 2$, the action of $G$ is not necessarily reducible. As a matter of fact, when considering the element $g \in (\ker\rho \bigcap \ker\sigma) \setminus \{1\}$, it may act as a rotation or as a reflection, which would not imply condition $(iii)$ in Definition 2.20.

As our aim is to search for classical solutions of the $n$-body problem, those finite group $G$ whose action makes $\hat{\Lambda}^G$ an empty set must be excluded.

**Definition 2.23** (Action bound to collision, Definition 2.10 [43])**.** We say that the action of $G$ on $\Lambda$ is *bound to collisions* if

$$\forall x \in \Lambda^G \text{ it holds } x^{-1}\Delta \neq \varnothing,$$

or, equivalently, if

$$\hat{\Lambda}^G = \varnothing.$$

In the following result, a necessary condition for $\hat{\Lambda}^G$ not to be empty is shown.

**Proposition 2.24** (Proposition 2.13, [60])**.** *If $\ker\tau \bigcap \ker\rho \neq 1$, then the action of $G$ is bound to collisions.*

In conclusion, as we want to exclude those finite groups that induce a reducible or bound to collisions action, one can provide a definition of *admissible* group action on $\Lambda$, imposing the following hypotheses on $G$.

**Definition 2.25** (Admissible action, Definition 2.12 [43])**.** We say that the action of a finite group $G$ on $\Lambda$ is *admissible* if:

- $\ker\tau \bigcap \ker\sigma = 1$. This shows in particular that $G$ is a subgroup of $O(2)\times\Sigma_n$ through canonical isomorphism considering the product homomorphism $(\tau, \sigma)$;
- $|\ker\rho \bigcap \ker\sigma| < 2$ and the non-trivial element in $\ker\rho \bigcap \ker\sigma$ acts as a reflection in time;
- $\ker\tau \bigcap \ker\rho = 1$. This shows in particular that $G$ is a subgroup of $O(2) \times O(d)$.

*Remark* 2.26. It is important to notice that an admissible action is not necessarily non reducible, nor not bound to collisions.

### 2.5.4 Classification of the group action and fundamental domain

From now on, we assume that $G$ generates an admissible action on $\Lambda$ as described in Definition 2.25. Our goal is to classify finite groups based on the *type of action* they exert on $\Lambda$. Given that $\tau(G)$ is a finite subgroup of $O(2)$, Proposition 2.10 and Remark 2.16 suggest that the most straightforward and meaningful classification considers the action $G$ induced on $\mathbb{T}$. Specifically, $G/\ker\tau$ is itself a finite subgroup of $O(2)$, and it is useful to examine how $G$'s action on $\chi$ relates to $G/\ker\tau$'s action on $\chi^{\ker\tau}$.
Let us begin with the definition of $\bar{G}$ as $\bar{G} := G/\ker\tau$. Recall that $G$ acts on the configuration space $\chi$ through the homomorphisms $\rho : G \to O(d)$ and $\sigma : G \to \Sigma_n$. As one has that $\ker\tau \triangleleft G$ and using Remark 2.14, it follows that $\bar{G}$ acts on the fixed space $\chi^{\ker\tau}$. Moreover, as the quotient $\bar{G}$ is a subgroup of $O(2)$, it is an advantageous refinement of the whole group $G$; as a matter of fact, it is easier to work with $\bar{G}$-*equivariant* loops in $\chi^{\ker\tau}$ as it is confirmed by the following proposition.

**Proposition 2.27** (Proposition 2.16, [60])**.** *The $H^1$ loop spaces*

$$\Lambda^G = H^1(\mathbb{T};\chi)^G, \quad \Omega^{\bar{G}} := H^1(\mathbb{T};\chi^{\ker\tau})^{\bar{G}}$$

*coincide pointwise.*

As one has that $\tau(G) \simeq \bar{G}$, then $\bar{G}$ is a finite subgroup of $O(2)$; from Proposition 2.10, it follows that $\bar{G}$ is either *cyclic* or *dihedral*. One can add another definition:

**Definition 2.28** (Brake)**.** If $H$ is a subgroup of $O(2)$ and $H = \{1, S\}$, with $S$ a reflection matrix, then we say that $H$ is a *brake subgroup* of $O(2)$.
Furthermore, if $x(t) \in \Lambda$ and there exists $g \in G$ which acts as a reflection on $\mathbb{T}$ and such that

$$x(\tau(g)t) = x(t), \quad \text{for every } t \in \mathbb{T},$$

we say that $x(t)$ is a *brake orbit*.

Now, a complete classification of a group action on $\Lambda$ can be given:

**Definition 2.29** (Classification of the action, Definition 2.14 [43])**.** We classify the action of a finite group $G$ on $\Lambda$ in this way:

- if the quotient $\bar{G}$ acts trivially on the orientation of $\mathbb{T}$, then $\bar{G}$ is cyclic and we say that the action of $G$ on $\Lambda$ is of *cyclic type*. In this case, $\bar{G}$ is a subgroup of $SO(2)$ and only contains rotations;
- if $\bar{G}$ is made by a single reflection on $\mathbb{T}$, then we say that the action of G on $\Lambda$ is of *brake type*. In particular, $\bar{G}$ is a cyclic group of order 2;
- if none of the previous is satisfied, we say that the action of $G$ on $\Lambda$ is of *dihedral type* and $\bar{G}$ is a dihedral subgroup of $O(2)$.

*Remark.* From Definition 2.12, since $G$ acts on $\mathbb{T}$ using $\tau$, if $t \in \mathbb{T}$, the isotropy group of $t$ in G is

$$G_t = \{g \in G : \tau(g)t = t\}$$

and it is also a subgroup of $G$.

**Definition 2.30** ($\mathbb{T}$-isotropy subgroups, Definition 2.15 [43])**.** The isotropy subgroups of the action of $G$ on $\mathbb{T}$ through $\tau$ are called the $\mathbb{T}$-*isotropy subgroups* of $G$.

According to the previous definition, the $\mathbb{T}$-isotropy subgroups of $G$ correspond to the isotropy subgroups of the action of $\bar{G}$ on $\mathbb{T}$. This insight is quite valuable because it allows us to classify

the $\mathbb{T}$-isotropy subgroups of $G$ as subgroups of $O(2)$ using Proposition 2.10.
Moreover, we recall briefly that a subgroup $H$ of $G$ is defined *maximal* if a proper subgroup $K$ that contains $H$ strictly does not exist. Identifying the maximal $\mathbb{T}$-isotropy subgroups of $G$ is crucial, as they help in identifying a subinterval $\mathbb{I}$ of $\mathbb{T}$ sufficient to characterise a complete $G$-equivariant loop.

*Remark* 2.31. The number of distinct $\mathbb{T}$-isotropy subgroups fully determines the action type of $G$. First, note that $\ker\tau$ is always a $\mathbb{T}$-isotropy subgroup of $G$ since $G$ is finite. Considering the action of $\bar{G}$, we examine the isotropy subgroups of this action, which are finite subgroups of $O(2)$. By definition, no $\bar{G}_t$ can include rotations. Let $l$ denote the number of distinct $\mathbb{T}$-isotropy subgroups; the previous observation leads to the following characterisation:

- $l = 1$, i.e., the maximal $\mathbb{T}$-isotropy subgroup is $\ker\tau \iff$ the action is of *cyclic type*;
- $l = 2$, i.e., the $\mathbb{T}$-isotropy subgroups are $\ker\tau$ and a maximal one $\iff$ the action is of *brake type*. In this case, the maximal $\mathbb{T}$-isotropy subgroup coincides with $G$;
- $l \geq 3$, i.e., the $\mathbb{T}$-isotropy subgroups are either maximal or $\ker\tau \iff$ the action is *dihedral type*. In this case, every maximal $\mathbb{T}$-isotropy subgroup is cyclic of order 2. This follows from considering the isotropy subgroups of the action of $\bar{G}$ and noticing that no rotations can belong to any $\bar{G}_t$. Note that the product of two reflections is a rotation.

Once the distinct $\mathbb{T}$-isotropy subgroups of $G$ are identified, it becomes possible to define a subinterval of $\mathbb{T}$ sufficient to describe the action of $G$ on the entire $\mathbb{T}$. This step is crucial for the minimisation process, as it enables the transition from analysing periodic loops to minimising fixed-end paths with initial and final conditions set at the endpoints of this subinterval.

**Definition 2.32** (Fundamental domain, Definition 2.17 [43])**.** The *fundamental domain* $\mathbb{I} \to \mathbb{T}$ of the action of $G$ on $\mathbb{T}$ is defined as follows:

- if the action of $G$ is cyclic, then $\mathbb{I}$ is the closed interval which connects the instant $t = 0$ with its image $\tau(g^{-1})0$, where $g$ is the representative of a cyclic generator of $\bar{G}$;
- if the action of $G$ is brake or dihedral, then $\mathbb{I}$ is a closed interval whose extrema are two distinct elements of $\mathbb{T}$ generating two distinct maximal $\mathbb{T}$-isotropy subgroups, such that no other instants related to maximal $\mathbb{T}$-isotropy subgroups are included in $\mathbb{I}$. Note that, if the action is of brake type, the unique maximal $\mathbb{T}$-isotropy subgroup is $G$ itself.

It follows that there are $|\bar{G}|$ of such domains and we have that

$$\mathbb{T} = \bigcup_{[g]\in\bar{G}} \tau(g^{-1})\mathbb{I} \quad \text{and} \quad |\mathbb{I}| = \frac{|\mathbb{T}|}{|\bar{G}|}.$$

## 2.6 $G$-equivariant minimisers of the action functional

In this Section, we introduce the theoretical framework that allows the transition from optimising the action functional on loops defined over the entire period to addressing a generalised Bolza problem. Subsequently, we delve into the numerical methods that generate $G$-equivariant periodic orbits for the symmetrical $n$-body problem.

**Proposition 2.33** (Proposition 3.1, [60])**.** *Consider a finite group $G$ and the usual representations $\rho, \tau$ and $\sigma$. Assume that $\bar{G}$ is not cyclic. Define $\mathbb{I} = [t_0, t_1]$ as the fundamental domain of the action of $G$ and let $H_0$ and $H_1$ be the two maximal $\mathbb{T}$-isotropy subgroups associated respectively with the instants $t_0$ and $t_1$. Then, the space $\Omega^{\bar{G}} = H^1(\mathbb{T}; \chi^{\ker\tau})^{\bar{G}}$ is homeomorphic to the space of fixed-end paths*

$$Y = Y(H_0, H_1, \ker\tau) = \{x \in H^1(\mathbb{T}; \chi^{\ker\tau}) : x(t_0) \in \chi^{H_0},\, x(t_1) \in \chi^{H_1}\}.$$

Although Proposition 2.33 is not true if the action is of cyclic type, one can prove an analogous result. Since $\bar{G}$ is cyclic, we have $\bar{G} = \langle g \rangle$. From Definition 2.32, it follows that $\mathbb{I} = [0, \tau(g^{-1})0]$, and that the unique maximal $\mathbb{T}$-isotropy subgroup is $\ker \tau$. Hence, one cannot define the two manifolds $\chi^{H_i}$ as there is a different constraint. For the sake of simplicity, let $t_1 = \tau(g^{-1})0$ so we have that $\mathbb{I} = [0, t_1]$. Now we define $Y$ as

$$Y := \{x \in H^1([0,t_1]; \chi^{\ker \tau}) : gx(0) = x(t_1)\},$$

where the action of $g$ is clearly on the space and indexes, via $\rho$ and $\sigma$. From this definition, it follows that the space $Y$ is homeomorphic to $\Omega^{\langle g \rangle}$, and hence a statement analogous to Proposition 2.33 holds in the cyclic case.
Now, for any group action type, the restriction of the action functional to the fundamental domain $\mathbb{I}$ on $Y$ can be defined in this way

$$\begin{aligned} \mathcal{A}_{\mathbb{I}} : Y &\longrightarrow \mathbb{R} \cup \{+\infty\} \\ y &\longmapsto \mathcal{A}_{\mathbb{I}}(y) := \int_{\mathbb{I}} L(y(t), \dot{y}(t))\, \mathrm{d}t. \end{aligned} \tag{2.4}$$

With a similar proof to Proposition 2.33, let $y \in Y$ and $y = \pi(x)$, where $x$ is the concatenation of symmetrised paths $gy$, and thanks to the equivariance of the action functional proved in Proposition 2.17, it easily follows that

$$\mathcal{A}(x) = |\bar{G}| \mathcal{A}_{\mathbb{I}}(y). \tag{2.5}$$

**Lemma 2.34** (Lemma 3.2, [60]). *The functional $\mathcal{A}_{\mathbb{I}}$ is coercive on $Y$ if and only if $\chi^G = \{0\}$.*

**Definition 2.35** (Local minimiser). We say that $y \in Y$ is a *local minimiser* of $\mathcal{A}_{\mathbb{I}}$ if, for any $\varphi \in Y$ and for any $\varepsilon$ sufficiently small we have

$$\mathcal{A}_{\mathbb{I}}(x) \leq \mathcal{A}_{\mathbb{I}}(x + \varepsilon\varphi).$$

**Definition 2.36.** Let $G$ be a finite group acting orthogonally on $\mathbb{R}^d$ through the representation $\rho$ and $\mathbb{S} \subseteq \mathbb{R}^d$ be a circle centred at the origin. We say that $\mathbb{S}$ is *rotating under a subgroup $H$ of $G$* if the restriction $h|_{\mathbb{S}}$ of the action of every element $h \in H$ is a rotation of an angle $\theta_h$. In particular, in this case $\mathbb{S}$ is invariant under the action of $H$.

**Definition 2.37.** Let $G$ be a finite group acting on the index set $\{1, \dots, n\}$ through the representation $\sigma$. Fix $i \in \{1, \dots, n\}$ and let $H$ be a subgroup of $G$, let $K_i$ denote the isotropy subgroup of the index $i$ in $H$, i.e.,

$$G_i := \{h \in K : hi = i\}.$$

A circle $\mathbb{S} \subseteq \mathbb{R}^d$ centred at the origin is termed *rotating for $i$ under $H$* if

- $\mathbb{S}$ is rotating under $H$;
- $\mathbb{S} \subseteq (\mathbb{R}^d)^{K_i}$.

**Definition 2.38** (Definition 10.1 [35]). We say that a group $G$ *acts on $\Lambda$ with the rotating circle property* if, for every $\mathbb{T}$-isotropy subgroup $G_t$ and for at least $n-1$ indexes $i \in \{1, \dots, n\}$, there exists a circle $\mathbb{S} \subseteq \mathbb{R}^d$ centred at the origin rotating for $i$ under $G_t$.

**Lemma 2.39** (Theorem 10.3, [35]). *Assume that $\ker \tau$ acts on $\Lambda$ with the rotating circle property. Then, any local minimiser $y \in Y$ of $\mathcal{A}_{\mathbb{I}}$ is free of interior collisions, i.e., $y(t) \notin \Lambda$, for any $t \in (t_0, t_1)$. Moreover, if also $H_0$ and $H_1$ act on $\Lambda$ with the rotating circle property, $y(t_0), y(t_1) \notin \Lambda$ and the minimiser is always collision-less.*

*Remark* 2.40. Note that, in the previous lemma, we do not differentiate between the cyclic and non-cyclic action cases. The reason is that when $\bar{G}$ is cyclic, then $H_0 = H_1 = \ker \tau$, and the absence

of collisions throughout the entire period results from the fact that $\ker\tau$ acts with the rotating circle property. Furthermore, one can observe that if $\ker\tau = 1$, the rotating circle property holds trivially, so any cyclic group $G$ produces $G$-equivariant collision-free solutions as action minimisers.

**Theorem 2.41** (Theorem 3.9, [60]). *Assume that $\chi^G = \{0\}$ and that $\ker\tau, H_0$ and $H_1$ act on $\Lambda$ with the rotating circle property. Then, any local minimiser of $\mathcal{A}_{\mathbb{I}}$ is collision-less. In particular, there exists a classical solution of the $G$-equivariant $n$-body problem.*

The theoretical knowledge developed in [35] and [60], and outlined in the previous Sections, shows some precise assumptions on $G$ that ensure the existence of collision-less periodic solutions of the $G$-equivariant $n$-body problem. Following [60], one can numerically find some examples of such solutions as critical points of the action functional.
As Theorem 2.41 states, in order to find collision-less minimisers, the fundamental hypotheses are related to the maximal $\mathbb{T}$-isotropy subgroups $H_0$ and $H_1$, and $\ker\tau$. From now on, we assume that $G$ satisfies these requirements. Furthermore, one can deduce from Remark 2.31 that the action type has a strong influence on the nature of these subgroups; therefore, when working on the numerical implementation, one needs to adapt the optimisation routine accordingly to the action type. In Sections 2.6.1 and 2.6.2, the projectors on these subgroups which ensure that, after any iteration, the path satisfies the required boundary conditions are presented in the different action types. In Section 2.6.3, we present two equivalent ways to numerically model the action functional depending on the aim of research.

### 2.6.1 Numerical optimisation: dihedral or brake action

Assume that the action of $G$ on $\Lambda$ is either dihedral or brake type and meets the conditions of Theorem 2.41. Initially, we give the definition of a finite-dimensional approximation of the space $Y$ described in Proposition 2.33. The space $Y$ already includes two finite-dimensional components: the initial manifold $\chi^{H_0}$ and the final manifold $\chi^{H_1}$, making it trivially diffeomorphic to the space

$$H^1(\mathbb{I};\chi^{\ker\tau}) \times \chi^{H_0} \times \chi^{H_1}.$$

To numerically optimise the problem, the most straightforward approach is to approximate each function in $H^1(\mathbb{I};\chi^{\ker\tau})$ using a truncated Fourier series, while the manifolds $\chi^{H_i}$ are already embedded in $(\mathbb{R}^d)^n$. Consequently, our optimisation process will occur in the space

$$Y_F := ((\mathbb{R}^d)^n)^F \times (\mathbb{R}^d)^n \times (\mathbb{R}^d)^n,$$

where $F \in \mathbb{N}$ stands for the length of the truncated Fourier series. The boundary conditions on $Y$ can be enforced using a pair of projectors as outlined in Definition 2.12. Recall that, as we have that the action is either dihedral or brake, $H_0$ and $H_1$ are both cyclic of order 2, with $h_0$ and $h_1 \in G$ as their generators (see also Remark 2.31). The projectors are then defined as the linear maps

$$\begin{aligned} \pi_i : (\mathbb{R}^d)^n &\longrightarrow \chi^{H_i} \\ v &\longmapsto \pi_i v := \left(\frac{1+h_i}{2}\right) v. \end{aligned}$$

On the other hand, the functions, which we aim to approximate pointwise, belong to the space $\chi^{\ker\tau}$, with the centre of mass fixed at the origin. Therefore, we need also to introduce the projector $\pi_{\ker\tau} : (\mathbb{R}^d)^n \longrightarrow \chi^{\ker\tau}$, defined accordingly, along with a linear map $\pi_c$ that projects any configuration onto $\chi$.

### 2.6.2 Numerical optimisation: cyclic action

Assume that $\bar{G} = \langle g \rangle$ is a cyclic group, and recall that in this situation, a different constraint is required on the fundamental domain (see also Remark 2.40). The approximation space remains

$Y_F$, as defined for the brake/dihedral case, but we clearly need to define a different projector to ensure that, after each iteration, the path meets the required boundary conditions. The projector in this case operates as follows:

$$\pi_g : (\mathbb{R}^d)^n \times (\mathbb{R}^d)^n \longrightarrow V_g$$
$$(v, w) \longmapsto \pi_g(v, w) \coloneqq \left(\frac{1}{2}(v + g^{-1}w), \frac{1}{2}(gv + w))\right).$$

where $V_g \coloneqq \{(v, w) \in (\mathbb{R}^d)^n \times (\mathbb{R}^d)^n : gy = w\}$. Consequently, any arbitrary boundary condition $(v, w)$ is transformed into another that meets the symmetry constraint imposed by a cyclic action on the boundary.

### 2.6.3 Numerical optimisation: action

Following [62], there are two possible approaches for discretising the action functional $\mathcal{A}$ defined in (2.2).

**Discretisation through points.** The first approach explained in this paragraph involves discretising the integral operator using an appropriate quadrature technique and then approximating the derivatives using finite differences. Specifically, we consider a uniform grid $0 = t_0 < t_1 < \ldots < t_{M+1} = \pi$, where $t_k$ represents the $k$-th time instant within the interval $\mathbb{I}$. We define the function $G(t)$ as follows:

$$G(t) = G(x_1(t), x_2(t), \ldots, x_n(t)) = \sum_{i=1}^{n} \frac{m_i}{2} \|\dot{x}_i\|^2 + \sum_{i<j} \frac{m_i m_j}{\|x_i - x_j\|}$$

Next, we apply the composite trapezoidal rule to approximate the integral over the $M+1$ intervals, yielding:

$$\mathcal{A}_{\mathbb{I}} = \int_0^{\pi} G(t)\, dt \sim \frac{h}{2}\left[G(t_0) + G(t_{M+1}) + 2\sum_{k=0}^{M} G(t_k)\right]$$

where $h = \frac{\pi}{M+1}$, and $y_i^k \sim x_i(t_k) \in \mathbb{R}^d$ for $i = 1, \ldots, n$ and $k = 0, \ldots, M+1$. Using a forward difference formula for the approximation of the derivative with respect to time, we obtain the following approximation for the action functional:

$$\begin{aligned}\mathcal{A}_{\mathbb{I}} \sim \sum_{k=1}^{M}\left[\sum_{i=1}^{n} \frac{m_i}{2}\frac{\|y_i^{k+1} - y_i^k\|^2}{h} + h\sum_{i<j}\frac{m_i m_j}{\|y_i^k - y_j^k\|}\right] \\ + \sum_{i=1}^{n}\frac{m_i}{4h}\left[\|y_i^1 - y_i^0\|^2 + \|y_i^{M+2} - y_i^{M+1}\|^2\right] \\ + \frac{h}{2}\sum_{i<j} m_i m_j \frac{\|y_i^0 - y_j^0\| + \|y_i^{M+1} - y_j^{M+1}\|}{\|y_i^0 - y_j^0\|\|y_i^{M+1} - y_j^{M+1}\|} = \mathcal{A}_h^{(1)}\end{aligned} \tag{2.6}$$

where we introduce appropriate ghost points $y_i^{M+2}$. For further details, see [43, 62–66].
As a result, we need to optimise the objective function:

$$f_1(y_i^k) = \mathcal{A}_h^{(1)}, \quad i = 1, \ldots, n,\ k = 0, \ldots, M+1 \tag{2.7}$$

The numerical solution is represented by a block real matrix $(y_i^k)_{i,k} \in \mathbb{R}^d$, with $i = 1, \ldots, n$ and $k = 0, \ldots, M+1$, having dimensions $d \times n \times (M+2)$.

**Discretisation through Fourier coefficients.** Another approach is using Fourier coefficients; for further details, see [40, 42, 67, 68]. In our specific case following [43], one approximates the

solution as the sum of a linear part, in particular the segment between the initial configuration $x_0$ and the final configuration $x_1$, and a truncated Fourier series, in details

$$x(t) = x_0 + \frac{t}{\pi}(x_1 - x_0) + \psi(t), \tag{2.8}$$

where $t \in [0, \pi]$ and $\psi(0) = \psi(\pi) = 0$. We approximate the function $\psi(t)$ using a truncated Fourier polynomial where $F \in \mathbb{N}$ stands for the length of Fourier polynomials $\psi(t) = \sum_{k=1}^{F} A_k \sin(kt) \in \mathbb{R}^{(d \times n)}$, for $t \in [0, \pi]$. In this expression, we only use sine functions to ensure that the approximate solution remains periodic. Substituting the expression (2.8) into the action functional (2.2) results in:

$$f_2(x_0, x_1, A_k) = \mathcal{A}_h^{(2)}, \quad k = 1, \ldots, F. \tag{2.9}$$

This is a function in $(F+2) \times d \times n$ variables which can be minimised using an appropriate optimisation technique. For further details, see [43].

In both models, we introduce a scalar function $f_i$, for $i = 1, 2$, given by either (2.7) or (2.9), in the variables $x$, represented by $y_i^k$ or $\{x_0, x_1, A_k\}$, which correspond to the unknowns in our problem. The functions $f = f_i$, for $i = 1, 2$, to be maximised or minimised, are typically referred to as the objective function.
There can be various methods and approaches used to collect as many critical points as possible of the action functional, defined in (2.6). For more details, see [62]. These critical points include minimum, maximum and saddle points. Once obtained this large set, it is fundamental to be able to distinguish them or to consider them equal when they give the same orbit. As a matter of fact, an overview of a large scale of possible orbits of the $n$-body dynamical system would give us more pieces of information on how this system develops. Moreover, using the orbits obtained numerically, we might be able to discover new properties and previously hidden patterns.

# Chapter 3

# Finding critical points

In this Chapter, we present the numerical optimisation routine and the selection of algorithms used to solve our problem. The Chapter has a dual purpose. The first part (from Section 3.1 to Section 3.2.4) is primarily expository and provides a structured review of existing numerical optimisation techniques relevant to the computation of critical points of the action functional. These Sections survey classical local methods, constrained optimisation strategies, and metaheuristic approaches, with the aim of motivating the choice of algorithms suitable for highly nonconvex and multimodal problems. No claim of originality is made for these methods, which are drawn from the established optimisation literature; the review presented here closely follows and summarises the discussion developed in [62].
The original contribution of this Chapter lies in the practical adaptation, implementation, and integration of existing algorithms within a unified computational framework tailored to the equivariant $n$-body problem. In particular, the Multi-Population Adaptive Inflationary Differential Evolution Algorithm (MP-AIDEA [69]), originally proposed in the literature, is reimplemented in the `Julia` programming language [70] and combined with the Lattice domain-decomposition strategy. This coupling results in a novel hybrid optimisation framework that enables the systematic detection of multiple critical points and supports efficient parallel execution. The effectiveness of this approach is demonstrated in Section 3.3.

## 3.1 Numerical optimisation routine

Following [43], we outline the following steps for the numerical optimisation routine:

**Step I** Consider a random sequence of vectors

$$A_0, A_1, \ldots, A_{F+1} \in \mathbb{R}^{(d\times n)},$$

where $A_0$ and $A_{F+1}$ represent the initial and final points of the ansatz. Since this is a *Bolza*-conditioned problem, project these vectors onto $\chi$ and then onto the boundary manifolds. For further details, see [43].

**Step II** Discretise the action functional defined in (2.2) using Fourier coefficients as outlined in Equation (2.8). From this, construct an initial guess $y(t) := x(t) + s(t)$, where $x(t)$ denotes the straight line segment between $x_0$ and $x_1$:

$$x(t) = x_0 + \frac{t}{\pi}(x_1 - x_0), \quad \text{for } t \in [0, \pi],$$

and $s(t)$ is the truncated Fourier series:

$$s(t) \coloneqq \sum_{k=1}^{F} A_k \sin(kt), \quad \text{for } t \in [0, \pi].$$

**Step III** Discretise the interval $[0, \pi]$ with a grid $(t_h)$, and project any configuration $y(t_h)$ onto the symmetric configuration space in order to satisfy the Bolza conditions of the problem.

**Step IV** Express the action functional on the interval $[0, \pi]$, evaluated at $y$ as in expression (2.9). This is now a function of $(F + 2) \times d \times n$ variables, which are the parameters to optimise. The choice of optimisation algorithm is discussed in the following paragraph, but one could, for instance, use a *steepest descent method* or a *Newton method* to move from $y$ to $y_{\text{new}}$. Once this is done, project $y_{\text{new}}$ back into the symmetric configuration space.

**Step V** The optimised path, now defined on the fundamental domain $\mathbb{I}$ through Fourier coefficients, can be extended to the entire period using the symmetries of the group and an inverse Fourier transform (for more details, see [43]).

## 3.2 Choice of the algorithm

In this Section, we want to deeply discuss the choice of the algorithm in **Step IV**. It is well known that there are no universal algorithms to solve a specific problem, but rather a collection of algorithms, each of which is tailored to a particular optimisation problem.
Our aim is to present a brief survey of the possible optimisation methods. With the advent of artificial intelligence, particularly in the realms of machine learning and deep learning, the optimisation techniques have gained renewed interest and importance while, historically, optimisation techniques have been pivotal in areas ranging from operations research to economics. As a result, it is important to highlight the research developments and improvements achieved in recent years. In this Section, several well-known methods, such as *Gradient Methods* and *Newton's Methods*, are not included in the analysis. These two categories represent fundamental optimisation techniques that have been extensively generalised and refined over time. In the upcoming Sections, we delve into various optimisation techniques and their applications. Section 3.2.1 focuses on trust region methods, laying the groundwork for more specialised topics; see for example [71–74]. In Section 3.2.2, we explore constrained optimisation, with particular attention given to penalisation methods, nonlinear optimisation, and the integral roles of merit functions and filters in the optimisation process. Lastly, in Section 3.2.3, we pivot to metaheuristic methods, discussing techniques such as tabu search, simulated annealing, and genetic algorithms. Each of these Sections aims to provide comprehensive insights into these respective methods and their implications.
Anticipating the developments that follow, Section 3.2.4 presents practical criteria and a decision workflow for *selecting* and *combining* optimisation methods, highlighting the role of hybrid strategies when a single technique is insufficient. Section 3.2.5 then presents the *Multi-Population Adaptive Inflationary Differential Evolution Algorithm* (MP-AIDEA), detailing its multi-population design, adaptive controls, and restart/local-search mechanisms for navigating multimodal landscapes. Section 3.2.6 introduces the *Lattice* domain-decomposition scheme, which recursively partitions the search space, guided by the empirical distribution of discovered minimum points, to focus computation where it is most informative. Finally, Section 3.3 applies the combined MP-AIDEA&Lattice framework to the symmetrical $n$-body problem, demonstrating how the hybrid approach uncovers multiple periodic orbits while enabling parallel execution across subregions.

Most of the methods described above primarily aim to identify minimum points. However, when the objective is to locate saddle points or, more generally, critical points with a Morse index greater than two, several alternative strategies can be employed. One common approach is to first use standard minimisation techniques to find local minimum points, and then apply additional procedures to explore Mountain-Pass-type critical points, that is, saddle points with a Morse index equal to one. Another strategy involves minimising the norm of the gradient, which allows for the identification of generic critical points, including saddle points. Beyond these techniques, specific algorithms can be designed explicitly for the detection of saddle points. An example of such an approach is presented in Chapter 5, where we describe a bisection algorithm for computing numerical Mountain Pass solutions. Overall, metaheuristic methods remain among the most flexible and powerful tools for exploring the broader landscape of critical points beyond local or global minimum points.

### 3.2.1 Trust-region methods

A general minimisation algorithm, starting from an initial guess $x_0$, generates a sequence of iterates $\{x_k\}_{k=0}^{\infty}$. The process terminates either when no further improvement can be achieved or when the solution is considered sufficiently accurate. To obtain the next iterate $x_{k+1}$ from the current point $x_k$, the algorithm uses information about the objective function $f$ evaluated at $x_k$, and possibly data from previous iterations such as $x_0, x_1, \ldots, x_{k-1}$. Based on this information, a new iterate $x_{k+1}$ is computed so that the function value at $x_{k+1}$ is lower than at $x_k$.

There are two main frameworks for updating the iterates. In the *line search* approach, the algorithm first selects a search direction, denoted by $p_k$. Typical choices include $p_k = -\nabla f(x_k)$ for gradient-based methods or $p_k = -H_k(x_k)^{-1}\nabla f(x_k)$ for Newton-type methods, where $H_k(x_k)$ represents the Hessian matrix of $f$ at $x_k$. A new point is then sought along this direction by moving from $x_k$ to a position that reduces the value of $f$. The step length $\alpha > 0$ is determined by approximately solving the one-dimensional optimisation problem

$$\min_{\alpha>0} f(x_k + \alpha p_k).$$

Although solving this problem exactly would yield the optimal progress along $p_k$, doing so is usually computationally expensive and unnecessary. Instead, practical implementations evaluate a few candidate step sizes and select one that sufficiently decreases the objective function. Once the new point is identified, the algorithm updates the search direction and step size, repeating this process until convergence.

In the *trust-region* framework, the available information about the objective function $f$ is used to construct a local approximation, denoted by $m_k$, that reflects the behaviour of $f$ near the current iterate $x_k$. Since this model may not accurately capture the function's behaviour far from $x_k$, the search for an improved point is restricted to a neighbourhood around $x_k$, referred to as the *trust region*. The trial step $p$ is then obtained by solving the following constrained optimisation problem:

$$\min_{p} m_k(x_k + p) \quad \text{subject to } x_k + p \text{ lying within the trust region.}$$

If the computed step fails to yield a sufficient reduction in $f$, the trust region is considered too large, and its radius is decreased before attempting another iteration. Typically, the trust region is represented as a ball

$$\|p\|_2 \leq \Delta,$$

where $\Delta > 0$ denotes its radius, though alternative geometries such as ellipsoids or rectangular domains may also be employed. The model function $m_k$ is commonly derived from a Taylor expansion of $f$ around $x_k$, given by

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T H_k(x_k) p + \mathcal{O}(\|p\|^3),$$

where $f_k = f(x_k)$ and $g_k = \nabla f(x_k)$. By approximating the Hessian $H_k(x_k)$ with a symmetric matrix $B_k$, we define the quadratic model

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p.$$

The difference between $m_k(p)$ and $f(x_k + p)$ is on the order of $\mathcal{O}(\|p\|^2)$, which becomes negligible for small steps. In trust-region Newton methods, setting $B_k = H_k(x_k)$ results in a model whose approximation error is of order $\mathcal{O}(\|p\|^3)$, thereby ensuring high accuracy in a neighbourhood of the current point.
Trust-region methods are widely recognised for their robustness and flexibility, as they rely on only mild assumptions regarding the matrix $B_k$, namely, symmetry and uniform boundedness. These properties make them applicable to a broad class of optimisation problems.
In details, at each iteration, the step $p_k$ is obtained by solving the *trust-region subproblem*:

$$\min_{p_k \in \mathbb{R}^n} m_k(p_k) = f_k + g_k^T p_k + \frac{1}{2} p_k^T B_k p_k \quad \text{s.t.} \quad \|p_k\| \leq \Delta_k, \tag{3.1}$$

where $\Delta_k > 0$ denotes the radius of the trust region. The optimal step $p_k^\star$ that solves (3.1) satisfies the following system:

$$(B_k + \lambda_k I) p_k^\star = -g_k,$$

for some scalar $\lambda_k \geq 0$, where $B_k + \lambda_k I$ is positive semidefinite, and the complementarity condition $\lambda_k(\Delta_k - \|p_k^\star\|) = 0$ is fulfilled.

*Remark.* If $B_k$ is positive definite and the unconstrained minimiser satisfies $\|B_k^{-1} g_k\| \leq \Delta_k$, then the solution of (3.1) corresponds to the unconstrained minimiser

$$p_k^B = -B_k^{-1} g_k,$$

which is referred to as the *full step*.

**Trust-region radius $\Delta_k$.** A central aspect of trust-region methods is the determination of the radius $\Delta_k$ at each iteration. The choice of this radius is guided by how well the model function $m_k$ approximates the true objective function $f$ in previous steps. To assess this agreement, a ratio $\rho_k$ is introduced, defined as

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}, \tag{3.2}$$

where the numerator represents the *actual reduction* in the objective function, and the denominator corresponds to the *predicted reduction* according to the model.
If $\rho_k < 0$, indicating that $f(x_k + p_k)$ exceeds $f(x_k)$, the trial step is rejected. When $\rho_k \approx 1$, the model $m_k$ provides an accurate local approximation of $f$, and the trust-region radius is typically enlarged for the next iteration. Conversely, if $0 < \rho_k \ll 1$, the current region size is maintained, while values of $\rho_k$ close to zero suggest poor agreement, prompting a reduction of the trust-region radius in the subsequent step.
We denote $\hat{\Delta}$ as an overall bound on the step lengths.

*Remark.* One increases the radius only in the case in which $\|p_k\|$ reaches the boundary of the trust region. When the step does not reach the boundary the current value of $\Delta_k$ is considered not to be interfering with the progress of the algorithm, therefore it is not altered.

**The Cauchy point.** Although, in theory, one seeks the exact minimiser of the subproblem (3.1), it is sufficient for global convergence to compute an approximate solution $p_k$ that lies within the trust region and yields a *significant reduction* in the model function. This notion of sufficient

**Algorithm 1** Trust region method

**Input:** maximum trust region radius $\hat{\Delta} > 0$, initial radius $\Delta_0 \in (0, \hat{\Delta})$, threshold $\eta \in [0, \frac{1}{4})$
**Initialise:** $x_0$, $\Delta_0$
**Output:** Approximation of the solution $x_k$
**for** $k = 0, 1, 2, \ldots$ **do**
    Compute $p_k$ by approximately solving the trust region subproblem (3.1)
    Evaluate the ratio $\rho_k$ using (3.2)
    **if** $\rho_k < \frac{1}{4}$ **then**
        $\Delta_{k+1} \leftarrow \frac{1}{4}\Delta_k$
    **else if** $\rho_k > \frac{3}{4}$ **and** $\|p_k\| = \Delta_k$ **then**
        $\Delta_{k+1} \leftarrow \min(2\Delta_k, \hat{\Delta})$
    **else**
        $\Delta_{k+1} \leftarrow \Delta_k$
    **end if**
    **if** $\rho_k > \eta$ **then**
        $x_{k+1} \leftarrow x_k + p_k$
    **else**
        $x_{k+1} \leftarrow x_k$
    **end if**
**end for**
**return** Current iterate $x_k$ as approximate solution.

reduction can be quantified through the so-called *Cauchy point*, denoted by $p_k^c$. The Cauchy point is defined through the following construction.
First, consider the vector $p_k^s$ that solves a simplified, linearised version of the trust-region subproblem:

$$p_k^s = \arg\min_{p_k \in \mathbb{R}^n} \; f_k + g_k^T p_k \quad \text{s.t.} \quad \|p_k\| \leq \Delta_k. \tag{3.3}$$

Next, determine a scalar $\tau_k > 0$ that minimises $m_k(\tau p_k^s)$ while satisfying the trust-region constraint:

$$\tau_k = \arg\min_{\tau \geq 0} \; m_k(\tau p_k^s) \quad \text{s.t.} \quad \|\tau p_k^s\| \leq \Delta_k. \tag{3.4}$$

The Cauchy point is then obtained as

$$p_k^c = \tau_k p_k^s.$$

The Cauchy step $p_k^c$ can be computed inexpensively and plays a crucial role in determining whether a candidate solution of the trust-region subproblem is acceptable. In particular, it can be shown that global convergence of trust-region methods is guaranteed if the computed step $p_k$ provides a decrease in the model $m_k$ that is at least a fixed positive fraction of the reduction achieved by the Cauchy step.

*Remark.* The Cauchy point itself is not typically adopted as the iteration step, since doing so would effectively correspond to applying the steepest descent method with a specific step length. It is well known that the steepest descent approach exhibits slow convergence even when optimal step sizes are used.

Consequently, trust-region algorithms generally compute the Cauchy point first and then attempt to refine it. This refinement is achieved by designing the algorithm to select the full step $p_k^B = -B_k^{-1} g_k$ when $B_k$ is positive definite and $\|p_k^B\| \leq \Delta_k$. Two common strategies for approximating the solution of (3.1) based on this principle are:

1. *Dogleg method:* used when the model Hessian $B_k$ is positive definite;

2. *Two-dimensional subspace minimisation:* applied when $B_k$ is indefinite, requiring an estimate of the most negative eigenvalue of $B_k$.

#### 3.2.1.1 The Doglet method

As previously discussed, this method is applicable when the matrix $B_k$ is positive definite.
It is instructive to examine how the trust-region radius $\Delta_k$ influences the solution $p_k^\star(\Delta_k)$ of the subproblem (3.1). When $B_k$ is positive definite, the unconstrained minimiser of the model $m_k$ is given by

$$p_k^B = -B_k^{-1} g_k.$$

If this point satisfies the trust-region constraint, it is also the optimal solution, i.e.

$$p_k^\star(\Delta_k) = p_k^B, \quad \text{whenever } \Delta_k \geq \|p_k^B\|.$$

When $\Delta_k$ is relatively small compared to $\|p_k^B\|$, the constraint $\|p_k\| \leq \Delta_k$ limits the influence of the quadratic term in $m_k$. In such cases, the optimal step can be well approximated by

$$p_k^\star(\Delta_k) \approx -\Delta_k \frac{g_k}{\|g_k\|}, \quad \text{for small } \Delta_k.$$

For intermediate values of $\Delta_k$, the solution $p_k^\star(\Delta_k)$ typically follows a curved trajectory between these two extremes.
The *Dogleg method* provides an efficient approximation of this curved path by replacing it with a piecewise linear trajectory composed of two segments. The first segment extends from the origin in the direction of steepest descent until it reaches the minimiser of $m_k$ along that line, defined by

$$p_k^U = -\frac{g_k^T g_k}{g_k^T B_k g_k} g_k.$$

The second segment connects this point $p_k^U$ to the unconstrained minimiser $p_k^B$. Together, these two segments define a composite path $\tilde{p}_k(\tau)$, parametrised by $\tau \in [0, 2]$, as follows:

$$\tilde{p}_k(\tau) = \begin{cases} \tau p_k^U, & 0 \leq \tau \leq 1, \\ p_k^U + (\tau - 1)(p_k^B - p_k^U), & 1 \leq \tau \leq 2. \end{cases}$$

In summary, the Dogleg method determines the step $p_k$ that minimises the model $m_k$ along this piecewise linear path, subject to the trust-region constraint. It can be shown that the minimiser along this trajectory can be computed efficiently, making the Dogleg Method an attractive choice when $B_k$ is positive definite.

#### 3.2.1.2 Two-dimensional subspace minimisation

When the matrix $B_k$ is positive definite, the Dogleg method can be further refined. In this case, the search for the step $p_k$ can be extended to the entire two-dimensional subspace generated by $p_k^U$ and $p_k^B$, that is, by the vectors $g_k$ and $-B_k^{-1} g_k$. Consequently, the original subproblem (3.1) can be reformulated as

$$\min_{p_k} \; m_k(p_k) = f_k + g_k^T p_k + \frac{1}{2} p_k^T B_k p_k \quad \text{s.t.} \quad \|p_k\| \leq \Delta_k, \; p_k \in \operatorname{span}[g_k, B_k^{-1} g_k]. \tag{3.5}$$

It can be verified that the Cauchy point $p_k^c$ is a feasible point for (3.5). Therefore, the reduction in the model function $m_k$ achieved by the optimal solution of this problem is at least as large as that provided by the Cauchy step, ensuring the global convergence properties of the algorithm.

This approach can also be adapted for the case in which $B_k$ is indefinite. Here, we outline the general concept without delving into computational details. When $B_k$ possesses negative eigenvalues, the two-dimensional subspace in (3.5) is modified as follows:

$$\text{span}\,[g_k, (B_k + \alpha I)^{-1} g_k], \quad \text{for some } \alpha \in (-\lambda_1, -2\lambda_1], \tag{3.6}$$

where $\lambda_1$ denotes the most negative eigenvalue of $B_k$. The parameter $\alpha$ is chosen such that $B_k + \alpha I$ becomes positive definite. This range of values for $\alpha$ provides flexibility in selecting computational strategies for constructing the modified subspace. When the condition $\|(B_k + \alpha I)^{-1} g_k\| \leq \Delta_k$ is satisfied, the search within the subspace defined in (3.5) and (3.6) is replaced by defining the step as

$$p_k = -(B_k + \alpha I)^{-1} g_k + v,$$

where $v$ is a vector satisfying $v^T (B_k + \alpha I)^{-1} g_k \leq 0$. This ensures that $\|p_k\| \geq \|(B_k + \alpha I)^{-1} g_k\|$.
Finally, when $B_k$ has zero eigenvalues but no negative ones, the step $p_k$ is simply taken as the Cauchy point, i.e. $p_k = p_k^c$.

### 3.2.2 Constrained optimisation

In mathematical optimisation, *constrained optimisation* refers to the process of optimising an objective function with respect to a set of variables that are subject to one or more constraints. The objective function may represent a *cost* or *energy* function to be minimised, or alternatively, a *reward* or *utility* function to be maximised. Constraints can take different forms. They are often classified as *hard constraints*, which specify strict conditions that the variables must satisfy, or *soft constraints*, where violations of the constraint conditions are permitted but penalised within the objective function.
A general nonlinear constrained optimisation problem can be expressed as follows:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & c_i(x) = 0, \quad \text{for } i \in \mathcal{I} \quad \text{(equality constraints)}, \\ & d_i(x) \geq 0, \quad \text{for } i \in \mathcal{E} \quad \text{(inequality constraints)}, \end{aligned} \tag{3.7}$$

where the objective function $f$, as well as the constraint functions $c_i$ and $d_i$, are assumed to be smooth, real-valued mappings defined on a subset of $\mathbb{R}^n$.
Before introducing specific optimisation methods, we first present some essential definitions.

**Definition 3.1** (Feasible set)**.** We define the *feasible set* $\Gamma$ to be the set of points $x$ that satisfy the constraints; that is,

$$\Gamma = \{x \,|\, c_i(x) = 0,\ i \in \mathcal{I} \text{ and } d_i(x) \geq 0,\ i \in \mathcal{E}\}.$$

**Definition 3.2** (Active set)**.** The *active set* $\Omega(x)$ at any feasible $x$ consists of the equality constraint indices from $\mathcal{I}$ together with the indices of the inequality constraints $i$ for which $d_i(x) = 0$; that is:

$$\Omega(x) = \mathcal{I} \cup \{i \in \mathcal{E} \,|\, d_i(x) = 0\}.$$

At a feasible point $x$, the inequality constraint $i \in \mathcal{E}$ is said to be *active* if $d_i(x) = 0$ and *inactive* if the strict inequality $d_i(x) > 0$ is satisfied.

In other words, an *active constraint* is one that directly *binds* or *restricts* the solution at a given point, whereas an *inactive constraint* has no influence on the solution at that point. In constrained optimisation, identifying which constraints are active is essential, as this determines the feasible directions in which the objective function can be improved. Active-set methods are built on this

principle: they begin with an initial estimate of the active constraint set and solve a simplified optimisation problem that only involves these constraints. If this estimate is correct, the resulting point is optimal; otherwise, the active set is updated, and the procedure is repeated until convergence.

**Definition 3.3** (LICQ)**.** Given the point $x$ and the active set $\Omega(x)$, we say that the *linear independence constraint qualification* (LICQ) holds if the set of active constraint gradients $\{\nabla d_i(x), i \in \Omega(x)\}$ is linearly independent.

**Definition 3.4** (Lagrangian function)**.** We define the *Lagrangian function* for the general problem (3.7) as

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{I}} \lambda_i c_i(x) - \sum_{i \in \mathcal{E}} \lambda_i d_i(x). \tag{3.8}$$

**Theorem 3.5.** *Suppose that $x^\star$ is a local solution of* (3.7)*, that the functions $f, c_i$ and $d_i$ in* (3.7) *are continuously differentiable, and that the LICQ holds at $x^\star$. Then there is a Lagrange multiplier vector $\lambda^\star$, with components $\lambda_i^\star$, $i \in \mathcal{I} \cup \mathcal{E}$, such that the following conditions are satisfied at*

$$\begin{aligned} \nabla_x \mathcal{L}(x^\star, \lambda^\star) &= 0, \\ c_i(x^\star) &= 0, \quad i \in \mathcal{I} \\ d_i(x^\star) &\geq 0, \quad i \in \mathcal{E} \\ \lambda_i^\star &\geq 0, \quad i \in \mathcal{E} \end{aligned} \tag{3.9}$$

$$\lambda_i^\star d_i(x^\star) \geq 0, \quad i \in \mathcal{E} \cup \mathcal{I}. \tag{3.10}$$

*The conditions* (3.9) *are often known as the Karush–Kuhn–Tucker conditions, or KKT conditions for short. The conditions* (3.10) *are complementarity conditions; they imply that either constraint $i$ is active or $\lambda_i^\star = 0$, or possibly both.*

#### 3.2.2.1 Penalty methods

Penalty methods reformulate constrained optimisation problems as unconstrained ones by introducing additional terms, known as *penalty terms*, into the objective function. These terms impose a cost for violating the constraints, effectively discouraging infeasible solutions. Referring to problem (3.7), we define the general penalty function as

$$Q(x, \mu) = f(x) + \mu \sum_{i \in \mathcal{I}} \psi(c_i(x)) + \mu \sum_{i \in \mathcal{E}} \phi(d_i(x)),$$

where $\mu > 0$ denotes the penalty parameter, and $\psi$ and $\phi$ are functions that penalise violations of equality and inequality constraints, respectively. Different penalty methods arise from distinct choices of the functions $\psi$ and $\phi$. Typically, these functions are selected to be smooth, and the corresponding penalty problem is solved for a sequence of increasing values of $\mu$, until an adequate approximation to the constrained optimum is obtained.
Alternatively, one may employ *exact penalty functions*, which trade smoothness for the property that, for a suitable choice of the penalty parameter, the minimiser of the penalised problem coincides exactly with the solution of the original constrained problem (see, for instance, [75, 76]). This property is particularly advantageous, as it reduces the dependence of the method on the specific updating strategy for $\mu$.
Penalty-based approaches can be viewed from several perspectives. In what follows, we summarise three main categories:

1. *Quadratic penalty methods:* These methods augment the objective function with a quadratic term representing the squared magnitude of the constraint violations. The resulting penalty function is smooth and differentiable, facilitating both analysis and computation. However, enforcing feasibility often requires very large values of the penalty parameter, which may lead to numerical ill-conditioning.

2. *Nonsmooth exact penalty methods:* In this approach, the constrained optimisation problem is replaced by a single unconstrained one, rather than a sequence, by employing penalty functions that yield *exact* solutions, i.e., the solutions to the penalised problem coincide with those of the original constrained formulation. Although these methods avoid large penalty parameters, they introduce nonsmoothness, which can complicate the optimisation process. A well-known example of such a function is the $\ell_1$-penalty. With the rise of large-scale problems in areas such as machine learning, nonsmooth optimisation has become increasingly relevant, and numerous efficient algorithms for these problems have been developed (see, e.g., [77]).

3. *Method of multipliers (augmented Lagrangian method):* This method combines the Lagrangian function with additional quadratic penalty terms, forming what is known as the *augmented Lagrangian*. It can be interpreted as an exact penalty method that alleviates the ill-conditioning commonly associated with purely quadratic penalties by explicitly incorporating estimates of the Lagrange multipliers. The method typically converges rapidly and can handle both equality and inequality constraints, although each iteration requires solving an unconstrained subproblem.

In addition, one can consider *interior penalty functions*, which add a term to the objective function that penalises approaches to the boundary of the feasible region. The idea is to keep the iterates strictly within the feasible set by introducing a barrier term that tends to infinity as the solution approaches the boundary. A classic example of such a function is the *logarithmic barrier function*, where the penalty depends on the logarithm of the constraint functions.
Interior penalty functions are particularly effective in maintaining feasibility, which is desirable in applications such as linear and quadratic programming. However, these methods also have limitations: they require an initial point that lies strictly within the feasible region, and the steep growth of the penalty term near the boundary can make the optimisation landscape very sharp, increasing the difficulty of finding the optimal solution.

**The quadratic penalty method.** The quadratic penalty method reformulates a constrained optimisation problem as an unconstrained one by introducing a *penalty function* that augments the original objective function with additional terms corresponding to the constraint violations. Each penalty term is nonzero only when a constraint is violated and increases quadratically with the degree of violation, which gives rise to the term *quadratic penalty*.
A classical strategy consists of defining a sequence of such penalty functions, where each penalty term is multiplied by a positive coefficient that controls the severity of constraint enforcement. As this coefficient grows, violations of the constraints are penalised more heavily, driving the minimiser of the penalised problem closer to the feasible region of the original constrained problem.
The simplest quadratic penalty formulation arises when the penalties are proportional to the squares of the constraint violations. Let us first consider an equality-constrained optimisation problem of the form

$$\min_x f(x) \quad \text{subject to} \quad c_i(x) = 0, \quad i \in \mathcal{I}. \tag{3.11}$$

The corresponding quadratic penalty function $Q(x;\mu)$ is defined as

$$Q(x;\mu) \coloneqq f(x) + \frac{\mu}{2} \sum_{i \in \mathcal{I}} c_i^2(x), \tag{3.12}$$

where $\mu > 0$ is the *penalty parameter*. As previously discussed, larger values of $\mu$ (i.e., as $\mu \to \infty$) impose stronger penalties for constraint violations, thereby pushing the minimiser of $Q(x;\mu)$ closer to the feasible set. It is therefore natural to consider a sequence $\{\mu_k\}$ with $\mu_k \to \infty$ as $k \to \infty$, and to compute an approximate minimiser $x_k$ of $Q(x;\mu_k)$ for each $\mu_k$. Since the penalty function in (3.12) is smooth, standard unconstrained optimisation methods can be employed to determine each $x_k$.

For the general constrained optimisation problem (3.7), the quadratic penalty function extends naturally to include both equality and inequality constraints:

$$Q(x;\mu) \coloneqq f(x) + \frac{\mu}{2}\sum_{i\in\mathcal{I}} c_i^2(x) + \frac{\mu}{2}\sum_{i\in\mathcal{E}}\big([d_i(x)]^-\big)^2, \tag{3.13}$$

where $[y]^- = \max\{-y, 0\}$ denotes the negative part of $y$.

*Remark.* It is worth noting that, due to the use of the operator $[\,\cdot\,]^-$, the function $Q(x;\mu)$ may possess a lower degree of smoothness than the original objective and constraint functions.

A generic framework for algorithms based on the quadratic penalty function can be expressed as in Algorithm 2.

**Algorithm 2** Quadratic penalty method

1: **Input:** Penalty parameter $\mu_0 > 0$, tolerance sequence $\{\tau_k\}$ with $\tau_k \to 0$, starting point $x_0^s$.
2: **Initialise:** $k \leftarrow 0$
3: **Output:** Approximate solution $x_k$, final penalty parameter $\mu_k$.
4: **while** final convergence test is not satisfied **do**
5: Compute an approximate minimiser $x_k$ of $Q(x;\mu_k)$, starting from $x_k^s$.
6: Stop inner minimisation when $\|\nabla_x Q(x_k;\mu_k)\| \le \tau_k$.
7: **if** convergence criteria for the overall method are satisfied **then**
8: **break**
9: **end if**
10: Choose new penalty parameter $\mu_{k+1} > \mu_k$.
11: Select new starting point $x_{k+1}^s$.
12: $k \leftarrow k + 1$
13: **end while**
14: **return** Approximate solution $x_k$.

One can choose the parameter sequence $\{\mu_k\}$ accordingly to the difficulty of minimising the penalty function at each iteration. For instance, the more expensive the minimisation of $Q(x;\mu_k)$ tends to be, the less larger $\mu_{k+1}$ is in relation to $\mu_k$.

*Remark.* Unfortunately, it could happen that the iterates may move away from the feasible region when the penalty parameter is not large enough; therefore, the stop rule $\|\nabla_x Q(x;\mu_k)\| \le \tau_k$ may not be satisfied. In order to avoid this, one has to include safeguards that increase the penalty parameter when either the decreasing speed of the constraint violation is not high enough or when the iterates appear to be diverging.

*Remark.* Although $Q(x;\mu_k)$ is smooth when only equality constraints are considered, it is important to note that as $\mu_k$ becomes large, the minimisation of $Q(x;\mu_k)$ becomes challenging. As a matter of fact, the Hessian $\nabla^2_{xx} Q(x;\mu_k)$ becomes arbitrary ill conditioned near the minimiser. As a result, even if we could employ algorithms for unconstrained minimisation (such as Newton's, quasi-Newton's methods, or conjugate gradient), their performance might be suboptimal. To mitigate this, a judicious choice of the starting point $x_{k+1}^s$ can be made or one could set $x_{k+1}^s = x_k$ and choose $\mu_{k+1}$ to be slightly larger that $\mu_k$.

For more details on convergence theorems and on how to reformulate the problem when there is ill conditioning for the Hessian see [78, 79].

**Nonsmooth penalty functions.** A widely used example of a nonsmooth penalty function in nonlinear programming is the $\ell_1$ *penalty function*, defined as

$$Q_1(x;\mu) = f(x) + \mu\sum_{i\in\mathcal{I}} |c_i(x)| + \mu\sum_{i\in\mathcal{E}} [d_i(x)]^-, \tag{3.14}$$

where $[y]^- = \max\{-y, 0\}$ and $\mu > 0$ is the penalty parameter. It can be shown that the $\ell_1$ penalty function is an *exact* penalty function, meaning that for an appropriate value of $\mu$, the minimiser of $Q_1(x;\mu)$ coincides with the solution of the original constrained problem.
More generally, exact nonsmooth penalty functions can be formulated using other vector norms. A common generalisation takes the form

$$Q(x;\mu) = f(x) + \mu\|d_{\mathcal{E}}(x)\| + \mu\|[c_{\mathcal{I}}(x)]^-\|,$$

where $\|\cdot\|$ denotes an arbitrary vector norm, and the vector functions $d_{\mathcal{E}}(x)$ and $c_{\mathcal{I}}(x)$ correspond to the sets of equality and inequality constraints, respectively. In practice, the most commonly employed norms are the $\ell_1$, $\ell_\infty$, and the (nonsquared) $\ell_2$ norms.
Although $Q_1$ is not differentiable, it admits directional derivatives. Specifically, the directional derivative of $Q_1(x;\mu)$ along a direction $p$ is denoted by $D(Q_1(x;\mu);p)$. A point $\hat{x} \in \mathbb{R}^n$ is said to be a *stationary point* of the penalty function $Q_1(x;\mu)$ if it satisfies

$$D(Q_1(\hat{x};\mu);p) \geq 0, \quad \forall\, p \in \mathbb{R}^n.$$

Characterising such stationary points is important when analysing the convergence behaviour of nonsmooth penalty methods.
To quantify constraint violations, we define the *measure of infeasibility* as

$$h(x) = \sum_{i\in\mathcal{I}} |c_i(x)| + \sum_{i\in\mathcal{E}} [d_i(x)]^-. \tag{3.15}$$

The basic structure of the classical $\ell_1$ penalty method can be summarised in Algorithm 3.

**Algorithm 3** Classic $\ell_1$ penalty method

1: **Input:** Initial penalty parameter $\mu_0 > 0$, tolerance $\tau > 0$, starting point $x_0^s$.
2: **Output:** Approximate minimiser $x_k$ of the penalised problem $Q_1(x;\mu)$.
3: **for** $k = 0, 1, 2, \ldots$ **do**
4: Find an approximate minimiser $x_k$ of $Q_1(x;\mu_k)$ starting from $x_k^s$.
5: **if** $h(x_k) \leq \tau$ **then**
6: **Stop** and return $x_k$ as approximate solution.
7: **end if**
8: Choose new penalty parameter $\mu_{k+1} > \mu_k$.
9: Choose new starting point $x_{k+1}^s$.
10: **end for**
11: **return** Current iterate $x_k$ as approximate solution.

The minimisation of $Q_1(x;\mu_k)$ is made difficult by the nonsmoothness of the function.

**Augmented Lagrangian method.** In this paragraph, we describe the approach known as the *method of multipliers* or the *augmented Lagrangian method*. Unlike classical penalty methods, this technique incorporates explicit estimates of the Lagrange multipliers into the objective function, thereby mitigating the ill-conditioning often associated with large penalty parameters. Moreover, the augmented Lagrangian formulation retains much of the smoothness of the original problem, allowing the use of standard unconstrained optimisation techniques for its solution.
Consider the equality-constrained optimisation problem (3.11). The quadratic penalty function introduced in (3.12) penalises constraint violations through squared infeasibilities, each scaled by the factor $\mu/2$. However, the approximate minimisers $x_k$ obtained from the quadratic penalty problem do not exactly satisfy the feasibility conditions $c_i(x) = 0$ for all $i \in \mathcal{I}$; instead, the residuals $c_i(x_k)$ only approach zero as $\mu_k \to \infty$. This observation motivates the development of a modified formulation that reduces such systematic infeasibility.

The *augmented Lagrangian function* $\mathcal{L}_A(x, \lambda; \mu)$ achieves this by introducing explicit estimates of the Lagrange multipliers $\lambda$, based on the infeasibility observed in previous iterations.

**Definition 3.6** (Augmented Lagrangian function)**.** The *augmented Lagrangian function* is defined as

$$\mathcal{L}_A(x, \lambda; \mu) \coloneqq f(x) - \sum_{i \in \mathcal{I}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{I}} c_i^2(x), \tag{3.16}$$

where $\mu > 0$ is the penalty parameter.

The function $\mathcal{L}_A(x, \lambda; \mu)$ differs from the standard Lagrangian (3.8) by the presence of the quadratic terms in $c_i(x)$, and from the quadratic penalty function (3.12) by the inclusion of the multiplier terms involving $\lambda$. Hence, it can be regarded as a hybrid between the Lagrangian and the quadratic penalty formulations.
To derive an iterative algorithm, we fix the penalty parameter $\mu = \mu_k > 0$ at iteration $k$, set the multiplier estimate to $\lambda = \lambda^k$, and minimise $\mathcal{L}_A(x, \lambda^k; \mu_k)$ with respect to $x$. The first-order optimality condition for this unconstrained minimisation gives

$$0 \approx \nabla_x \mathcal{L}_A(x_k, \lambda^k; \mu_k) = \nabla f(x_k) - \sum_{i \in \mathcal{I}} [\lambda_i^k - \mu_k c_i(x_k)] \nabla c_i(x_k).$$

Comparing this expression with the optimality conditions of the original constrained problem, we obtain the following approximation for the optimal Lagrange multipliers:

$$\lambda_i^\star \approx \lambda_i^k - \mu_k c_i(x_k), \quad \forall i \in \mathcal{I},$$

which can equivalently be written as

$$c_i(x_k) \approx -\frac{1}{\mu_k}(\lambda_i^\star - \lambda_i^k), \quad \forall i \in \mathcal{I}.$$

Thus, as $\lambda^k \to \lambda^\star$, the infeasibility in $x_k$ becomes much smaller than $1/\mu_k$, in contrast with the quadratic penalty method, where infeasibility is typically proportional to $1/\mu_k$. This leads naturally to an update rule for the Lagrange multiplier estimates, defined as

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(x_k), \quad \forall i \in \mathcal{I}. \tag{3.17}$$

The augmented Lagrangian method for equality-constrained problems can be summarised in Algorithm 4.

**Algorithm 4** Augmented Lagrangian method — equality constraints

**Input:** Penalty parameter $\mu_0 > 0$, tolerance $\tau_0 > 0$, initial guess $x_0^s$, initial multipliers $\lambda^0$.
**Output:** Approximate minimiser $x_k$ of the augmented Lagrangian $\mathcal{L}_A$.
**for** $k = 0, 1, 2, \ldots$ **do**
    Compute an approximate minimiser $x_k$ of $\mathcal{L}_A(x, \lambda^k; \mu_k)$ starting from $x_k^s$.
    Terminate inner loop when $\|\nabla_x \mathcal{L}_A(x_k, \lambda^k; \mu_k)\| \le \tau_k$.
    **if** convergence criteria for problem (3.11) are satisfied **then**
        **Stop** and return $x_k$ as approximate solution.
    **end if**
    Update multipliers: $\lambda^{k+1}$ using (3.17).
    Update penalty parameter: choose $\mu_{k+1} \ge \mu_k$.
    Set new starting point: $x_{k+1}^s \leftarrow x_k$.
    Choose new tolerance $\tau_{k+1}$.
**end for**
**return** Current iterate $x_k$ as approximate solution.

*Remark.* It can be shown that the augmented Lagrangian method converges without the need to increase the penalty parameter $\mu$ indefinitely. As a result, issues related to ill-conditioning are significantly reduced, and the choice of the initial point $x_{k+1}^s$ becomes less critical for the convergence of the algorithm.

Two important theoretical properties of the augmented Lagrangian can be established:

1. If the exact Lagrange multiplier vector $\lambda^\star$ is known, the solution $x^\star$ of the equality-constrained problem (3.11) is a strict local minimiser of $\mathcal{L}_A(x, \lambda^\star; \mu)$ for all sufficiently large values of $\mu$. In practice, even when $\lambda^\star$ is not known exactly, a good approximation of $x^\star$ can still be obtained by minimising $\mathcal{L}_A(x, \lambda; \mu)$, provided that the current estimate $\lambda$ is reasonably close to $\lambda^\star$, and that $\mu$ is of moderate size.

2. When $\lambda \neq \lambda^\star$, under suitable regularity assumptions, it is possible to prove the existence of a minimiser of $\mathcal{L}_A(x, \lambda; \mu)$ that lies in a neighbourhood of $x^\star$. Furthermore, one can derive error bounds for both the primal variable $x_k$ and the updated Lagrange multiplier estimate $\lambda^{k+1}$ obtained at iteration $k$.

From these results, it follows that the augmented Lagrangian method provides two distinct mechanisms for improving the accuracy of $x_k$: either by refining the multiplier estimate $\lambda_k$ or by increasing the penalty parameter $\mu_k$. In contrast, the quadratic penalty method relies solely on enlarging $\mu_k$ to achieve the same effect.
In the literature, the quadratic penalty approach is often applied when the number of constraints is relatively small, and it serves as a useful regularisation component within other optimisation schemes, such as sequential quadratic programming (SQP). However, the augmented Lagrangian method is generally preferred in practice due to its simplicity and robustness. The associated subproblems are typically no more difficult to solve than in the quadratic penalty case, and the inclusion of multiplier estimates reduces the need for excessively large penalty parameters, thereby avoiding ill-conditioning and improving numerical stability.
In recent developments, augmented Lagrangian ideas have also been successfully extended to non-smooth settings, such as those involving the $\ell_1$ penalty function. This extension has proven particularly effective for solving challenging problems like mathematical programs with complementarity constraints, which often violate standard constraint qualifications. In such cases, the problematic constraints can be incorporated as penalty terms, broadening the applicability and effectiveness of augmented Lagrangian-based methods.

#### 3.2.2.2 Nonlinear constrained optimisation

In this Section, we focus on algorithms designed to compute local solutions of problem (3.7), where both the objective function and the constraint functions are nonlinear. The principal challenge in this context lies in determining which of the inequality constraints are *active* at the solution and which are *inactive*. Active-set methods address this difficulty by iteratively estimating the subset of constraints that are active at the optimal point. Let $\Omega^\star$ denote the true active set, i.e., the collection of constraints that hold as equalities at the local solution. Since $\Omega^\star$ is unknown, the algorithm begins with an initial estimate, referred to as the *working set* $W$. At each iteration, a simplified optimisation problem is solved in which the constraints belonging to $W$ are treated as equalities, while all remaining inequality constraints are temporarily disregarded. After computing a candidate solution $x^\star$ for this subproblem, the algorithm checks whether there exists a set of Lagrange multipliers such that the pair $(x^\star, \lambda^\star)$ satisfies the Karush–Kuhn–Tucker (KKT) conditions (3.9). If this is the case, $x^\star$ is accepted as a local solution to (3.7); otherwise, the working set $W$ is modified, and the process is repeated. It is worth noting that, in principle, there are $2^{|\mathcal{E}|}$ possible choices for the working set, since for each inequality constraint $i \in \mathcal{E}$, one must decide whether it belongs to $W$ or not. Consequently, the number of potential working sets grows exponentially with the number of inequality constraints, a phenomenon commonly referred to as the *combinatorial difficulty* of

nonlinear programming. For this reason, constructing an algorithm that systematically explores all possible working sets is computationally infeasible. In practice, however, various heuristics and problem-specific insights can be used to guide the selection of $W$, thereby reducing the number of working sets that need to be tested. Such heuristics typically exploit structural properties of the objective and constraint functions to eliminate unpromising candidates and accelerate convergence. A practical illustration of this approach can be found in [80].

**Reducing the number of variables.** To simplify an optimisation problem and decrease the number of degrees of freedom, it is often advantageous to exploit the constraints in order to eliminate some of the variables. However, care must be taken when applying such *elimination techniques*, as they may unintentionally alter the problem's structure or introduce numerical ill-conditioning. In particular, when nonlinear constraints are used to eliminate variables, the resulting transformations can generate hidden errors that are difficult to track. For this reason, most optimisation algorithms avoid nonlinear elimination and instead rely on *linearised constraints* to simplify the problem while maintaining numerical stability.
We now present a procedure for variable elimination using linear constraints. Consider the optimisation problem:

$$\min f(x) \quad \text{subject to} \quad Ax = b, \tag{3.18}$$

where $A \in \mathbb{R}^{m\times n}$ with $m \leq n$ and full row rank. Hence, there exists a subset of $m$ columns of $A$ that are linearly independent. Let $B \in \mathbb{R}^{m\times m}$ denote the submatrix formed by these independent columns, and define a permutation matrix $P \in \mathbb{R}^{n\times n}$ that reorders the columns of $A$ so that the selected independent columns appear first. It follows that

$$AP = [B \,|\, N],$$

where $N \in \mathbb{R}^{m\times(n-m)}$ contains the remaining columns of $A$.
We partition the variable vector $x \in \mathbb{R}^n$ accordingly:

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = P^T x,$$

where $x_B \in \mathbb{R}^m$ and $x_N \in \mathbb{R}^{n-m}$ are called the *basic* and *non-basic variables*, respectively. Since $PP^T = I$, substituting into the constraint $Ax = b$ yields

$$b = Ax = AP(P^T x) = Bx_B + Nx_N.$$

Solving for the basic variables gives

$$x_B = B^{-1}b - B^{-1}Nx_N.$$

Thus, for any choice of $x_N$, a feasible vector $x$ satisfying $Ax = b$ can be obtained by computing $x_B$ from the above relation. Substituting this expression into the objective function leads to an equivalent unconstrained problem:

$$\min_{x_N} f\left(P\begin{bmatrix} B^{-1}b - B^{-1}Nx_N \\ x_N \end{bmatrix}\right).$$

This procedure effectively eliminates the equality constraints in (3.18), reducing the optimisation task to a problem involving only the non-basic variables.
This approach is commonly referred to as the *simple elimination of variables*. The simple elimination of variables can be interpreted in a particularly insightful way. Assume, without loss of generality, that $P = I$, so that the basic columns of $A$ occupy the first $m$ positions. From the relations derived earlier, any feasible point $x$ satisfying the linear constraint $Ax = b$ can be written

as

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = Yb + Zx_N,$$

where the matrices $Y$ and $Z$ are defined as

$$Y = \begin{bmatrix} B^{-1} \\ 0_{(n-m)\times m} \end{bmatrix} \in \mathbb{R}^{n\times m}, \qquad Z = \begin{bmatrix} -B^{-1}N \\ I_{(n-m)\times(n-m)} \end{bmatrix} \in \mathbb{R}^{n\times(n-m)}.$$

The matrix $Z$ has $n-m$ linearly independent columns and satisfies the relation $AZ = 0$; hence, $Z$ forms a *basis for the null space* of $A$.

This representation shows that any feasible vector $x$ can be expressed as the sum of a particular solution of $Ax = b$, namely $Yb$, and a linear combination of the columns of $Z$, which span the null space of the constraints. In other words, the feasible set consists of all displacements from the particular solution $Yb$ along directions that leave the constraints unchanged. The particular solution $Yb$ can be interpreted as the point obtained by fixing $n-m$ components of $x$ to zero and adjusting the remaining $m$ variables until the constraints are satisfied.

*Remark.* Although simple elimination is computationally inexpensive, it may introduce numerical instabilities. The choice of the basis matrix $B$ plays a crucial role: as a matter of fact, an unsuitable basis can lead to excessive coordinate relaxation or large numerical errors. To mitigate these issues, one can define the particular solution $Yb$ as the *minimum-norm step* that satisfies the constraints.

*Generalising the simple elimination of variables.* Let us now consider matrices $Y \in \mathbb{R}^{n\times m}$ and $Z \in \mathbb{R}^{n\times(n-m)}$ satisfying

$$[Y \mid Z] \in \mathbb{R}^{n\times n} \text{ is nonsingular}, \qquad AZ = 0.$$

Then, any vector $x$ satisfying $Ax = b$ can be expressed as

$$x = Yx_Y + Zx_Z,$$

for some $x_Y \in \mathbb{R}^m$ and $x_Z \in \mathbb{R}^{n-m}$. Substituting into the constraint gives

$$Ax = (AY)x_Y = b.$$

Since $AY$ is nonsingular, one can solve explicitly for $x_Y$:

$$x_Y = (AY)^{-1}b.$$

Hence, any feasible point $x$ can be written as

$$x = Y(AY)^{-1}b + Zx_Z,$$

where $x_Z \in \mathbb{R}^{n-m}$ is free. This leads to an equivalent unconstrained optimisation problem:

$$\min_{x_Z} f\big(Y(AY)^{-1}b + Zx_Z\big).$$

From this formulation, it is evident that choosing $Y$ such that $AY$ is well-conditioned is desirable to improve numerical stability.

A practical way to achieve this is to use the *QR factorisation*[*] of $A^T$, given by

$$A^T\Pi = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix},$$

[*]A QR decomposition factorises a matrix $A$ into the product $A = QR$, where $Q$ is orthogonal and $R$ is upper triangular.

where $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ is orthogonal, $Q_1 \in \mathbb{R}^{n\times m}$ and $Q_2 \in \mathbb{R}^{n\times(n-m)}$ have orthonormal columns, $R \in \mathbb{R}^{m\times m}$ is upper triangular and nonsingular, and $\Pi \in \mathbb{R}^{m\times m}$ is a permutation matrix. Setting

$$Y = Q_1, \qquad Z = Q_2,$$

we obtain an orthonormal basis for $\mathbb{R}^n$ satisfying

$$AY = \Pi R^T, \qquad AZ = 0.$$

Thus, any feasible vector $x$ satisfying $Ax = b$ can be expressed as

$$x = Q_1 R^{-T} \Pi^T b + Q_2 x_Z,$$

for some $x_Z \in \mathbb{R}^{n-m}$. In this representation, the computation of $R^{-T}\Pi^T b$ is efficient, requiring only a single triangular substitution.

*Remark.* A straightforward computation shows that the particular solution $Q_1 R^{-T}\Pi^T b$ can also be written as $A^T(AA^T)^{-1}b$. This vector corresponds to the solution of the following optimisation problem:

$$\min \|x\|^2 \quad \text{subject to} \quad Ax = b,$$

which is known as the *minimum-norm solution* of the system $Ax = b$.

Since the dominant computational cost of this orthogonal-basis approach lies in performing the QR factorisation, it offers excellent numerical stability. However, for large and sparse matrices $A$, computing the QR decomposition can be significantly more expensive than the simple elimination strategy. In practice, hybrid elimination methods have been developed to balance the numerical robustness of orthogonal factorisations with the efficiency of simpler reduction schemes.

#### 3.2.2.3 Merit functions and filters

During the execution of an optimisation algorithm, it may occur that a proposed step decreases the objective function while simultaneously increasing constraint violations. The key question is whether such a step should be accepted. To balance these conflicting objectives, one may employ *merit functions* and *filters*. Generally, in constrained optimisation, a step $p$ is accepted if it leads to a sufficient reduction in a chosen merit function $Q$, or alternatively, if it satisfies the criteria of a filter.

The utility of merit functions becomes particularly significant in nonlinear or complex problems where analytical methods are impractical and numerical procedures must be used. In such contexts, merit functions and filters help steer the algorithm toward a solution that both minimises the objective function and respects the problem constraints.

**Merit functions.** A merit function is a scalar quantity that evaluates the overall quality of a candidate solution by combining the objective value with a measure of constraint violation. The aim is to minimise this function; doing so ideally yields a solution that also satisfies the original constrained problem. In unconstrained optimisation, the objective function itself naturally serves as a merit function. This remains valid in constrained settings if all iterates respect the constraints (see, for instance, [81]). However, when intermediate iterates are permitted to violate constraints, the merit function must be extended to penalise such violations. A common approach is to use an $\ell_1$-penalty formulation, as defined in (3.14), where the parameter $\mu$, known as the *penalty parameter*, balances constraint satisfaction against objective minimisation. The $\ell_1$-based merit function, denoted $Q_1$, is known to be exact but nondifferentiable.

**Definition** (Exact merit function)**.** A merit function $Q(x;\mu)$ is said to be *exact* if there exists a scalar $\mu^\star > 0$ such that for all $\mu > \mu^\star$, any local minimiser of the nonlinear programming problem (3.7) is also a local minimiser of $Q(x;\mu)$.

It can be shown that, under certain regularity conditions, the $\ell_1$ merit function $Q_1(x;\mu)$ is exact, with the corresponding threshold value

$$\mu^\star = \max\{|\lambda_i^\star| : i \in \mathcal{I} \cup \mathcal{E}\},$$

where $\lambda_i^\star$ denote the Lagrange multipliers associated with an optimal solution $x^\star$.
For equality-constrained problems, where $c(x)$ represents the vector of equality constraints, one can employ the (exact but nonsmooth) $\ell_2$-based merit function:

$$Q_2(x;\mu) = f(x) + \mu\|c(x)\|_2.$$

Note that $Q_2$ is nondifferentiable at points $x$ where $c(x) = 0$, since its gradient is undefined at those locations.
To achieve both smoothness and exactness, additional terms may be incorporated into the merit function. A notable example for equality-constrained problems is the *Fletcher augmented Lagrangian*, given by

$$Q_F(x;\mu) = f(x) - \lambda(x)^T c(x) + \frac{1}{2}\mu \sum_{i\in\mathcal{I}} c_i(x)^2,$$

where $\mu > 0$ is the penalty parameter and

$$\lambda(x) = [A(x)A(x)^T]^{-1} A(x)\nabla f(x),$$

with $A(x)$ denoting the Jacobian of $c(x)$.

*Remark.* Although this formulation possesses appealing theoretical properties, it is seldom applied in practice due to the computational effort involved in determining $\lambda(x)$.

Another well-known example is the *standard augmented Lagrangian*, defined for equality-constrained problems as

$$\mathcal{L}_A(x,\lambda;\mu) = f(x) - \lambda^T c(x) + \frac{1}{2}\mu\|c(x)\|_2^2.$$

*Remark.* Strictly speaking, the standard augmented Lagrangian is not a merit function, since the pair $(x^\star,\lambda^\star)$ corresponding to a solution of the nonlinear problem typically yields only a stationary point rather than a minimiser. Nonetheless, by appropriately updating $\mu$ and $\lambda$ within sequential quadratic programming methods, $\mathcal{L}_A$ can effectively serve as a merit function.

When merit functions are nondifferentiable, one may instead rely on their *directional derivatives*, denoted $\mathrm{D}(Q(x;\mu);p)$, whenever they exist. This is often the case for $\ell_1$ and $\ell_2$-type merit functions. For instance, in line search methods, the step length $\alpha > 0$ must be chosen small enough to satisfy

$$Q(x+\alpha p;\mu) \le Q(x;\mu) + \eta\,\alpha\,\mathrm{D}(Q(x;\mu);p),$$

for some $\eta \in (0,1)$. In trust-region methods, one typically approximates $Q$ by a quadratic model $q(p)$, leading to the sufficient decrease condition

$$Q(x+p;\mu) \le Q(x;\mu) - \eta\,(q(0) - q(p)),$$

again for some $\eta \in (0,1)$.

**Filters.** Filters represent an alternative approach to merit functions in handling constrained optimisation problems. Unlike a merit function, which merges the objective value and the measure of constraint violation into a single scalar, a filter keeps track of pairs consisting of objective function values and infeasibility measures that are deemed acceptable.

Consider the measure of infeasibility defined analogously to (3.15), but with $\mu = 1$:

$$h(x) = \sum_{i \in \mathcal{I}} |c_i(x)| + \sum_{i \in \mathcal{E}} [d_i(x)]^-.$$

The optimisation goal is therefore to reduce both $f(x)$ and $h(x)$ simultaneously. While merit functions achieve this by merging the two objectives into one, filter methods keep them distinct. In essence, a filter-based algorithm accepts a trial point $x^+$ if its corresponding pair $(f(x^+), h(x^+))$ is not *dominated* by any previously accepted pair $(f_l, h_l) = (f(x_l), h(x_l))$ already in the filter.

**Definition.** a. A pair $(f_k, h_k)$ *dominates* another pair $(f_l, h_l)$ if $f_k \leq f_l$ and $h_k \leq h_l$.

b. A *filter* is a collection of pairs $(f_l, h_l)$ such that no pair dominates another.

c. An iterate $x_k$ is said to be *acceptable* to the filter if its associated pair $(f_k, h_k)$ is not dominated by any pair already in the filter.

When a new iterate $x_k$ is deemed acceptable, its pair $(f_k, h_k)$ is added to the filter, and any existing pairs that it dominates are removed. In *line search methods* (see Section 3.2.1), if a trial step $x^+ = x_k + \alpha_k p_k$ produces a pair $(f^+, h^+)$ that satisfies the filter acceptance criteria, it becomes the next iterate ($x_{k+1} = x^+$). Otherwise, a backtracking procedure is employed to find a suitable step length. In *trust-region methods* (see Section 3.2.1), if a trial step is rejected by the filter, the trust-region radius is reduced, and a new step is computed. To ensure both global convergence and practical efficiency, certain refinements are necessary. In particular, new iterates should not be accepted if their $(f, h)$ pairs are too close to existing ones in the filter. To enforce this, an additional acceptability condition is introduced. A trial point $x^+$ is accepted if, for all $(f_j, h_j)$ in the filter,

$$f(x^+) \leq f_j - \beta h_j \quad \text{or} \quad h(x^+) \leq h_j - \beta h_j,$$

where $\beta \in (0, 1)$. In practice, $\beta = 10^{-5}$ often yields satisfactory results. However, for theoretical analysis, the first inequality is typically replaced with

$$f(x^+) \leq f_j - \beta h^+.$$

Despite their advantages, filter methods may encounter certain difficulties. For example, in line search algorithms, the accepted step lengths $\alpha_k$ might become exceedingly small to satisfy the filter conditions, leading to stagnation. When $\alpha_k$ falls below a minimum threshold $\alpha_{\min}$, the algorithm transitions into a *feasibility restoration phase*. Similarly, in trust-region frameworks, repeated rejection of trial steps may cause the trust-region radius to shrink excessively, rendering the subproblem infeasible. In both situations, the algorithm switches to a feasibility restoration phase, whose objective is to reduce constraint violation by approximately solving

$$\min_x h(x).$$

Although $h(x)$ is generally nonsmooth, several established techniques allow one to minimise it effectively by solving a sequence of smooth constrained optimisation subproblems.
Algorithm 5 presents a representative example of a filter-based algorithm in which the iterates are generated using a trust-region strategy.

### 3.2.3 Metaheuristics

It is worth emphasising that certain optimisation problems can be so complex that obtaining an exact optimal solution is infeasible in practice. As a result, alternative methodologies have been developed to provide structured strategies and guiding principles for finding satisfactory solutions. These are known as *metaheuristic methods*; see, for instance, [82–84].

**Algorithm 5** General filter method

**Input:** Starting point $x_0^s$, initial trust-region radius $\Delta_0$.
**Initialise:** $k \leftarrow 0$.
**Output:** Approximation of the solution $x_k$.
**while** convergence test not satisfied **do**
    **if** step-generation subproblem is infeasible **then**
        Compute $x_{k+1}$ using the feasibility restoration phase.
    **else**
        Compute trial iterate $x^+ = x_k + p_k$.
        **if** $(f^+, h^+)$ is acceptable to the filter **then**
            Set $x_{k+1} \leftarrow x^+$.
            Add $(f_{k+1}, h_{k+1})$ to the filter.
            Choose $\Delta_{k+1} \geq \Delta_k$.
            Remove all pairs from the filter dominated by $(f_{k+1}, h_{k+1})$.
        **else**
            Reject the step, set $x_{k+1} \leftarrow x_k$.
            Choose $\Delta_{k+1} < \Delta_k$.
        **end if**
    **end if**
    $k \leftarrow k + 1$.
**end while**
**return** Current iterate $x_k$ as approximate solution.

**Definition 3.7** (Heuristic method)**.** A *heuristic method* refers to a problem-solving procedure designed to identify a feasible solution—one that satisfies all the problem constraints—even though it may not be optimal. A well-crafted heuristic typically yields a solution that is close to optimal; if it fails to do so, one may reasonably infer that no such near-optimal solution exists.

Heuristic methods are often inspired by intuitive, experience-based reasoning, and rely on simple principles to generate acceptable solutions efficiently. It is crucial, however, to adapt and refine these principles carefully to address the characteristics of the specific problem under consideration.

**Definition 3.8** (Metaheuristic method)**.** A *metaheuristic method* provides a high-level framework or general strategy for developing problem-specific heuristic algorithms tailored to particular types or classes of optimisation problems.

When tackling difficult nonlinear optimisation tasks, one may employ a simple heuristic as a *local improvement procedure*. Such an approach typically begins from an initial candidate solution and iteratively explores its neighbourhood to find an improved solution.
This process can be visualised as a *downhill search* along the landscape of the objective function (in the case of minimisation), descending until a local minimum is reached. In general, a well-designed local improvement method converges successfully to a local optimum. However, it typically halts once that local minimum is found, even if it is not globally optimal. Hence, the principal limitation of such approaches lies in their tendency to become trapped in local minimum points. On the positive side, these procedures often converge rapidly to locally optimal solutions, which makes them particularly attractive for complex or large-scale problems.
*What follows?* Building upon these concepts, it becomes evident that a more sophisticated framework is needed; more precisely, one that uses information about the problem to direct the search towards a global optimum rather than a merely local one.

**Definition** (Nature of metaheuristic)**.** The *nature of metaheuristic methods* lies in combining local improvement mechanisms with higher-level strategies that allow the search process to escape from local minimum points and perform a comprehensive exploration of the feasible region.

In summary, the *purpose of metaheuristic approaches* is to address optimisation problems that are too complex or large to be solved effectively with exact algorithms. For practical examples, see [58, 85, 86].
In the following sections, we examine three representative metaheuristic techniques:

1. *Tabu search*;
2. *Simulated annealing*;
3. *Genetic algorithms*.

#### 3.2.3.1 Tabu search

It is a well-established metaheuristic technique that, drawing on intuitive and practical ideas, enables the search process to escape from local minimum points.
In essence, it operates similarly to a local improvement method, with the key distinction that each new candidate solution is not necessarily required to improve upon the previous one. Indeed, the algorithm typically incorporates a local search routine specifically adapted to the problem under consideration.
The distinctive feature of this approach lies in its persistence: it allows moves that do not immediately improve the objective function, thereby enabling the search to explore the neighbourhood of a local minimum more effectively. This characteristic prevents premature termination of the optimisation process. Broadly speaking, this approach can be viewed as a form of the *steepest ascent/mildest descent* strategy; as a matter of fact, at each iteration, the algorithm selects the move that yields the greatest reduction in the objective function (or, if no such move exists, the one that results in the smallest deterioration).
However, one potential drawback arises: after departing from a local minimum point, the search process may repeatedly return to that same local region, causing cycling behaviour. To prevent this, the *tabu search* method introduces a mechanism that temporarily prohibits moves leading back to recently visited solutions.

**Definition** (Tabu list)**.** A *tabu list* is a short-term memory structure that stores information about recently executed moves, thereby forbidding their immediate repetition.

The efficiency of the tabu search can be further enhanced through two complementary strategies:

- *Intensification:* focuses the search on promising regions of the feasible space that appear to contain high-quality solutions, encouraging deeper exploration of these areas;
- *Diversification:* directs the search toward unexplored or less-visited regions of the feasible space, promoting broader coverage and preventing premature convergence.

This procedure is summarised in Algorithm 6.
Once this algorithm has been chosen, one still needs to properly adapt it to the problem. As a matter of fact, one should define:

- the local search procedure;
- how the chosen procedure defines the neighbourhood structure, specifying which solutions are considered immediate neighbours of a given trial solution;
- how tabu moves should be listed in the tabu list;
- what tabu move should be added to the tabu list at each iteration;
- the time of permanence of a tabu move in the tabu list;
- the stopping rule.

**Algorithm 6** Tabu search

1: **Input:** Feasible initial trial solution $x_0$.
2: **Output:** Approximate minimum solution.
3: **for** each iteration $k = 0, 1, 2, \ldots$ **do**
4: Generate the set of feasible moves in the local neighbourhood of the current solution $x_k$.
5: Exclude tabu moves unless they improve the best solution.
6: Select the move from the remaining candidates that provides the best improvement.
7: Update the current solution $x_{k+1}$ by adopting the selected move.
8: Update the tabu list accordingly.
9: **end for**
10: **return** Current solution $x_k$ as approximate minimum.

#### 3.2.3.2 Simulated annealing

This metaheuristic technique focuses on locating the highest peak or the deepest valley of the objective function, depending on whether the problem is one of maximisation or minimisation. Since the precise location of this optimum is unknown, the algorithm begins by exploring the feasible region through random steps in various directions. This stochastic exploration allows the method to investigate a broad portion of the search space.
Over time, as the process continues, most of the accepted moves tend to lead either upward (in maximisation problems) or downward (in minimisation problems). Consequently, the algorithm progressively concentrates its search within the regions of the feasible space that are most likely to contain the optimal solution. Eventually, it converges toward the highest peak or lowest valley.
More specifically, at each iteration, the method transitions from the current candidate solution to one of its immediate neighbours within the local neighbourhood. What distinguishes this technique from other local search approaches is the criterion used to decide whether a neighbouring solution should become the next trial solution.
Let

- $z_c$ denote the objective function value corresponding to the current solution;
- $z_n$ denote the objective function value for the neighbouring candidate solution;
- $T$ denote a control parameter, often called the *temperature*, that governs the likelihood of accepting a non-improving move.

**Definition** (Move selection rule)**.** The *move selection rule* determines how the next candidate solution is chosen among the neighbours of the current one. The decision depends on both the difference between $z_c$ and $z_n$ and the temperature parameter $T$.
For a *minimisation* problem:

$$\begin{aligned} &\text{if } z_n \le z_c, && \text{always accept the candidate;} \\ &\text{if } z_n > z_c, && \text{accept the candidate with probability} \\ & && \text{Prob}\{\text{acceptance}\} = e^x, \quad \text{where } x = \frac{z_c - z_n}{T}. \end{aligned}$$

For a *maximisation* problem:

$$\begin{aligned} &\text{if } z_n \ge z_c, && \text{always accept the candidate;} \\ &\text{if } z_n < z_c, && \text{accept the candidate with probability} \\ & && \text{Prob}\{\text{acceptance}\} = e^x, \quad \text{where } x = \frac{z_n - z_c}{T}. \end{aligned}$$

*Remark.* The parameter $T$ plays a crucial role, as it regulates the degree of randomness in the acceptance of non-improving moves. This flexibility distinguishes the simulated annealing method from approaches such as tabu search. A higher value of $T$ allows for more frequent acceptance of uphill (in minimisation) or downhill (in maximisation) moves, promoting exploration of the search space.

This probabilistic acceptance rule can be viewed in relation to a random number uniformly distributed in $[0, 1]$:

- For *minimisation*: if the random number > Prob{acceptance}, reject the step; otherwise, accept it.
- For *maximisation*: if the random number < Prob{acceptance}, accept the step; otherwise, reject it.

The mathematical form of the move selection rule is directly inspired by the physical process of *annealing* in metallurgy, where a material is slowly cooled to reach a state of minimum energy.
A second key element of this algorithm is the design of the *temperature schedule*. This involves three crucial choices:

1. Selecting an initial temperature $T$ that is sufficiently large to permit broad exploration;
2. Defining a sequence of progressively smaller temperature values to gradually reduce randomness;
3. Determining the number of iterations to perform at each temperature level.

Together, these elements balance exploration and exploitation, guiding the algorithm toward convergence to a near-optimal or optimal solution.
This procedure is summarised in Algorithm 7.

**Algorithm 7** Simulated annealing

**Input:** Feasible initial trial solution $x_0$.
**Output:** Approximate minimum or optimum solution.
**for** each iteration **do**
    Use the move selection rule to define the next trial solution.
    **if** no immediate neighbour is accepted **then stop**.
    **end if**
    *Update temperature:* After the scheduled moves at $T$, set $T$ to the next value and continue.
    *Stopping rule:* After completing the scheduled moves at the lowest temperature, **stop**.
**end for**
**return** The best trial solution found during all iterations as the final solution.

Once the simulated annealing algorithm has been selected, it must be properly adapted to the specific optimisation problem at hand. In particular, several design choices must be clearly defined:

- the procedure used to generate the initial trial solution;
- the definition of the neighbourhood structure, which determines which solutions are considered immediate neighbours (i.e., reachable in a single iteration) of the current solution;
- the rule by which one randomly selects a neighbouring solution as the next candidate within the move selection process;
- a temperature schedule appropriately tailored to the problem characteristics.

**Hybrid algorithm** It is worth noting that simulated annealing can be effectively combined with features of the tabu search method, resulting in a *hybrid algorithm* that leverages the advantages of both techniques. Several such combinations have been shown to enhance search performance. For example:

1. The *strategic oscillation* mechanism from tabu search can be integrated into the temperature schedule of simulated annealing. This modification accelerates the temperature decrease relative to the standard schedule, and then deliberately oscillates the temperature back and forth around the levels where high-quality solutions have previously been discovered.

2. The *candidate-list strategy* from tabu search can also be incorporated into the move selection rule of simulated annealing. In this hybrid formulation, multiple neighbouring candidates are first evaluated to identify potential improving moves before applying the probabilistic acceptance rule that determines whether the selected candidate becomes the next trial solution.

#### 3.2.3.3 Genetic algorithms

Genetic algorithms have proven highly effective in optimisation contexts that require extensive exploration of the feasible region while progressively converging toward high-quality solutions. These algorithms are inspired by the principles of natural evolution, particularly the concepts of *natural selection* and *survival of the fittest*. The following provides an overview of their underlying mechanisms.
*Natural Selection:* A useful analogy can be drawn between feasible solutions of an optimisation problem and members of a biological population. In this analogy, the objective function value represents the *fitness* of each individual solution. Rather than focusing on a single solution, a genetic algorithm maintains and evolves an entire *population* of candidate solutions simultaneously. At each iteration, often referred to as a *generation*, the current population consists of a set of trial solutions being evaluated. These can be viewed as the living members of a species at that stage of evolution. As in nature, a subset of the population, typically the fittest individuals, are selected to survive and reproduce. Pairs of these individuals, chosen at random, act as *parents* and generate *offspring* (new trial solutions) that inherit traits, or *genes*, from both parents.
Because fitter individuals are more likely to reproduce, successive generations tend to yield *improving populations*, that is, groups of trial solutions whose average quality increases over time. In addition, the algorithm allows for occasional *mutations*, where certain offspring acquire new features not directly inherited from either parent. Such mutations introduce diversity into the population and may lead to the discovery of better solutions.
In summary, by emulating the evolutionary process and adhering to the principle of the "survival of the fittest," a genetic algorithm iteratively refines its population of solutions. This evolutionary search typically converges to a solution that represents, or closely approximates, the global optimum. This procedure is summarised in Algorithm 8.
Once this algorithm has been chosen, one still needs to properly adapt it to the problem. As a matter of fact, one should define:

- the size of the population;
- the criteria to select members of the current population to become parents;
- how the children's features should come from the parents' ones;
- the stopping rule to use.

### 3.2.4 Selecting and combining optimisation algorithms

At the beginning of the current Chapter, a variety of optimisation techniques were examined, with particular attention to their practical implementation and range of applicability. The field of

**Algorithm 8** Genetic algorithms

1: **Input:** Initial population of feasible trial solutions; evaluate fitness of each member
2: **Output:** Approximate minimum or optimum solution
3: **for** each iteration **do**
4: Select an even number of parents via a fitness-biased random process.
5: Randomly pair parents; each pair yields two mixed, occasionally mutated offspring.
6: Discard infeasible children (considered miscarriages).
7: Retain the fittest children to form the new population of the same size.
8: Evaluate fitness of each new population member.
9: *Stopping rule:* Stop after a predefined number of iterations or other criteria.
10: **end for**
11: **return** The best trial solution found

optimisation is vast and continuously advancing, providing researchers and practitioners with an extensive toolkit to address diverse problem classes. Each algorithm is typically tailored to specific types of challenges, reflecting the inevitable trade-offs among accuracy, computational cost, and robustness. Despite major progress, several persistent difficulties remain, particularly those related to the treatment of large-scale systems, the assurance of global optimality, and the management of computational resources.

Looking forward, research trends increasingly point toward the development of *hybrid algorithms* that combine complementary strategies. The advent of emerging technologies, such as quantum computing, further expands the frontier of possible optimisation approaches. Likewise, the integration of optimisation with machine learning and data-driven methodologies is creating powerful new paradigms. This interdisciplinary convergence is expected to yield innovative frameworks capable of addressing ever more complex real-world problems.

Selecting an appropriate algorithm is a crucial step that directly affects both solution quality and computational efficiency. A well-chosen method must achieve an appropriate balance between numerical precision and resource consumption, including time and memory requirements. Ideally, the chosen algorithm should also exhibit robustness against perturbations such as noise in input data or rounding errors arising from finite-precision arithmetic. However, these objectives often conflict, necessitating a careful evaluation of trade-offs.

The analytical properties of the functions involved play a central role in determining the most suitable algorithm. Smoothness and continuity allow numerical methods to exploit local information about the problem's structure, thereby improving convergence behaviour. This is especially important when distinguishing between local and global optima. In nonlinear optimisation, most methods tend to converge to local minimum points that are optimal within a neighbourhood but not necessarily across the entire domain. Achieving *global* optimality remains far more challenging, particularly for non-convex problems where the objective function admits multiple local minimum points. For convex problems, by contrast, every local minimum is guaranteed to be global. Consequently, global optimisation methods must incorporate mechanisms that explore the entire solution landscape rather than relying solely on local descent.

In the context of identifying multiple critical points of the action functional defined in (2.6), the inherent complexity of the problem motivates a composite algorithmic strategy. The proposed approach combines two complementary methods: a multi-population adaptive inflationary differential evolution algorithm (MP-AIDEA, see [69]) and a domain decomposition technique referred to as the *Lattice* method.

The MP-AIDEA algorithm integrates the principles of differential evolution with features inspired by monotonic basin hopping, such as restart and local search mechanisms, enabling effective navigation through intricate optimisation landscapes. In parallel, the Lattice technique promotes systematic exploration by subdividing the domain into smaller subregions. When employed together, these two methods provide a powerful hybrid framework capable of identifying multiple

feasible and physically meaningful solutions—specifically, periodic orbits that satisfy the governing differential equations.

### 3.2.5 MP-AIDEA

Building upon the general concepts of metaheuristic optimisation introduced in Section 3.2.3, we now focus on a specific algorithm particularly well-suited for high-dimensional and complex optimisation problems: the *Multi-Population Adaptive Inflationary Differential Evolution Algorithm* (MP-AIDEA), first presented in [69]. MP-AIDEA is an advanced, population-based evolutionary framework designed to address single-objective global optimisation problems in continuous domains. It extends the core ideas of Differential Evolution (DE) [87] by introducing multiple interacting populations, adaptive control parameters, and restart mechanisms, collectively enhancing both global exploration and local refinement.

At its core, MP-AIDEA employs a *multi-population strategy*, where several groups of candidate solutions, referred to as *agents*, evolve in parallel. Each population evolves independently but intermittently exchanges information with others, increasing population diversity and mitigating the risk of premature convergence. Furthermore, the algorithm incorporates restart mechanisms and local search procedures inspired by *Monotonic Basin Hopping* (MBH), enabling efficient traversal of complex, multimodal fitness landscapes where conventional optimisation methods are often trapped in local minimum points.

Let $N_{\mathrm{P}}$ denote the number of populations and $N_{\mathrm{A}}$ the number of agents per population. The MP-AIDEA procedure consists of the following main stages:

1. *Differential Evolution (DE):* The algorithm begins by running $N_{\mathrm{P}}$ independent DE processes [87], one for each population. Differential Evolution generates new candidate solutions by combining existing ones through stochastic operations. Initially, agents are uniformly distributed across the parameter space to ensure wide exploration. Each DE process consists of $N_{\mathrm{S}}$ stages, where new parameter vectors are created via three main operations:

   - *Mutation:* combining selected parameter vectors at random to generate a trial vector;
   - *Crossover:* mixing components of the trial vector with those of the current agent to promote diversity;
   - *Selection:* comparing the new candidate with the current agent and retaining the better-performing solution.

   This evolutionary cycle continues until a stopping criterion is met. For further details, see [62, 69, 87].

2. *Local search:* Following the DE phase, a local optimisation step refines the best agent in each population. This step is only applied if the current best solution does not lie within the basin of attraction of a previously identified local minimum. Here, the *basin of attraction* refers to the subset of the search space that leads to convergence toward a specific local minimum.

3. *Basin hopping:* To enhance exploration and avoid stagnation, each population is then restarted near the local minimum found in the previous phase. This *Basin Hopping* mechanism [88, 89] enables transitions between distinct basins of attraction, allowing the algorithm to escape local minimum points and continue searching for better solutions.

By combining Differential Evolution, Local Search, and Basin Hopping, MP-AIDEA effectively balances exploration and exploitation. This integrated design allows populations to traverse the so-called *funnel structure* of the objective landscape [90], progressively transitioning from one local minimum to another until the global minimum is reached.

A distinctive feature of MP-AIDEA is its mechanism for preventing redundant rediscovery of the same local minimum points. Whenever a population re-enters the basin of attraction of an already-detected minimum, it is reinitialised in a different region of the search space. To manage this process, an archive $\mathfrak{A}$ is maintained to store all previously identified minimum points together with the corresponding population states at each restart. This ensures a diverse, non-redundant exploration of the feasible domain. Further details can be found in [69].

**New implementation in `Julia`.** A recent update of MP-AIDEA (for more details see [91]) has been developed in the `Julia` programming language [70], extending the original version presented in [69]. The `Julia` implementation introduces enhanced performance, improved scalability, and more flexible modular structures for algorithmic experimentation. In particular, the new version benefits from `Julia`'s just-in-time (JIT) compilation and native parallel computing capabilities, enabling faster population-level operations and smoother integration with scientific computing packages. This upgrade makes MP-AIDEA more efficient, portable, and easier to adapt for large-scale and high-precision optimisation tasks.

### 3.2.6 Lattice algorithm

The principal objective of the Lattice algorithm is to explore the optimisation domain in a systematic and adaptive manner, identifying as many local minimum points as possible. The method achieves this by decomposing the search domain into progressively smaller subregions, guided by the spatial distribution of the local minimum points discovered during the optimisation process. This adaptive partitioning refines the exploration strategy over time, allowing the search to become increasingly focused and efficient.
Initially, the Lattice algorithm requires an initial set of local minimum points, which are obtained by running a chosen optimisation method. Once these minimisers are identified, they serve as reference points to guide the domain decomposition. At each iteration, referred to as a *layer* $j$, where $j = 1, \dots, \mathrm{NoL} \in \mathbb{N}$, the domain is divided into two subregions. The optimisation algorithm is then applied independently within each subregion to locate additional minimum points, which subsequently inform the next layer of decomposition. This hierarchical refinement process enables an increasingly detailed exploration of the solution space. In this work, we employ MP-AIDEA (Section 3.2.5) as the underlying optimisation engine for each stage of the decomposition; however, other global optimisation algorithms could be used in its place.
Let $\Xi$ denote the domain of the optimisation parameters. The main challenge lies in determining how to partition $\Xi$ into two subregions, $\Xi_1$ and $\Xi_2$. A simple equal division is generally suboptimal, as it does not account for the spatial distribution of the known minimum points. Instead, the Lattice algorithm performs an *unequal partition*, focusing the search on regions with sparse distributions of minimum points while still retaining coverage of denser regions. In this way, the algorithm allocates computational effort more efficiently across the domain.
To illustrate the procedure, consider the following example.

**Example 3.9.** For simplicity, let us focus on the first decomposition layer, $j = 1$. Let $a, b, c, d \in \mathbb{R}$ with $a < b$ and $c < d$. Define the optimisation parameters $x \in I = [a, b] \subset \mathbb{R}$ and $y \in J = [c, d] \subset \mathbb{R}$, such that the full domain is $\Xi = I \times J$. This construction naturally extends to higher dimensions $n > 2$, where $\Xi \subset \mathbb{R}^n$.
Let $\mathcal{M} = \{m_i\}_{i=1}^{M}$, with $M < +\infty$, represent the set of local minimum points detected by the optimisation procedure. Each $m_i = (x_i, y_i) \in I \times J$ corresponds to a point where the objective function attains a local minimum value. Our aim is to analyse how the minimum points in $\mathcal{M}$ are distributed within $\Xi$ and to use this information to guide the next domain partition.
The process proceeds as follows:

1. Evaluate the distribution of the minimum points $m_i \in \mathcal{M}$ with respect to each optimisation parameter using a chi-square test to assess uniformity.

2. Identify the parameter exhibiting the strongest deviation from uniformity—that is, the one with the smallest p-value. If multiple parameters share the same p-value, the first is selected. Suppose this parameter is $y$; then the minimum points are more unevenly distributed along the $y$-axis.

3. Compute the 70th percentile of the identified parameter values, denoted by $\bar{y}_m \in J$.

4. Partition the parameter domain accordingly:

$$J = J_1 \cup J_2, \quad \text{where } J_1 = [c, \bar{y}_m] \quad \text{and} \quad J_2 = [\bar{y}_m, d].$$

5. Finally, divide the entire domain using this partition:

$$\Xi = I \times J = [a, b] \times [c, d], \quad \Xi_1 = I \times J_1, \quad \Xi_2 = I \times J_2.$$

   This ensures that

$$\Xi_1 \cup \Xi_2 = \Xi,$$

   guaranteeing complete domain coverage with no overlap or omission.

This decomposition process is repeated for each layer $j = 1, \ldots, \text{NoL}$, producing a progressively refined partition of the domain. Figure 3.2 illustrates a representative outcome for the case where NoL = 3.

The main concept behind the Lattice algorithm is summarised in Figure 3.1, which illustrates the first two layers for simplicity.

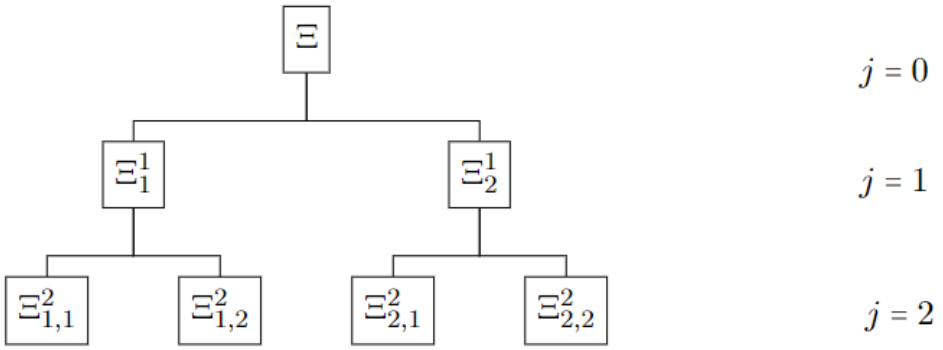



Figure 3.1: Lattice Scheme (Figure 2 [92])

The Lattice algorithm is shown in Algorithm 10, where it is implemented as a recursive function. Algorithm 9 provides the steps to perform the division for just one layer.

**Algorithm 9** One layer algorithm (Algorithm 1 [92])

**function** ONE_LAYER_FUNCTION($\Xi$)
    **Input:** A domain $\Xi$.
    **Output:** Two subregions $\Xi_1$ and $\Xi_2$, with $\Xi_1 \bigcup \Xi_2 = \Xi$.
    **Step 1.** Find the p-value ($p$) and column index (min_p_index) for the column with the minimum p-value in domain $\Xi$. If multiple minimum p-values exist, pick first.
    **Step 2.** Calculate the 70th percentiles for the column with the minimum p-value
    **Step 3.** Split the min_p_index column at the 70th percentile into $\Xi_1$ and $\Xi_2$.
    **Step 4. return** $\Xi_1$ and $\Xi_2$.
**end function**

**Algorithm 10** Lattice algorithm (Algorithm 2 [92])

**procedure** LATTICE_FUNCTION($\Xi$, NoL, Output)
  **if** $j$ = NoL **then**
    **return** Output
  **else**
    **Step 1.** $\Xi_1, \Xi_2 \leftarrow$ OUTPUT {ONE_LAYER_FUNCTION($\Xi$)}
    **Step 2.** Output[$j$] $\leftarrow$ OUTPUT {LATTICE_FUNCTION($\Xi_1$, $j+1$, Output) $\cup$
    LATTICE_FUNCTION($\Xi_2$, $j+1$, Output)}
    **Step 3.** $j \leftarrow j+1$
    **Step 4. return** Output
  **end if**
**end procedure**

## 3.3 Solutions of the symmetrical $n$-body problem

The integration of the MP-AIDEA algorithm (presented in Section 3.2.5) with the Lattice algorithm (described in Section 3.2.6) establishes a robust and adaptive framework for global optimisation in complex, high-dimensional landscapes. The two methods complement each other in a natural way: MP-AIDEA efficiently explores the parameter space through evolutionary and basin-hopping strategies, while the Lattice algorithm systematically refines the search domain based on the spatial distribution of the discovered minimum points. Together, they provide both depth and breadth in exploration, enabling a more comprehensive analysis of the solution space than either method could achieve independently. To initialise the Lattice algorithm, a collection of local minimum points is required, denoted by

$$\mathcal{M} = \{m_i\}_{i=1}^{M}, \qquad M < +\infty.$$

Traditionally, constructing such a set would require multiple independent runs of an optimisation algorithm, an approach that is both time-consuming and computationally inefficient. In contrast, the proposed hybrid framework takes advantage of the information already generated by MP-AIDEA during its execution.
Specifically, MP-AIDEA maintains a dynamic archive, denoted $\mathfrak{A}$, which stores all the local minimum points identified throughout the optimisation process, as well as the associated population states at each restart. This archive naturally serves as the foundation for the Lattice algorithm, providing a consistent and evolving representation of the landscape's structure. Hence, we define

$$\mathcal{M} = \mathfrak{A}.$$

By reusing this internally collected information, the hybrid method eliminates the need for redundant computations and ensures that the domain decomposition in the Lattice algorithm is informed by the most up-to-date and reliable data available.

The integration of the MP-AIDEA algorithm (presented in Section 3.2.5) with the Lattice algorithm (described in Section 3.2.6) establishes a robust and adaptive framework for global optimisation in complex, high-dimensional landscapes. The combined MP-AIDEA-Lattice framework is implemented as a recursive procedure: each layer of the Lattice decomposition invokes an MP-AIDEA optimisation step to search within its corresponding subregion. The recursion continues until a specified stopping condition is met, such as the exhaustion of layers or the absence of newly detected minimum points. The overall process is summarised in Algorithm 12, while the specific operations performed within a single generation are detailed in Algorithm 11.

*Remark.* In this initial approach, we do not consider any exit flag for the MP-AIDEA algorithm. Note that **Step 3** of Algorithm 11 is crucial, as it is possible that some subregions of the domain

**Algorithm 11** One generation algorithm (Algorithm 3 [92])

**Step 0.** Set the number of layers for each generation to 1, NoL = 1. Define initial parameters MP-AIDEA_param = ($f$, optimisation parameters, $\beta$, $\Xi$, options), where $f$ is the cost function, $\beta$ the number of parameters, and $\Xi \subset \mathbb{R}^\beta$ denotes their domain.
**function** ONE_GENERATION_FUNCTION(MP-AIDEA_param, $\Xi$)
    **Step 1.** Set initial values for MP-AIDEA's parameters in $\Xi$.
    **Step 2.** Run MP-AIDEA Algorithm and set $\mathcal{M} = \mathfrak{A}$.
    **Step 3.** Analyse the set $\mathcal{M}$ to collect all critical points. Hence, given a tolerance tol $\ll 1$,

$$m_i \text{ is considered a critical point of } f \text{ if } \nabla f(m_i) \leq \text{tol}. \tag{3.19}$$

    **Step 4.** Apply the Lattice algorithm to $\mathcal{M}$, setting $\Xi \coloneqq \mathcal{M}$.
    Obtain $\Xi_1$ and $\Xi_2$ as $\Xi_1, \Xi_2 \leftarrow$ ONE_LAYER_FUNCTION($\Xi$).
    **Step 5. return** MP-AIDEA_Output, $\Xi_1$ and $\Xi_2$.
**end function**

**Algorithm 12** MP-AIDEA & Lattice algorithm (Algorithm 4 [92])

**Step 0.** Set NoL $\in \mathbb{N}$, set NoL = 1, $\forall$ NoL. Define initial parameters
MP-AIDEA_param = ($f$, optimisation parameters, $\beta$, $\Xi$, options).
**procedure** MP-AIDEA_LATTICE_FUNCTION($\Xi$, MP-AIDEA_param, NoG, Output)
    **if** $j$ = NoG **then**
        **return** Output
    **else**
        **Step 1.** Run ONE_GENERATION_FUNCTION(MP-AIDEA_param, $\Xi$) to obtain
        MP-AIDEA_Output, $\Xi_1$, and $\Xi_2$.
        Set $\tilde{\Xi} \coloneqq \Xi_1 \bigcup \Xi_2$. In MP-AIDEA_param (i)-(iv), update $\Xi$ with $\tilde{\Xi}$.
        **Step 2.**
        Output[$j$] $\leftarrow$ {MP-AIDEA_LATTICE_FUNCTION($\tilde{\Xi}$, MP-AIDEA_updated_param $j$, Output)
        **Step 3.** $j \leftarrow j + 1$
        **Step 4. return** Output
    **end if**
**end procedure**

do not contain actual minimum points, even though the archive $\mathfrak{A}$ is not empty.

*Remark.* Furthermore, for each layer processed by MP-AIDEA, the subregions are independent of one another. This allows the algorithm to run in parallel across layers, thus reducing the overall runtime.

The synergy between the two algorithms yields several benefits. First, the decomposition process in the Lattice algorithm becomes fully adaptive, as it is guided by the evolving knowledge extracted by MP-AIDEA. Second, this integration enhances computational efficiency by reusing the populations and minimum points already detected, thereby avoiding unnecessary reinitialisations. Third, the structure of the combined framework lends itself naturally to parallel implementation, since the subregions defined by the Lattice algorithm can be processed independently by separate MP-AIDEA instances. This parallelism significantly reduces total runtime, especially for large and highly multimodal problems.
Overall, this modular structure facilitates both scalability and extensibility, allowing straightforward integration of alternative optimisation engines or adaptive control mechanisms in future developments.

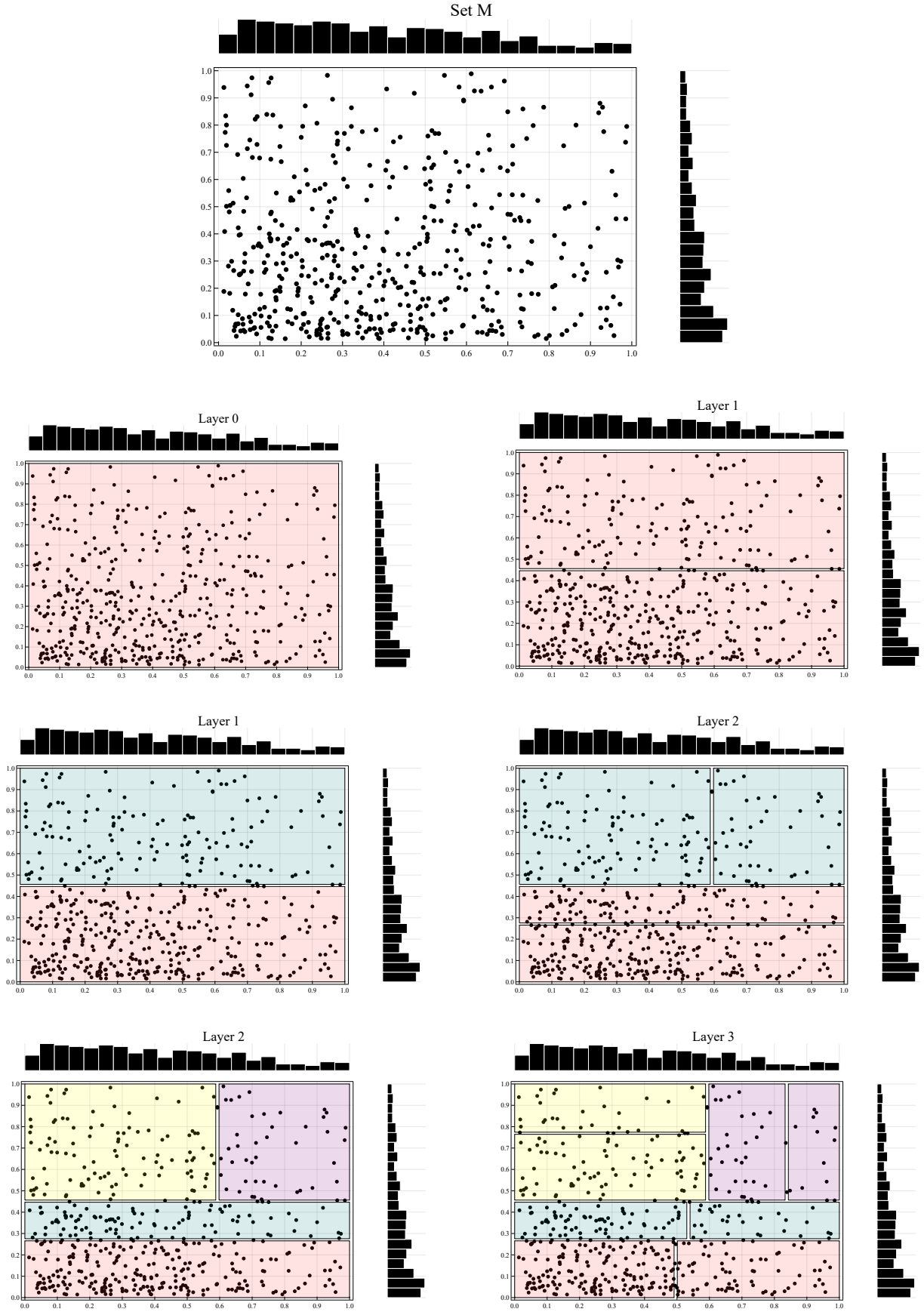



Figure 3.2: Visualisation of a two-dimensional example of the Lattice algorithm. The first figure displays the set $M$ along with the distributions of the variables. The subsequent images illustrate the progressive division of the domain at each layer, following the process described in Example 3.9 (Figure 1 [92]).

# Chapter 4

# Analysing critical points

In this Chapter we shift from the computation of candidate periodic solutions to the analysis of their qualitative structure within the action landscape. Assuming the trajectories produced in Chapter 3 are critical points of the discretised action functional, we investigate their local character, distinguishing local minimum points, saddle points, and more degenerate stationary configurations and how such features are organised globally. The theoretical notions employed in this Chapter, such as the Morse index, landscape-level distances, and winding-based topological invariants, are well established in the optimisation and dynamical systems literature. However, their systematic adaptation to the discretised variational framework of the equivariant $n$-body problem, together with their integration into a coherent computational pipeline, is entirely novel. In particular, these concepts are reformulated so as to be applicable to high-dimensional, constrained, and symmetry-reduced discretisations of the action functional, and are implemented as concrete numerical diagnostics for the post-processing of computed orbits.
The Chapter is organised as follows. Section 4.1 introduces a practical procedure for classifying the computed solutions via the discrete Morse index, obtained from the spectrum of a discretised Hessian. Section 4.2 then provides a concise graphical description of the landscape by grouping minima into levels and computing the intra-level and trans-level distances $\rho_{\mathrm{IL}}$ and $\rho_{\mathrm{TL}_k}$ (following [58]). Finally, Section 4.3 introduces the winding number as a topological invariant that helps distinguish planar periodic trajectories by their rotational behaviour around a reference point. The emphasis of this Chapter is methodological: the notions recalled from Morse theory, landscape diagnostics, and winding-based invariants are specialised to our discretised variational setting and assembled into concrete computational tools for post-processing the orbits produced by the optimisation stage.

## 4.1 Discrete Morse index

The analysis of critical points has its origins in the classical Morse theory, developed by Marston Morse [93] in the 1930s, which establishes a deep connection between the topology of smooth manifolds and the critical points of smooth functions defined on them. Central to this framework is the *Morse index*, which counts the number of negative eigenvalues of the Hessian matrix at a critical point. Intuitively, the Morse index quantifies the degree of instability: minimum points have index 0, while saddle points and Mountain-Pass configurations correspond to higher indices.
A key generalisation of the Morse Index Theorem was introduced by Smale [94], extending the classical theory from one-dimensional to multi-variable systems. This laid the groundwork for modern Morse theory and its applications in Hamiltonian and variational frameworks.
In Lagrangian and Hamiltonian mechanics, the Morse index plays a central role in describing the qualitative nature of periodic orbits and bifurcations between dynamically distinct regimes (see, e.g., [95–97]). Within the variational formulation of the $n$-body problem, it provides crucial information about the nature of critical points of the action functional, distinguishing between stable configurations and saddle-type orbits [34, 98].

In the planar three-body problem, Barutello, Jadanza, and Portaluri [99] computed the Morse index of Lagrangian circular orbits under $\alpha$-homogeneous and logarithmic potentials, identifying regions of linear stability. Their work, based on the Maslov-type index theory of Long, Hu, and Sun, showed that stability boundaries correspond to curves where the Morse indices of iterates change discretely. Similarly, Hu and Sun [100] related the Morse index to the spectral stability of elliptic Lagrangian solutions via the monodromy matrix. Extending this perspective, Barutello, Hu, Portaluri, and Terracini [101] developed a general index theory for asymptotic motions under singular potentials, thereby broadening the concept of Morse theory to colliding and parabolic solutions in the $n$-body problem, and relating the Morse/Maslov index of the trajectory with its asymptotic properties. Fukuda, Fujiwara, and Ozaki [102] analysed figure-eight choreographies in the equal-mass three-body problem, showing numerically that variations in the Morse index coincide precisely with bifurcation points, confirming the index as both a necessary and sufficient condition for bifurcation.
Together, these works trace the evolution of Morse-theoretic methods in celestial mechanics and Hamiltonian dynamics, from smooth variational formulations to singular and multi-parameter settings, highlighting the Morse index as a unifying tool for studying stability, bifurcation, and structural transitions in the $n$-body problem.

**From continuous to discrete Morse theory.** The transition from the continuous to the discrete setting was formalised by Forman [103], who introduced *discrete Morse theory*, offering a combinatorial reformulation of Morse's ideas. Building on this discrete–variational viewpoint, Fukuda, Fujiwara and Ozaki [102] developed a practical scheme for computing the *discrete Morse index* in variational dynamical problems. Their approach consists in discretising the action functional, restricting the second variation to a finite-dimensional subspace, typically via variational or Fourier discretisations, and determining the number of negative eigenvalues of the resulting matrix. This procedure effectively bridges analytical theory and computational practice, enabling a numerical estimation of the Morse index directly from discretised trajectories.
The Morse index $n^-$ of a critical point $p$ provides essential information about the local behaviour of a smooth function $f$. It is a non-negative integer (possibly $+\infty$) intrinsically associated with $p$, and one can make the following intuitive distinction:

- if $n^-(p) = 0$, then $p$ is a local minimum of the function $f$;
- if $n^-(p) > 0$, then $p$ is a saddle point of Morse index $n^-$ for the function $f$.

Therefore, the Morse index provides a natural way to classify the critical points identified in Chapter 3, distinguishing between local minimum points and saddle points. This classification will later prove essential in the study of Mountain Pass solutions in Chapter 5.

Let us now give a more precise definition. Consider a functional $F : H \to \mathbb{R}$ of class $\mathcal{C}^2$, with $H = W^{1,2}(\mathbb{T}, \hat{\chi})$ and $\mathbb{T} = \mathbb{R}/\mathbb{Z} \simeq \mathbb{S}^1$.

**Definition 4.1** (Critical point)**.** A point $x$ is called a *critical point* of a functional $F$, denoted $x \in \mathrm{crit}(F)$, if

$$\mathrm{D}F(x)[v] = 0 \quad \forall\, v \in T_x W^{1,2}(\mathbb{T}, \hat{\chi}) \simeq W^{1,2}(\mathbb{T}, T_x\hat{\chi}).$$

Here $T_x W^{1,2}(\mathbb{T}, \hat{\chi})$ denotes the tangent space to the Sobolev manifold $W^{1,2}(\mathbb{T}, \hat{\chi})$ at $x$, naturally identified with the space $W^{1,2}(\mathbb{T}, T_x\hat{\chi})$ of Sobolev vector fields along $x$. The symbol $\mathrm{D}F(x)$ stands for the Fréchet differential of $F$ at $x$, i.e. the bounded linear map

$$\mathrm{D}F(x) : T_x W^{1,2}(\mathbb{T}, \hat{\chi}) \to \mathbb{R}, \qquad \mathrm{D}F(x)[v] = \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} F(x + \varepsilon v).$$

We now introduce the *index form*, which plays a central role in the definition of the Morse index.

**Definition 4.2** (Index form)**.** Let $F : H \to \mathbb{R}$ be a $\mathcal{C}^2$ functional, and let $x \in \operatorname{crit}(F)$. The *index form* of $F$ at $x$ is the symmetric bilinear form

$$I : H \times H \longrightarrow \mathbb{R}, \qquad I(u,v) = \mathrm{D}^2F(x)[u,v] = \langle Au, v\rangle_{L^2},$$

where, given $\mathcal{D}(A) \subset L^2$ the domain of $A$, $A : \mathcal{D}(A) \subset L^2 \to L^2$ denotes the (self-adjoint) linear operator associated with the second differential of $F$ at $x$. Here $\mathrm{D}^2F(x)$ denotes the second Fréchet differential of $F$ at $x$, that is,

$$\mathrm{D}^2F(x)[u,v] = \left.\frac{\partial^2}{\partial s\,\partial t}\right|_{s=t=0} F(x + su + tv),$$

and $A$ is the corresponding Hessian (or Jacobi) operator satisfying $I(u,v) = \langle Au, v\rangle_{L^2}$ for all $u, v \in \mathcal{D}(A)$.

**Definition 4.3** (Morse index of $x$)**.** Let $x \in \operatorname{crit}(F)$ and $W$ be a subspace of $H$. The *Morse index* of $x$ is defined as

$$n^-(x) = \sup_{W \subseteq H} \left\{ \dim W \mid I\big|_{W\times W} \text{ is negatively defined} \right\} \in \mathbb{N} \cup \{+\infty\}.$$

When $A$ is a differential operator and $A \in \mathcal{C}(H)$, we have $n^-(A) = \dim \mathbb{E}^-(A)$.

One can introduce an analogous notion of *discrete Morse index*.

**Definition 4.4** (Discrete Morse index)**.** Let $H = \bigcup_{n\in\mathbb{Z}} H_n$, where $H_n$ is a subspace of $H$ with $\dim H_n = n$ and $H_n \subset H_{n+1}$ for all $n \in \mathbb{N}$. Choose a basis for $H_n$ given by $\{e^{inx} \mid n \in \mathbb{Z}\}$, and fix $\bar{n}$ sufficiently large such that $\dim H_{\bar{n}} < +\infty$. Using Definition 4.2, set $I_{\bar{n}} \coloneqq I\big|_{H_{\bar{n}}}$. Then, under the same assumptions as in Definition 4.3, the *discrete Morse index* is defined as

$$\tilde{n}^-(x) = \sup_{W \subseteq H_{\bar{n}}} \left\{ \dim W \mid I_{\bar{n}}\big|_{W\times W} \text{ is negatively defined} \right\} \in \mathbb{N} \cup \{+\infty\}.$$

*Remark.* If it were known that $H_{\bar{n}} \oplus H_{\bar{n}}^{\perp} = H$ and that $I_{\bar{n}}{}^{\perp} \coloneqq I\big|_{H_{\bar{n}}^{\perp}} \geq 0$, we would be in an ideal situation. Unfortunately, this property remains unproven. In fact, the following conjecture is often stated:

**Conjecture:** If $\bar{n}$ is sufficiently large, then $n^-(I_n) = n^-(I_{n+1})$ for all $n \geq \bar{n}$.

Nevertheless, it is well known that for both Bolza and regular problems it holds that

$$n^-(I) < +\infty.$$

### 4.1.1 Computation of the Morse index

Following Definition 4.4 and [92, 104, 105], one can reckon the discrete Morse index of our critical points numerically. In this case, we can give a more practical definition:

**Definition 4.5** (Computational discrete Morse index, Definition 4.1 [92])**.** The *index* of a non-degenerate critical point $p$ of $f : H \to \mathbb{R}$ of class $\mathcal{C}^2$ is the dimension of the largest subspace of the tangent space to $M$ at $p$ on which the Hessian is negative definite. Therefore one could define the *discrete Morse index* $\tilde{n}^-$ of a critical point $p$ as the number of negative eigenvalues of the Hessian matrix $H_f(p)$.

Therefore we can now apply it to the critical points found in Chapter 3. One can calculate the discrete Morse index either on the fundamental domain $\mathbb{I}$ or, once the orbit is built using the symmetries, on the entire period. Clearly, the first one can be less or equal to the latter one.

***On the fundamental domain.*** Following [60] and Section 2.6.3, one can compute the Hessian matrix of the action functional on the fundamental domain $\mathbb{I}$ in two ways:

1. *Using the discretisation through Fourier coefficients* in the same way as in (2.9). Therefore, recalling Section 2.6.3, we approximate the solution in the following way:

$$x_k(t) = x_{k,0} + t(x_{k,1} - x_{k,0}) + \psi_k(t), \quad t \in [0,T], \quad k = 1, \dots, n$$

where $\psi_k(0) = \psi_k(T) = 0$ and $T = l\pi$, $l \in \mathbb{N}$. Moreover, we write the function $\psi_k(t)$ using a Fourier polynomial $\psi_k(t) = \sum_{j_k=1}^{F} c_{j_k} \sin(\pi j_k t) \in \mathbb{R}^d$, $t \in [0,T]$; then we substitute this expression in the action functional defined in (2.2). One can find the expression of the Hessian matrix in [43].

2. *Using the discretisation through points* in the same way as in (2.6). It follows that a discretisation of the action functional defined in (2.2)

$$\mathcal{A}(\gamma) = \int_{\mathbb{I}} L(\gamma, \dot{\gamma}) \, \mathrm{D}t = \int_{\mathbb{I}} \frac{1}{2} \|\dot{\gamma}\|_M^2 + U(\gamma) \, \mathrm{D}t,$$

with $\gamma : [\mathbb{I}] \to \hat{\mathcal{X}}$, is given by

$$\mathcal{A}(\gamma) \sim \sum_{k=1}^{w-1} \left[ \sum_{i=1}^{n} \frac{m_i}{2} \frac{|x_i^{k+1} - x_i^k|^2}{h} + h \sum_{i<j} \frac{m_i m_j}{|x_i^k - x_j^k|} \right], \tag{4.1}$$

where $n$ is the number of bodies, $w$ is the number of times, $h = \frac{T}{w-1}$ is the time step and $x_i^k$ denotes the position of the $i$-th body at the $k$-th time.
It follows that the Hessian matrix has the following expression:

$$\begin{pmatrix}
\left[h\nabla^2 U(t_1) + \frac{2}{h}\mathrm{Id}\right] & \left[-\frac{1}{h}\mathrm{Id}\right] & \mathbf{0} & \dots & \mathbf{0} & \left[-\frac{1}{h}\mathrm{Id}\right] \\
\left[-\frac{1}{h}\mathrm{Id}\right] & \left[h\nabla^2 U(t_2) + \frac{2}{h}\mathrm{Id}\right] & \left[-\frac{1}{h}\mathrm{Id}\right] & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \left[-\frac{1}{h}\mathrm{Id}\right] & \left[h\nabla^2 U(t_3) + \frac{2}{h}\mathrm{Id}\right] & \left[-\frac{1}{h}\mathrm{Id}\right] & \dots & \dots \\
\dots & \dots & \dots & \dots & \dots & \mathbf{0} \\
\mathbf{0} & \dots & \dots & \dots & \mathbf{0} & \left[-\frac{1}{h}\mathrm{Id}\right] \\
\left[-\frac{1}{h}\mathrm{Id}\right] & \mathbf{0} & \dots & \mathbf{0} & \left[-\frac{1}{h}\mathrm{Id}\right] & \left[h\nabla^2 U(t_{w-1}) + \frac{2}{h}\mathrm{Id}\right]
\end{pmatrix} \tag{4.2}$$

where:

- Id represents an identity matrix of dimension $n \cdot dim$, where $dim$ is the number of coordinates of each body;
- $\mathbf{0}$ is a null matrix of dimension $(n \cdot dim) \times (n \cdot dim)$;
- $\nabla^2 U(t_i)$ is the Hessian matrix of the term $U(x)$ at the $i$-th time. It is a matrix of dimension $(n \cdot dim)\mathrm{x}(n)$.

***On the entire orbit.*** In a similar fashion, one can compute the Hessian matrix of the action functional on the *entire periodic orbit* rather than on the fundamental domain. Two equivalent discretisation approaches can again be adopted:

1. *Using Fourier coefficients*, the periodic trajectory of each body can be expanded using a truncated Fourier series of the form

$$x_k(t) = a_{k,0} + \sum_{j=1}^{F} \left[ a_{k,j} \cos\left(\frac{j\pi t}{T}\right) + b_{k,j} \sin\left(\frac{j\pi t}{T}\right) \right], \qquad t \in [0,T], \quad k = 1, \dots, n,$$

where $F$ denotes the number of Fourier modes and the coefficients $a_{k,j}, b_{k,j} \in \mathbb{R}^d$. Substituting this expansion into the action functional defined in (2.2) and differentiating twice with respect to the Fourier coefficients yields the discrete Hessian matrix associated with the entire periodic orbit. This formulation naturally enforces periodic boundary conditions and provides a symmetric, block-structured Hessian whose negative eigenvalues determine the discrete Morse index of the orbit.

2. *Using points*, one can proceed as in Section 2.6.3 by discretising the trajectory at $w$ equally spaced time nodes over the full period $T$. Starting from the discretised action functional in (4.1), the resulting Hessian matrix takes the form

$$\begin{pmatrix} \left[h\nabla^2 U(t_1) + \frac{1}{h}\mathrm{Id}\right] & \left[-\frac{1}{h}\mathrm{Id}\right] & \mathbf{0} & \dots & \dots & \dots & \mathbf{0} \\ \left[-\frac{1}{h}\mathrm{Id}\right] & \left[h\nabla^2 U(t_2) + \frac{2}{h}\mathrm{Id}\right] & \left[-\frac{1}{h}\mathrm{Id}\right] & \mathbf{0} & \dots & \dots & \dots \\ \mathbf{0} & \left[-\frac{1}{h}\mathrm{Id}\right] & \left[h\nabla^2 U(t_3) + \frac{2}{h}\mathrm{Id}\right] & \left[-\frac{1}{h}\mathrm{Id}\right] & \mathbf{0} & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \mathbf{0} \\ \dots & \dots & \dots & \mathbf{0} & \left[-\frac{1}{h}\mathrm{Id}\right] & \left[h\nabla^2 U(t_{w-1}) + \frac{2}{h}\mathrm{Id}\right] & \left[-\frac{1}{h}\mathrm{Id}\right] \\ \mathbf{0} & \dots & \dots & \dots & \mathbf{0} & \left[-\frac{1}{h}\mathrm{Id}\right] & \left[h\nabla^2 U(t_w) + \frac{1}{h}\mathrm{Id}\right] \end{pmatrix} \tag{4.3}$$

where

- Id represents an identity matrix of dimension $n \cdot dim$, where $dim$ is the number of coordinates of each body;
- $\mathbf{0}$ is a null matrix of dimension $(n \cdot dim) \times (n \cdot dim)$;
- $\nabla^2 U(t_i)$ is the Hessian matrix of the term $U(x)$ at the $i$-th time. It is a matrix of dimension $(n \cdot dim)\mathrm{x}(n)$.

Moreover the first and last blocks are now coupled to reflect the periodicity of the orbit, that is, $x_i^1 = x_i^w$ for each $i = 1, \dots, n$. As a matter of fact, only the initial point is taken into consideration as the initial point and final point of a periodic orbit coincide. The Hessian has dimension $(w \cdot n \cdot dim) \times (w \cdot n \cdot dim)$, and, as before, its negative eigenvalues correspond to the discrete Morse index of the critical point over the full period.

This dual approach, either through Fourier modes or pointwise discretisation, thus allows the computation of the discrete Morse index on the entire orbit, providing complementary numerical and analytical perspectives on the stability properties of periodic solutions.
Algorithm 13 shows how to reckon the discrete Morse index using the discretisation through points. One can adapt Algorithm 13 to the case of discretisation through Fourier coefficients.

**Algorithm 13** Morse index algorithm

1: **Input:** The positions $x_i(t_k)$ of the $i$-th body at the $k$-th time, $\forall i \in \{1, \dots, n\}$, $\forall k \in \{1, \dots, w\}$.
2: **Output:** Morse index of the solution.
3: **Step 1.** Build the Hessian of dimension $((w-1)\cdot n \cdot dim)\mathrm{x}((w-1)\cdot n \cdot dim)$ as in (4.2) or (4.3).
4: **Step 2.** Reckon the eigenvalues of the Hessian matrix found in the previous step.
5: **Step 3.** The Morse index of a critical point is the number of negative eigenvalues of the Hessian matrix on that point.

It is worth noting that, while the introductory part of this Chapter provided a historical and conceptual overview of Morse-theoretic methods in dynamical systems, the present section focuses on their practical implementation within the proposed framework. Here, the theoretical notions introduced earlier, such as the Morse index, index form, and discrete variational principles, are

translated into computational procedures applicable to any numerically obtained periodic orbit. In this way, the section connects the abstract theoretical background with the numerical analysis carried out in this work, linking classical Morse theory to its discrete, algorithmic counterpart.

### 4.1.2 Validation of the discrete Morse index

In order to validate the numerical procedure introduced in Section 4.1, we compare our results with those obtained by Barutello, Jadanza, and Portaluri [99], who analytically computed the Morse index of the Lagrangian circular solutions in the planar three-body-type problem. Although the physical setting and the family of orbits under investigation are different, the underlying variational and index-theoretic structure is analogous. In particular, both frameworks rely on the second variation of the Lagrangian action functional and on the count of negative eigenvalues of the associated Hessian operator.

In [99], the authors consider a planar three-body-type system governed by a singular potential of the form

$$U_\alpha(q) = \sum_{1\le i<j\le 3} \frac{m_i m_j}{|q_i - q_j|^\alpha}, \qquad \alpha \in (0,2), \tag{4.4}$$

and, in the logarithmic case,

$$U_{\log}(q) = \sum_{1\le i<j\le 3} m_i m_j \log\left(\frac{1}{|q_i - q_j|}\right). \tag{4.5}$$

The corresponding equations of motion are given by

$$m_i \ddot{q}_i = \frac{\partial U}{\partial q_i}, \tag{4.6}$$

and the periodic solutions are the critical points of the Lagrangian action functional

$$\mathcal{A}(\gamma) = \int_0^{2\pi} \left[\frac{1}{2}\sum_{i=1}^{3} m_i |\dot{q}_i|^2 + U(\gamma)\right] \mathrm{d}t. \tag{4.7}$$

Among these, the classical *Lagrangian circular orbits* correspond to configurations where the three bodies form an equilateral triangle that rotates uniformly around its barycentre. The stability of such periodic motions depends on the homogeneity parameter $\alpha$ and on the mass ratio parameter

$$\beta := \frac{27(m_1m_2 + m_2m_3 + m_1m_3)}{(m_1 + m_2 + m_3)^2} \in (0, 9]. \tag{4.8}$$

Barutello *et al.* derived the full stability diagram in the $(\alpha, \beta)$-plane, identifying regions of spectral instability (SI), spectral stability (SS), and linear stability (LS). They showed that the transition between stability and instability occurs along the critical curve

$$\beta = 9\,\frac{(\alpha-2)^2}{(\alpha+2)^2}. \tag{4.9}$$

The authors computed the Morse index of the circular solution $\gamma_{\alpha,\beta}$ and its iterations. For the Kepler-type problem ($\beta = 0$), they proved that

$$i_{\mathrm{Morse}}(\gamma_{\alpha,0}) = \begin{cases} 0, & \alpha \in [1,2), \\ 2, & \alpha \in [0,1), \end{cases} \tag{4.10}$$

and that for fixed $\alpha$ the index of the $k$-th iteration, $i_{\mathrm{Morse}}(\gamma_{\alpha,0}^k)$, diverges to $+\infty$ as $k \to \infty$. For the full three-body problem with $\beta > 0$, they showed that the Morse index is a non-increasing function

of $\beta$ and that for equal masses ($\beta = 9$) and $\alpha \in (1, 2)$ the circular solution is a strict local minimiser, while for $\alpha \in [0, 1)$ it becomes a saddle point.

**Numerical validation.** Applying our discrete framework (Definition 4.5) to a discretised version of the Lagrangian circular orbit, we obtain a discrete Hessian matrix whose negative eigenvalues yield the *discrete Morse index* $\tilde{n}^-$. When computed using the discretisation through points as in Equation (4.3), our results reproduce the same qualitative and quantitative behaviour as those obtained in [99]. In particular:

- For $\alpha \in [1, 2)$ the discrete Morse index is $\tilde{n}^- = 0$, confirming that the orbit is a local minimum of the discrete action functional.
- For $\alpha \in [0, 1)$ the discrete Morse index is $\tilde{n}^- = 2$, indicating a saddle-type configuration.
- Increasing the parameter $\beta$ (i.e., approaching equal masses) leads to a monotone decrease in the discrete Morse index, in agreement with the analytical stability regions shown in Figure 2(b) of [99].

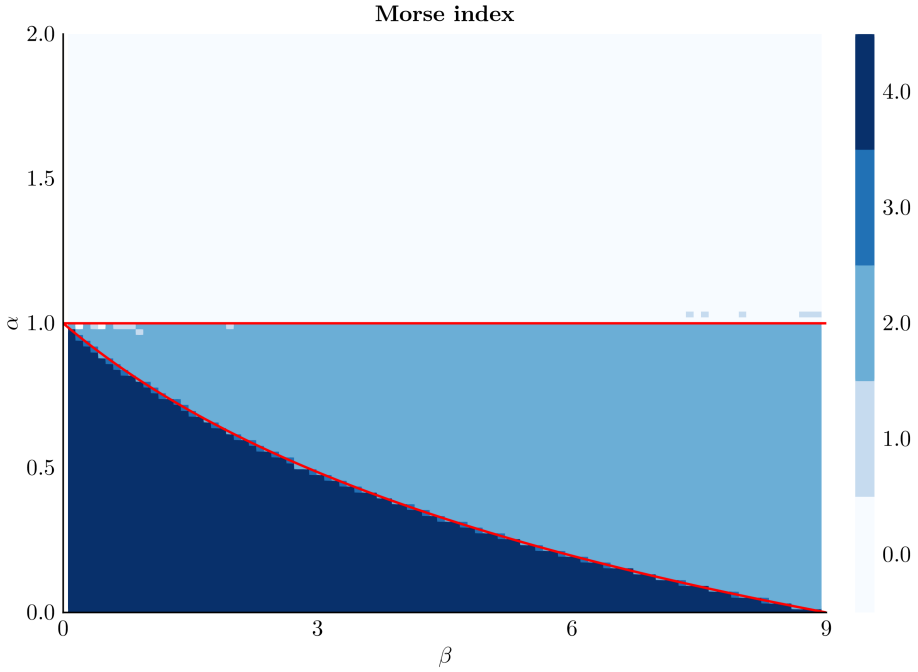



Figure 4.1: Discrete Morse index computed as Algorithm 13 across $(\alpha, \beta)$, matching the qualitative regions of [99].

As illustrated in Fig. 4.1, our discrete index reproduces the same transitions reported in [99], while Fig. 4.2 displays representative eigenvalue distributions for selected $(\alpha, \beta)$ pairs. An intelligent discretisation strategy was adopted, since setting a fixed tolerance is nontrivial; as a matter of fact, different regions of $(\alpha, \beta)$ require varying numerical precision. In practice, an orbit was first reconstructed with a given number of time steps, from which the corresponding Fourier coefficients were obtained. The discrete Morse index was then evaluated twice, first with 200 steps and then with 800 steps, to monitor the consistency of the computed spectrum. Only eigenvalues exhibiting significant variation between the two resolutions were considered potentially spurious. This procedure ensures that the reported Morse index values are robust across the parameter range.
Although our variational problem and the specific family of periodic orbits differ from those considered by Barutello *et al.*, the numerical agreement between the computed discrete Morse indices and their theoretical predictions provides a robust validation of our computational approach. This

consistency confirms that the discrete Morse index, evaluated from the Hessian of the discretised action functional, faithfully reproduces the qualitative stability information encoded in the classical (continuous) Morse index. Hence, the proposed discretisation scheme not only bridges the analytical and numerical perspectives but also aligns with established Morse-theoretic results in celestial mechanics.

## 4.2 Intra and trans level distances

In this Section, we categorise the minimum points found in Chapter 3 based on the value of their objective function, which in our case is the action functional, following the approach in [58,106,107]. We refer to these categories as *levels*. Each level groups local minimum points with very similar action values, providing a structured way to analyse their distribution.
For each level, we define the *intra-level* distance $\rho_{\mathrm{IL}}$ as the average of the relative distances between each local minimum and all other minimum points within the same level. Additionally, the *trans-level* distance $\rho_{\mathrm{TL}_k}$ quantifies the average relative distance between each local minimum and all other minima in the next lower level. For the lowest level, $\rho_{\mathrm{TL}_k}$ is computed as the average distance from the best-known solution.
We now provide a more precise definition of these concepts.

**Definition 4.6** (Intra and trans level distances, Definition 4.2 [92])**.** Let $\mathcal{M} = \{x_k\}_{k=1}^{M}$, with $M < +\infty$, be the set of possible local minimum points found during the optimisation step. For each $k$, we consider the value of the objective function in each minimum point $f_k = f(x_k)$; then, let $\varepsilon > 0$ be a suitable small number. We build an interval $I_k = [f_k - \varepsilon, f_k + \varepsilon]$. For each interval $I_k$, we have the following set:

$$\mathfrak{m}_{I_k} := \{x_i \in \mathcal{M} \mid f(x_i) \in I_k\}.$$

We define the *intra-level* distance $\rho_{\mathrm{IL}}$ of a minimum point $x_i$ as

$$\rho_{\mathrm{IL}}(x_i) := \frac{1}{|\mathfrak{m}_{I_i}|} \sum_{\substack{x_j \in \mathfrak{m}_{I_i} \\ x_j \neq x_i}} \|x_i - x_j\| \text{ with } x_i \in \mathfrak{m}_{I_i}. \tag{4.11}$$

Then, we order the set $\{f_k\}_k$ such that $f_{k-1} < f_k < f_{k+1}$, and define the *trans-level* distance $\rho_{\mathrm{TL}_k}$ of a minimum point $x_i$ as

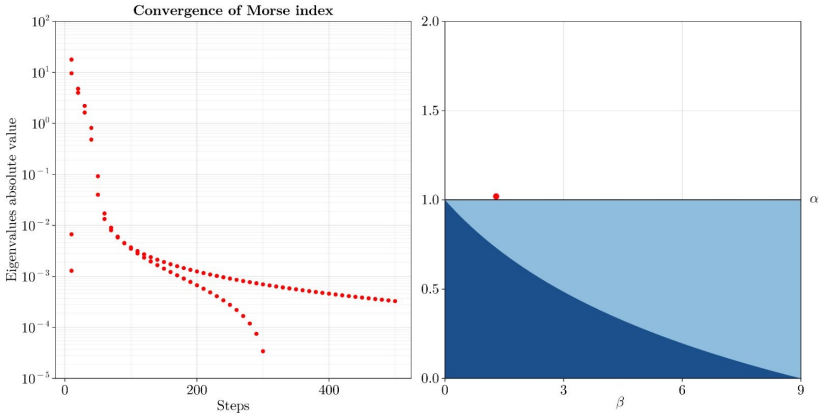
$$\rho_{\mathrm{TL}_k}(x_i) := \begin{cases} \dfrac{1}{|\mathfrak{m}_{I_k}|} \displaystyle\sum_{x_j \in \mathfrak{m}_{I_k}} \|x_i - x_j\|, & x_i \in \mathfrak{m}_{I_{k-1}} \text{ for } k > 1, \\ \dfrac{1}{|\mathfrak{m}_{I_1}|} \displaystyle\sum_{x_j \in \mathfrak{m}_{I_1}} \|x_1 - x_j\|, & x_1 \text{ is best-known sol.} \end{cases} \tag{4.12}$$

For $k = 1$, in our case, we consider the best-known solution as the one with the lowest norm of the gradient of the action functional.

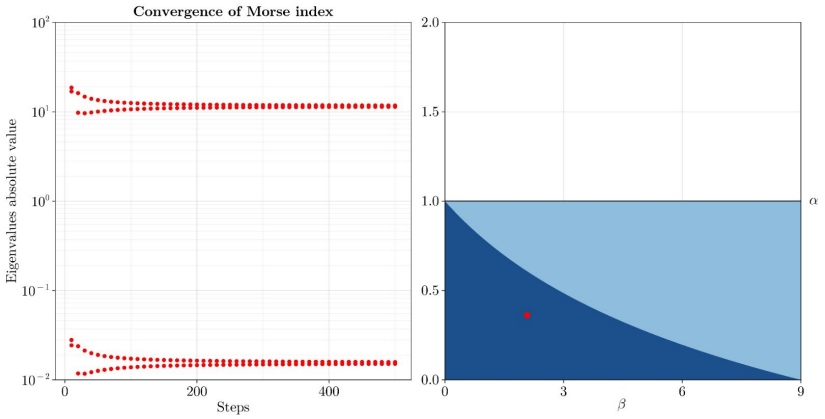
The values of $\rho_{\mathrm{IL}}$ and $\rho_{\mathrm{TL}_k}$ provide a clear indication of the diversity among local minima and the likelihood of transitions between neighbouring levels. In particular, small values of both $\rho_{\mathrm{IL}}$ and $\rho_{\mathrm{TL}_k}$ typically occur in the presence of a *funnel structure*, where minimum points are progressively closer both within and across levels as the objective function decreases. Conversely, if $\rho_{\mathrm{IL}}$ does not tend to zero, or if clusters appear with significantly different values of $\rho_{\mathrm{TL}_k}$, this may indicate the existence of distinct *multi-funnel* regions within the landscape, suggesting the coexistence of several basins of attraction separated by higher barriers. For more details, see [58, 90, 106, 107].

### 4.2.1 Computation of intra and trans level distances

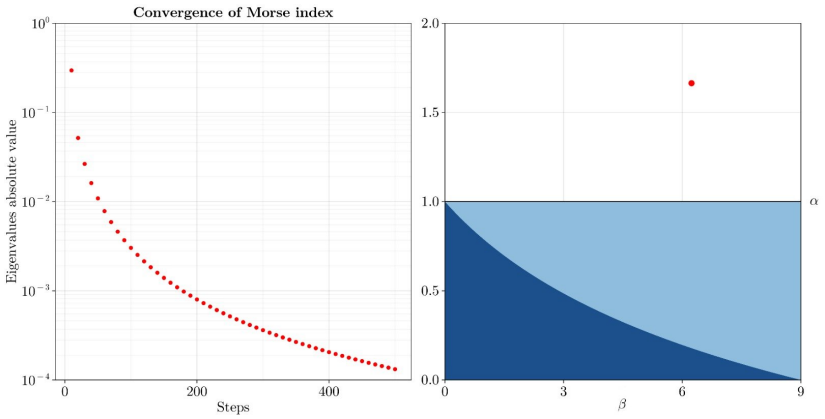
The definitions in Section 4.2 provide a theoretical framework to quantify the diversity and structural organisation of local minimum points. In practice, their computation requires two main



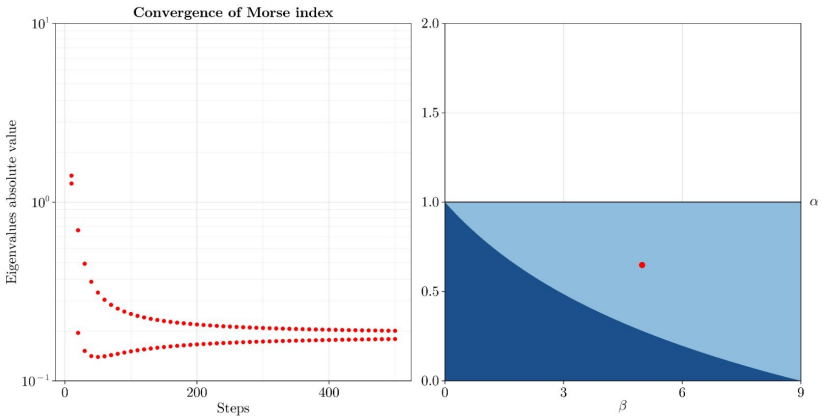

(a) $\beta = 1.29,\ \alpha = 1.01$ (on the border). One eigenvalue $\to 0$; the other remains uncertain.




(b) $\beta = 2.07,\ \alpha = 0.36$. Four eigenvalues clearly identifiable.




(c) $\beta = 6.23,\ \alpha = 1.66$. Dominant eigenvalue visibly tends to 0.




(d) $\beta = 4.98,\ \alpha = 0.64$. Two eigenvalues clearly visible.

Figure 4.2: Eigenvalue behaviour across representative $(\alpha, \beta)$ pairs complementing Fig. 4.1: (a) border case with one eigenvalue $\to 0$; (b) four eigenvalues; (c) single eigenvalue $\to 0$; (d) two eigenvalues.

ingredients: (i) clustering the minimum points into *levels* according to their action values, and (ii) computing averages of pairwise distances either within a cluster (intra-level) or between adjacent clusters (trans-level). We now describe the procedure in detail.

*Step 1: Partition into levels.* We group minimum points into levels by clustering their action values within a tolerance $\varepsilon$. Each minimum $x_i$ is assigned to a cluster $\mathfrak{m}_{I_\ell}$ around a representative value $L_\ell$, and points closer than a geometric tolerance $\tau$ are treated as duplicates and discarded.

*Step 2: Intra-level distance.* For each $x_i \in \mathfrak{m}_{I_\ell}$, we compute $\rho_{\mathrm{IL}}(x_i)$ as the mean Euclidean distance to all other distinct minimum points in the same cluster. If the cluster contains only one element, $\rho_{\mathrm{IL}}(x_i) = 0$.

*Step 3: Trans-level distance.* We order the levels by increasing action. For $x_i \in \mathfrak{m}_{I_\ell}$ with $\ell > 1$, the trans-level distance $\rho_{\mathrm{TL}}(x_i)$ is the mean distance to minimum points in the immediately lower level $\mathfrak{m}_{I_{\ell-1}}$. For $\ell = 1$, we measure the distance to the best-known solution $x^\star$ (or average over a small set of best candidates). This procedure is summarised in Algorithm 14.

**Algorithm 14** Intra- and trans-level distances

**Input:** Set of minimum points $\mathcal{M}$, action $f$, action tolerance $\varepsilon$, geometric tolerance $\tau$.
**Output:** Levels $\{\mathfrak{m}_{I_\ell}\}$, intra-level distances $\rho_{\mathrm{IL}}$, trans-level distances $\rho_{\mathrm{TL}}$.
Compute actions $f_k \leftarrow f(x_k)$ for all $x_k \in \mathcal{M}$.
Build levels $\{L_\ell\}$ by scanning sorted $\{f_k\}$ and starting a new one when $|f_k - L_\ell| > \varepsilon$.
Initialise $\mathfrak{m}_{I_\ell} \leftarrow [\,]$ for each $\ell$.
**for** each $x \in \mathcal{M}$ **do**
    Find $\ell$ such that $|f(x) - L_\ell| \le \varepsilon$.
    **if** no such $\ell$ **then**
        **continue**
    **end if**
    **if** $\exists\, y \in \mathfrak{m}_{I_\ell}$ with $\|x - y\| < \tau$ **then**
        **continue**
    **end if**
    Append $x$ to $\mathfrak{m}_{I_\ell}$.
**end for**
**for** $\ell = 1$ to $K$ **do**
    **for** each $x_i \in \mathfrak{m}_{I_\ell}$ **do**
        Compute $\rho_{\mathrm{IL}}(x_i)$ as average distance to all other elements in $\mathfrak{m}_{I_\ell}$.
        **if** $\ell > 1$ and $\mathfrak{m}_{I_{\ell-1}}$ non-empty **then**
            $\rho_{\mathrm{TL}}(x_i) \leftarrow$ average distance to $\mathfrak{m}_{I_{\ell-1}}$.
        **else if** $\ell = 1$ **then**
            $\rho_{\mathrm{TL}}(x_i) \leftarrow \|x_i - x^\star\|$.
        **else**
            $\rho_{\mathrm{TL}}(x_i) \leftarrow$ NaN.
        **end if**
    **end for**
**end for**
**return** $\{\mathfrak{m}_{I_\ell}\}, \{\rho_{\mathrm{IL}}(x_i)\}, \{\rho_{\mathrm{TL}}(x_i)\}$.

## 4.3 Winding number

In celestial mechanics and Hamiltonian dynamics, variants of the winding number and related topological indices are widely employed to classify periodic orbits and bifurcations. For instance, Moreno [108] discusses a *symplectic winding index* (the Conley-Zehnder index) associated with non-degenerate periodic orbits in the restricted three-body problem, which characterises how families of

periodic orbits connect under continuation. Montgomery [109] introduced a variational framework for the $n$-body problem where periodic orbits are constrained by an integer winding number: for elliptic Keplerian trajectories the winding number is $\pm 1$, while higher winding numbers correspond to collision orbits. Similarly, in the context of choreographic solutions of the $n$-body problem, Montaldi and Steckles [41] employ the notion that each pair of bodies winds around one another an integer number of times, and also consider the winding of the entire choreography around the origin as a classification parameter. Topological methods have also been used for the classification of periodic orbits in irregular gravitational potentials [110], where orbits are distinguished by how they wrap around singularities or forbidden regions. These works demonstrate that winding-based invariants are natural tools for identifying qualitative differences among periodic orbits, and motivate the numerical computation of the planar winding number presented in this Section.
More recently, Montgomery, in his book on the open problems of the $n$-body problem [17], highlights the so-called *braid problem*, which seeks to understand how the trajectories of bodies intertwine in space-time. In this perspective, the study of winding numbers can be viewed as a first step towards addressing that broader question: by introducing a topological descriptor in the spatial plane, one may eventually extend the analysis to include the temporal dimension and investigate whether the resulting orbits form braids in space-time.

**Definition** (Winding number: complex formulation)**.** Let $\gamma$ be a closed curve in $\mathbb{C}$, and let $a \in \mathbb{C}$ be a point not lying on $\gamma$. The winding number of $\gamma$ around $a$ is defined by the contour integral

$$n(\gamma, a) \;=\; \frac{1}{2\pi i} \oint_{\gamma} \frac{d\zeta}{\zeta - a},$$

where $\zeta$ denotes the complex coordinate along $\gamma$. This integral counts, with orientation, the number of times the curve winds around $a$.

**Definition 4.7** (Winding number: real formulation)**.** If the curve is parametrised in real coordinates as

$$\gamma(t) = (x(t), y(t)), \qquad t \in [t_0, t_1], \quad \gamma(t_0) = \gamma(t_1),$$

and $a = (a_x, a_y) \in \mathbb{R}^2$, then the winding number can equivalently be expressed as

$$n(\gamma, a) \;=\; \frac{1}{2\pi} \oint_{\gamma} \frac{(x(t) - a_x)\, dy(t) - (y(t) - a_y)\, dx(t)}{(x(t) - a_x)^2 + (y(t) - a_y)^2}.$$

This formula represents the total angular variation of the vector $\gamma(t) - a$ over one traversal of the curve, normalised by $2\pi$. In particular, $n(\gamma, a)$ is always an integer when the curve does not pass through $a$.

The winding number is positive if the curve winds around $a$ counterclockwise, negative if clockwise, and zero if $a$ lies outside the enclosed region. In our computations we adopt the real-coordinate formulation, since it can be directly implemented from a parametric representation $\gamma(t)$ and its derivatives, as in the Fourier series expansion used in our numerical framework.

### 4.3.1 Computation of the winding number

To compute the winding number numerically, we exploit the Fourier representation of the periodic orbits obtained in the optimisation step. Each orbit $\Gamma(t) = (x(t), y(t))$ with period $T$ is expressed as a truncated Fourier series, from which both the coordinates and their derivatives can be evaluated efficiently.
*Step 1: Fourier reconstruction.* Given Fourier coefficients $\{A_0^{(d)}, A_k^{(d)}, B_k^{(d)}\}$ for $d = 1, 2$ and modes $k = 1, \ldots, F$, the coordinates of the curve are

$$x(t) \;=\; A_0^{(1)} \;+\; \sum_{k=1}^{F} \Big( A_k^{(1)} \cos\tfrac{2\pi k t}{T} + B_k^{(1)} \sin\tfrac{2\pi k t}{T} \Big),$$

$$y(t) \;=\; A_0^{(2)} \;+\; \sum_{k=1}^{F} \Big( A_k^{(2)} \cos\tfrac{2\pi k t}{T} + B_k^{(2)} \sin\tfrac{2\pi k t}{T} \Big).$$

Their derivatives $\dot{x}(t), \dot{y}(t)$ are obtained by differentiating term by term.
*Step 2: Angular velocity integrand.* For a reference point $p = (p_1, p_2)$, the instantaneous angular velocity of $\Gamma(t)$ with respect to $p$ is

$$\omega(t) \;=\; \frac{1}{2\pi} \, \frac{(x(t) - p_1)\, \dot{y}(t) - (y(t) - p_2)\, \dot{x}(t)}{(x(t) - p_1)^2 + (y(t) - p_2)^2}. \tag{4.13}$$

*Step 3: Numerical integration and rounding.* The winding number is obtained by integrating $\omega(t)$ over one full period, using Definition (4.7):

$$w \;=\; \int_0^T \omega(t)\, dt.$$

In exact arithmetic $w$ is an integer. Numerically, the result is rounded to the nearest integer, and a tolerance check ensures $|w - w_{\text{num}}| < \text{tol}$, otherwise an error is raised.
The full procedure is summarised below in Algorithm 15.

**Algorithm 15** Computation of the winding number

**Input:** Periodic orbit $\Gamma$ in Fourier form, period $T$, number of modes $F$, reference point $p = (p_1, p_2)$, tolerance tol.
**Output:** Winding number $w \in \mathbb{Z}$.
Reconstruct $x(t), y(t)$ and $\dot{x}(t), \dot{y}(t)$ from Fourier coefficients.
Define integrand $\omega(t)$ as in Equation (4.13).
Evaluate $w_{\text{num}} = \int_0^T \omega(t)\, dt$ via numerical quadrature.
Round $w_{\text{num}}$ to nearest integer $w$.
**if** $|w - w_{\text{num}}| > \text{tol}$ **then**
    **Error:** non-integer winding number (curve crosses $p$ or large numerical error).
**end if**
**return** $w$.

*Remark.* We employ Gauss-Kronrod quadrature (`quadgk`) for the integral, as it provides reliable accuracy with adaptive refinement. The method is restricted to planar curves (dim = 2), since the winding number is not defined for higher dimensions. The tolerance tol ensures robustness against small numerical deviations from integrality.

*Remark.* The reference point $p = (p_1, p_2)$ can in principle be chosen arbitrarily, provided it does not lie on the curve. Unless otherwise specified, we take $p$ to be the centre of mass of the orbit, which provides a natural and consistent choice.

# Chapter 5

# Mountain Pass solutions

In this Chapter we turn to the detection of non-minimal critical points of the action functional by means of the Ambrosetti–Rabinowitz Mountain Pass framework. After recalling the Mountain Pass Theorem and the associated steepest–descent flow (see, e.g., [32, 111]), we follow the constructive approach developed in [59] and present an enhanced, numerically oriented variant as in [92,104]. The first part of the Chapter, more precisely Section 5.1, introduces the dynamical objects underlying the construction of positively invariant sets, $\omega$-limit sets, disconnected sublevels and their basins of attraction and states the key results that guarantee the existence of Palais–Smale sequences on the boundary between basins, which in turn lead to mountain-pass critical points under suitable compactness assumptions. Building on this theory, we then describe an improved implementation of Barutello and Terracini's procedure via the auxiliary routines Gradient_descent, Bisection, and Evolve, and we formulate the resulting *Mountain Pass Algorithm* (Algorithm 17), designed not only to approximate a mountain-pass critical point but also to record additional critical points encountered along the iterations. This capability is particularly important in the highly nonconvex and symmetric landscape of the $n$-body problem, where multiple distinct critical points may coexist and be encountered along the same variational construction.
Finally, in Section 5.2, we specialise the method to the symmetric $n$-body problem by introducing a regularised action functional satisfying the Palais–Smale condition (Proposition 5.8). We also examine the role of symmetry, distinguishing between isolated minima and non-isolated families of critical points (type S groups). While a full analytical treatment of the latter is mathematically delicate and lies beyond the scope of the present work, we indicate how their presence affects the behaviour of the algorithm and discuss practical diagnostics based on the values of the regularised action functional.

## 5.1 Mountain Pass Theorem

Consider the steepest descent flow associated with $f$, denoted by $\eta_f : \mathbb{R}_+ \times X \to X$, defined as the solution of the Cauchy problem:

$$\begin{cases} \dfrac{d}{dt}\eta_f(t,x) = -\dfrac{\nabla f(\eta_f(t,x))}{1+\|\nabla f(\eta_f(t,x))\|}, \\ \eta_f(0,x) = x. \end{cases} \tag{5.1}$$

A subset $X_0 \subset X$ is called *positively invariant* under the flow $\eta_f$ if $\eta_f(t,x_0) \in X_0$ for all $t \geq 0$ and every $x_0 \in X_0$. For each $x \in X$, we define the *$\omega$-limit set* of $x$ with respect to $\eta_f$ as

$$\omega_x := \left\{ \lim_{t_n \to +\infty} \eta_f(t_n, x) : (t_n)_n \subset \mathbb{R}_+ \right\}, \tag{5.2}$$

which is a closed and positively invariant subset of $X$.

Let $c \in \mathbb{R}$ be a value such that the sublevel set $f^c$ is not connected. We can write it as the disjoint union of its connected components $(F_i^c)_i$:

$$f^c = \bigcup_i F_i^c, \qquad F_i^c \cap F_j^c = \varnothing \quad \text{for } i \neq j.$$

For each index $i$, the *basin of attraction* of $F_i^c$ is defined as

$$\mathcal{F}_i^c := \{x \in X : \omega_x \subset F_i^c\}.$$

**Definition 5.1** (Path, Definition 5.1 [92])**.** A *path* is a continuous map $\gamma : [0,1] \to X$. For two distinct points $x_1, x_2 \in X$, we denote by

$$\Gamma_{x_1,x_2} := \{\gamma \in \mathcal{C}([0,1], X) : \gamma(0) = x_1,\ \gamma(1) = x_2\}$$

the set of all paths connecting $x_1$ and $x_2$.

The following can be established:

**Theorem 5.2** (Theorem 1.4, [59])**.** *Let $f^c$ be a disconnected sublevel for the functional $f$. Let $F_i^c$ be the disjoint connected components of $f^c$ and $\mathcal{F}_i^c$ their basins of attraction. For $x_i \in F_i^c, i = 1, 2$ and $\gamma \in \Gamma_{x_1,x_2}$, there exists $\bar{x} \in \gamma([0,1]) \cap \partial\mathcal{F}_1^c$.*

**Corollary 5.3** (Corollary 1.5, [59])**.** *Under the same conditions as Theorem 5.2, let $\bar{x} \in \gamma([0,1]) \cap \partial\mathcal{F}_1^c$. Then $f(\omega_{\bar{x}}) \geq c$ and there exists a sequence $(x_n)_n = \eta_f(t_n, \bar{x}) \subset \partial\mathcal{F}_1^c$ such that*

$$\lim_{n\to+\infty} \nabla f(x_n) = 0 \quad \textit{and} \quad \lim_{n\to+\infty} f(x_n) = f(\omega_x).$$

**Corollary 5.4** (Corollary 1.6, [59])**.** *Under the same conditions as Theorem 5.2, let $\bar{x} \in \gamma([0,1]) \cap \partial\mathcal{F}_1^c$. Then there exists a sequence $(\tilde{y}_n)_n \subset X$, $(\tilde{y}_n)_n := \eta_f(\tilde{T}_n, x_1^n)$ such that*

$$\lim_{n\to+\infty} \nabla f(\tilde{y}_n) = 0 \quad \textit{and} \quad c \leq f(\tilde{y}_n) \leq f(\bar{x}),\ \forall n \in \mathbb{N}.$$

In [59], Corollary 5.4 is established by constructing a sequence $(\tilde{y}_n)_n$ through an iterative process known as *Barutello and Terracini's Algorithm*. The convergence of this sequence is guaranteed under certain additional compactness assumptions imposed on the functional $f$.
Within this framework, we recall the following key concept.

**Definition 5.5** (Palais–Smale condition, Definition 7 [59])**.** Let $f : X \to \mathbb{R}$ be a differentiable functional. A sequence $(x_m)_m \subset X$ is said to be a *Palais–Smale sequence for $f$ in the interval $[a,b]$* if

$$a \leq f(x_m) \leq b, \quad \forall m \in \mathbb{N}, \qquad \text{and} \qquad \nabla f(x_m) \xrightarrow[m\to+\infty]{} 0.$$

The functional $f$ is said to satisfy the *Palais–Smale condition in $[a,b]$* if every such sequence admits a convergent subsequence $(x_{m_k})_k \to x_0 \in X$.
Likewise, $(x_m)_m \subset X$ is called a *Palais–Smale sequence for $f$ at level $c$* if

$$f(x_m) \xrightarrow[m\to+\infty]{} c \qquad \text{and} \qquad \nabla f(x_m) \xrightarrow[m\to+\infty]{} 0.$$

We say that $f$ satisfies the *Palais–Smale condition at level $c$*, denoted by $(\mathrm{PS})_c$, if every Palais–Smale sequence at level $c$ possesses a convergent subsequence.

*Remark.* Corollary 5.3 ensures the existence of a Palais–Smale sequence at level $f(\omega_{\bar{x}})$ for the functional $f$. If $f$ satisfies the condition $(\mathrm{PS})_{f(\omega_{\bar{x}})}$, then there exists a critical point $\bar{\bar{x}} \in X$ such that $f(\bar{\bar{x}}) = f(\omega_{\bar{x}})$. Furthermore, Corollary 5.4 provides the existence of a Palais–Smale sequence for $f$ within the interval $[c, f(\bar{x})]$. When the functional $f$ meets the Palais–Smale condition on this

interval, the convergence of the sequence $(\tilde{y}_n)_n$ generated by Barutello and Terracini's Algorithm is ensured.

**Enhanced version of Barutello and Terracini's Algorithm.** Following [92, 104], we now propose an improved formulation of Barutello and Terracini's Algorithm. To this end, we introduce three auxiliary procedures: the GRADIENT_DESCENT function, the BISECTION function, and the EVOLVE function. Their definitions are provided in Algorithm 16, where $P$ is considered a structure containing the objective functional $f$, its gradient $\nabla f$, the steepest descent flow $\eta_f$ (as introduced in Equation (5.1)), the list $L$ of known critical points, and the element $x_{\min}$ defined in Algorithm 17. The parameter `to_min` determines whether the stopping criterion of the GRADIENT_DESCENT function is satisfied when a minimum is reached or when the norm of $\nabla f$ begins to increase.

**Algorithm 16** Bisection function (Algorithm 5 [92])

**function** GRADIENT_DESCENT($x, P$, to_min = true)
  $x_{\text{new}} \leftarrow x, \quad x_{\text{old}} \leftarrow x, \quad dt \leftarrow 1 \times 10^{-3}$
  $\text{grad}_{\text{new}} \leftarrow \nabla f(x_{\text{new}}), \quad \text{grad}_{\text{old}} \leftarrow \nabla f(x_{\text{old}})$
  **while** (to_min $\wedge \|\text{grad}_{\text{new}}\| >$ g_tol) $\vee$ ($\neg$to_min $\wedge \|\text{grad}_{\text{new}}\| \leq \|\text{grad}_{\text{old}}\|$) **do**
    $x_{\text{old}} \leftarrow x_{\text{new}}$
    $x_{\text{new}} \leftarrow$ integrate $\eta_f$ starting from $x_{\text{old}}$ for one step
    $\text{grad}_{\text{old}} \leftarrow \text{grad}_{\text{new}}$
    $\text{grad}_{\text{new}} \leftarrow \nabla f(x_{\text{new}})$
    **if** $\neg$to_min $\vee$ ($x_{\text{new}}$ is known) **then**
      **return** $x$
    **end if**
  **end while**
  **return** $x_{\text{new}}$
**end function**
**function** BISECTION($x_0, x_1, P$)
  distance $\leftarrow \|x_0 - x_1\|$
  $x \leftarrow \frac{x_0 + x_1}{2}$
  $x_{\text{new}} \leftarrow$ GRADIENT_DESCENT($x, P$, to_min = true)
  **if** $x_{\text{new}}$ is a unknown critical point **then**
    Add $x_{\text{new}}$ to the list of critical points
    **return** $x_0, x$
  **end if**
  **if** $\|f(x_{\text{new}}) - f(P.x_{\min})\| <$ min_step **then**
    **return** $x, x_1$
  **else**
    **return** $x_0, x$
  **end if**
**end function**
**function** EVOLVE($x, P$)
  GRADIENT_DESCENT($x, P$, to_min = false)
**end function**

We now introduce an alternative approach to Barutello and Terracini's Algorithm, namely the *Mountain Pass Algorithm*, presented in Algorithm 17. This procedure takes as input two initial points $x_0$ and $x_1$, together with a functional $f$ and its gradient $\nabla f$, and aims to locate a critical point of $f$ that is not a local minimum. A detailed description follows.

**Input.** Define tolerance parameters `g_tol` and `min_step` to regulate the stopping criteria. Specify the initial points $x_0, x_1$, the function $f$, and its gradient $\nabla f$. The point $x_0$ should correspond to

a local minimum of $f$, while $x_1$ must lie outside the basin of attraction of $x_0$ under the steepest descent flow $\eta_f$ defined in Equation (5.1). Hence, $x_1$ should either represent another local minimum or satisfy $f(x_0) > f(x_1)$.

**Step 1.** Initialise the algorithm by setting up the problem structure $P$ and identifying the current minimum point.

**Step 2.** Iteratively apply

$$(x_0, x_1) \leftarrow \textsc{Bisection}(x_0, x_1, P)$$

to progressively approach the stable manifold associated with the desired non-minimal critical point. The iteration stops once $\|x_0 - x_1\| < \texttt{min\_step}$.

*Remark* 5.6. It is worth noting that the Bisection function records any newly discovered critical points during its execution. In particular, whenever the Gradient_Descent routine yields a new point $x_{\text{new}}$, it verifies whether this point has been previously identified or represents a distinct critical point.

**Step 3.** Evolve both $x_0$ and $x_1$ along the stable manifold until the norm of the gradient $\nabla f$ at one of these points begins to increase.

Repeat **Steps 2** and **3** until the convergence criterion

$$\left\|\nabla f\left(\tfrac{x_0+x_1}{2}\right)\right\| < \texttt{g\_tol}$$

is satisfied.

**Step 4.** Compute the midpoint $x = \frac{x_0+x_1}{2}$ and return it as the identified non-minimal critical point of $f$.

**Algorithm 17** Mountain Pass algorithm (Algorithm 6 [92])

**Input:** Set the tolerances g_tol and min_step and define the initial points $x_0, x_1$, the function $f$ and its gradient $\nabla f$.
**Output:** Non-minimal point of the function $f$.
**Step 1:** Initialise variable: $x_{\min} \leftarrow x_0$;
Initialise problem $P$: $P \leftarrow \text{Problem}(f, \nabla f, x_{\min}, L = \{x_0, x_1\})$.
**Steps 2. and 3.**
**while** $\|\nabla f\left(\frac{x_0+x_1}{2}\right)\| >$ g_tol **do**
    $(x_0, x_1) \leftarrow \textsc{bisection}(x_0, x_1, P)$
    **if** $x_0$ or $x_1$ is known **then**
        **continue**
    **end if**
    **if** $\|x_1 - x_0\| >$ min_step **then**
        **continue**
    **end if**
    $x_0 \leftarrow \textsc{evolve}(x_0, P)$
    $x_1 \leftarrow \textsc{evolve}(x_1, P)$
**end while**
**Step 4.** Compute and return midpoint: $x \leftarrow \frac{x_0+x_1}{2}$
**return** $x$

Algorithm 17 not only retains the fundamental properties of Barutello and Terracini's Algorithm, thereby allowing it to serve as a constructive proof of Theorem 5.2; but, as emphasized in Remark 5.6, it is also specifically structured to detect multiple critical points of the functional whenever possible. We can now state the Mountain Pass Theorem:

**Theorem 5.7** (Mountain Pass Theorem, Theorem 2.1 [59])**.** *Let $f$ be a $\mathcal{C}^2$ functional on a Hilbert*

*space $X$. Let $x_1, x_2 \in X$ let $\Gamma_{x_1,x_2}$ be the set of paths defined in* (5.1) *and $c_0$ the level*

$$c_0 =: \inf_{\gamma \in \Gamma_{x_1,x_2}} \sup_{s \in [0,1]} f(\gamma(s)), \tag{5.3}$$

*such that*

$$c_0 > \max\{f(x_1), f(x_2)\}. \tag{5.4}$$

*If the functional $f$ satisfies the Palais-Smale condition at level $c_0$, then there exists a critical point for the functional $f$ at level $c_0$, that is not a local minimiser.*

## 5.2 Mountain Pass solutions for the symmetrical $n$-body problem

Our next objective is to employ Theorem 5.7 in the context of the action functional corresponding to the $n$-body problem. By doing so, we aim to demonstrate that this theorem ensures the existence of a solution distinct from those obtained in Chapter 3. In order to apply Theorem 5.7, it is first necessary to regularise the action functional introduced in Equation (2.4).
Among the many approaches that can be used to achieve this regularisation (see, for instance [112]), we chose the following one, since it more effectively regularises collisional orbits by containing them, rather than pushing them away.

$$\begin{aligned} \mathcal{A}_{\mathbb{I}}^{\text{reg.}}(y) &:= \int_{\mathbb{I}} L_\varepsilon(y(t), \dot{y}(t))\,\mathrm{d}t, \\ &= \int_{\mathbb{I}} \left(K(\dot{y}(t)) + U_\varepsilon(y(t))\right)\,\mathrm{d}t, \\ &= \int_{\mathbb{I}} \left( \frac{1}{2} \sum_{i=1}^{n} m_i \|\dot{y}(t)\|^2 + \sum_{i<j} \frac{m_i m_j}{\sqrt{\varepsilon^2 + \|y_i(t) - y_j(t)\|^2}} \right) \mathrm{d}t. \end{aligned} \tag{5.5}$$

The regularised action functional $\mathcal{A}_{\mathbb{I}}^{\text{reg.}}$ is smooth, specifically $\mathcal{C}^2(Y)$.
We now state the following proposition:

**Proposition 5.8** (Proposition 5.8 [92])**.** *The regularised action functional defined in* (5.5) *satisfies the Palais-Smale condition at every level $c > 0$.*

*Proof.* Let $(y_\nu)_\nu \subset Y$ be such that

$$\mathcal{A}_{\mathbb{I}}^{\text{reg.}}(y_\nu) \xrightarrow{\nu \to +\infty} c \quad \text{and} \quad \nabla \mathcal{A}_{\mathbb{I}}^{\text{reg.}}(y_\nu) \xrightarrow{\nu \to +\infty} 0.$$

Our aim is to find an element $\bar{y} \in Y$ such that $y_\nu \longrightarrow \bar{y}$ as $\nu \longrightarrow +\infty$ in $Y$.
As coercivity does not depend on the chosen potential $U$ and $\mathcal{A}_{\mathbb{I}}$ is coercive, we have that also $\mathcal{A}_{\mathbb{I}}^{\text{reg.}}$ is coercive. It follows that $(y_\nu)_\nu$ is bounded in $Y$, therefore, up to subsequences, it weakly converges to $\bar{y}$. Moreover, it also follows that we can write $\nabla \mathcal{A}_{\mathbb{I}}^{\text{reg.}}$ as $(\mathrm{Id} + K)$, with $K$ a compact operator; hence:

$$\begin{aligned} \nabla \mathcal{A}_{\mathbb{I}}^{\text{reg.}}(y_\nu) &\xrightarrow{\nu \to +\infty} 0 \\ (\mathrm{Id} + K)(y_\nu) &\xrightarrow{\nu \to +\infty} 0 \\ y_\nu + K(y_\nu) &\xrightarrow{\nu \to +\infty} 0. \end{aligned} \tag{5.6}$$

As we have that $(y_\nu)_\nu \rightharpoonup \bar{y}$ and $K(y_\nu)_\nu$ is precompact, using relation (5.6), we have that there exists a subsequence of $(y_\nu)_\nu$, which we still call $(y_\nu)_\nu$, so that $K(y_\nu)_\nu \longrightarrow K(\bar{y})$. Using again relation (5.6), we conclude that $y_\nu \longrightarrow \bar{y}$ as $\nu \longrightarrow +\infty$ in $Y$. □

**Isolated and non-isolated minimum points.** In order to apply the Mountain Pass Theorem to the symmetric $n$-body problem, it is essential to determine whether the minimum points identified in Chapter 3 are isolated or not. To formalise this distinction, we introduce the following concept.

**Definition 5.9** (Type S, Definition 5.9 [92])**.** A symmetry group $G$ is said to be of *type S* if, for every $x \in Y^G$, there exists a one-parameter group of rotations $R_\theta$ such that $R_\theta x \in Y^G$.

*Remark.* If the group $G$ is of type S, then whenever $x$ is a minimum of the action functional defined in Equation (2.4) on $Y^G$, the rotated configuration $R_\theta x$ is also a minimum. Hence, this property entails a geometrically trivial multiplicity of solutions.

This observation implies that, whenever the property holds for a rotation group parameterised by $R_\theta$ with $\theta \in [a, b]$, no minimum point can be isolated.
In the present setting, the condition for a group to be of type S can be expressed as follows:

a. For all elements $g = (\rho, \tau, \sigma) \in \ker \tau$, one has $R_\theta^{-1} \rho R_\theta \in \ker \tau$;

b. If $\tau_g \neq \{\mathbb{1}\}$, then $R_\theta^{-1} \rho_g R_\theta = \rho_g$;

or, equivalently,

c. The normaliser $N_{SO(d)}(G)$ of $G$ in $SO(d)$ is a one-parameter subgroup of $SO(d)$.

In particular, defining $V^G = \{x \in \chi : \rho x = x\}$, the subspace $V^G$ must be invariant under the rotation $R_\theta$. Consequently, we distinguish the following two cases:

- *Not of type S:* If the symmetry group $G$ does not satisfy the type S condition, the corresponding minimum points are typically isolated. In this situation, one can directly apply Algorithm 17 to the regularised action functional $\mathcal{A}_{\mathbb{I}}^{\text{reg.}}$ introduced in Equation (5.5).
- *Type S:* When $G$ is of type S, the minimum points are not isolated, and a more refined analysis would be necessary to adapt Algorithm 17. For instance, one would need to distinguish the minimum points according to their disjoint connected components. Such an analysis, however, lies beyond the scope of the present work. For the examples discussed in Chapter 6, one may instead examine the values of the regularised action functional.

In the first case, Proposition 5.8 ensures that the regularised action functional $\mathcal{A}_{\mathbb{I}}^{\text{reg.}}$ satisfies the Palais–Smale condition for every level $c > 0$. Hence, Theorem 5.7 can be applied. By employing Algorithm 17, one can then numerically compute a solution distinct from those identified in Section 3.3.

# Chapter 6

# Examples

In this Chapter we illustrate the practical use of the theoretical and computational framework developed in the previous Chapters by presenting representative numerical experiments for the planar three-body problem with equal masses under symmetry constraints. For each symmetry group considered, periodic trajectories are first obtained via the hybrid optimisation routine of Algorithm 12; the resulting candidates are then analysed using the tools introduced in Chapter 4, namely the discrete Morse index, the intra- and trans-level distance diagnostics, while the Mountain Pass Algorithm 17 of Chapter 5 is employed to detect saddle-type solutions connecting distinct basins of attraction. The purpose of the Chapter is twofold: on the one hand, to provide concrete examples of the multiplicity of local minimum points produced by the optimisation stage and to describe their distribution across action levels; on the other hand, to demonstrate that the Mountain-Pass construction yields additional non-minimising periodic orbits that are distinguished both by their action value and by their discrete Morse index. We begin with the case $G = \mathbb{Z}_2$ in Section 6.1 and we then consider a different symmetry constraint, $G = D_6$, in Section 6.2.

## 6.1 Symmetry group $G = \mathbb{Z}_2$

In this Section, we illustrate the application of the developed theoretical and computational framework to a concrete example. We consider the planar three-body problem with equal masses and impose the symmetry group $G = \mathbb{Z}_2$. Periodic trajectories in $\mathbb{R}^2$ are obtained using Algorithm 12. Among the computed solutions are the circular orbit shown in Figure 6.1 (action value 9.802), the "Ducati" orbit in Figure 6.2 (action value 10.442), as well as the orbits depicted in Figures 6.3 and 6.4, corresponding to action levels 13.579 and 13.865, respectively.

To model this configuration, we employed the implementation provided in [43], which includes the definition of the cost function used during the optimisation process. Following the numerical setup introduced in Section 3.1, we represented each trajectory by means of 24 Fourier coefficients and assumed unit masses for all bodies.

For the genetic algorithm component, the number of populations was fixed at $N_{\mathrm{P}} = 4$, with $N_{\mathrm{A}} = 8$ agents per population. Each agent was initialised with 24 randomly generated Fourier coefficients. The MP-AIDEA framework was then used to construct the archive $\mathfrak{A}$, which records all detected local minimum points together with the population individuals at each restart.

As this configuration represents a relatively simple test case, only a modest number of generations was required in Algorithm 12. Setting NoG = 8 proved sufficient to detect four distinct periodic solutions, which were the only configurations within this setup exhibiting a small gradient norm of the action functional.

Figure 6.5 illustrates the evolution of the action values obtained throughout the optimisation process. Two dominant minima, with action levels 9.802 and 10.442, were consistently detected across successive generations. In contrast, the additional critical points with action values 13.579 and 13.865 emerged only from the sixth generation onward. These observations underline the effective-

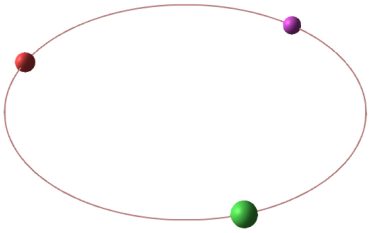


Figure 6.1: Circular orbit of action level 9.802 (Figure 3 [92])

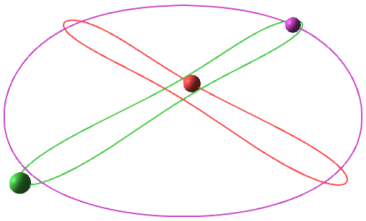


Figure 6.2: Ducati orbit of action level of 10.442 (Figure 4 [92])

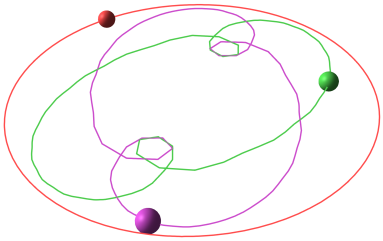


Figure 6.3: Orbit of action level 13.579 (Figure 5 [92])

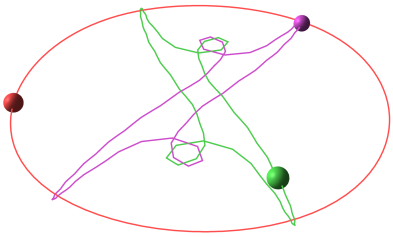


Figure 6.4: Orbit of action level of 13.865 (Figure 6 [92])

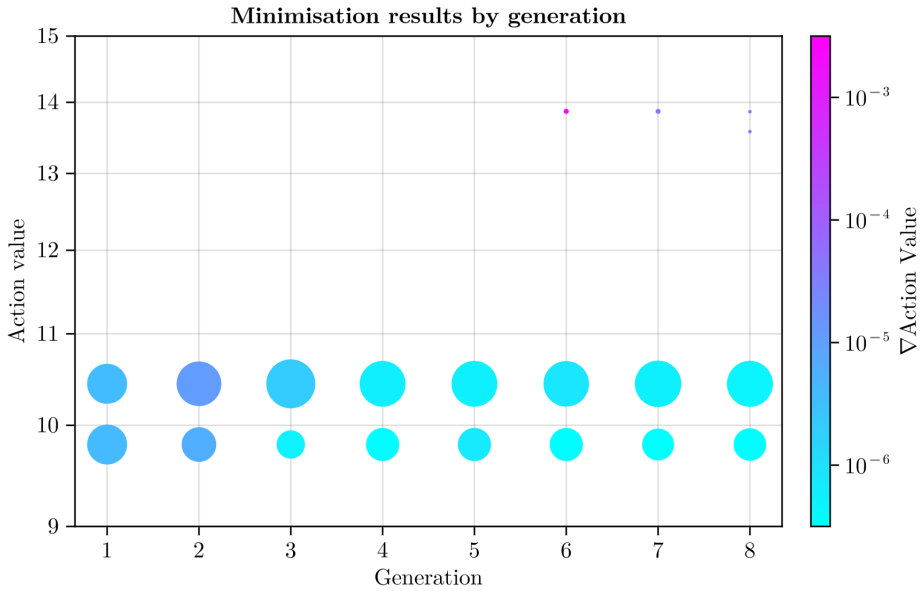



Figure 6.5: Minimisation results by generation. The plot represents the action values of identified solutions across generations. Larger dots indicate a higher number of solutions found at that action value, while the colour gradient represents the magnitude of the action gradient ($\nabla$ Action Value), with blue indicating smaller gradients and magenta representing higher gradients (Figure 7 [92]).

ness of Algorithm 12 in locating low-action configurations. As discussed earlier, the main purpose of Algorithm 12 is to perform a systematic and exhaustive exploration of the configuration space to uncover multiple local minimum points. The integration of the Lattice search strategy with the MP-AIDEA framework enhances the adaptability and organisation of the optimisation process, enabling a more thorough investigation of the variational landscape.

**Mountain Pass solution.** Starting from the orbits presented in Figures 6.1 and 6.2 as initial configurations, the Mountain Pass Algorithm 17 yielded a saddle-type periodic solution with an

action value of 11.706.

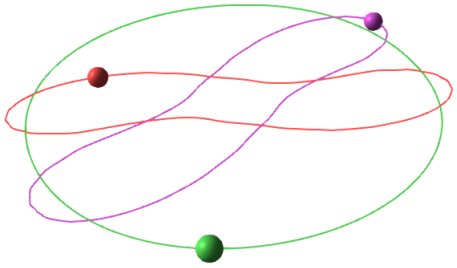


Figure 6.6: Mountain Pass solution (Figure 8 [92])

**Morse index.** The periodic trajectories shown in Figures 6.1, 6.2, 6.3, and 6.4 possess a discrete Morse index of 0 on $\mathbb{I}$, which characterises them as local minimum points of the action functional. In contrast, the orbit illustrated in Figure 6.6 exhibits a discrete Morse index $> 0$ on $\mathbb{I}$, indicating that it corresponds to a non-minimising, saddle-type solution.

**Intra- and trans-level distances.** The intra-level ($\rho_{\mathrm{IL}}$) and trans-level ($\rho_{\mathrm{TL}_k}$) distance values associated with the orbits in Figures 6.1, 6.2, 6.3, and 6.4 are displayed in Figure 6.7. Each orbit is represented by blue, yellow, green, and pink points, respectively. These data summarise the various solutions identified by Algorithm 12 after NoG = 8 generations.

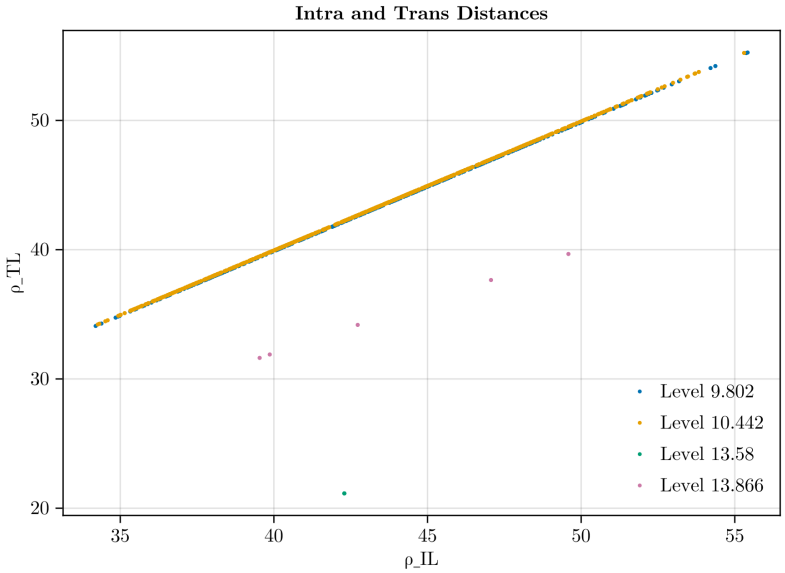



Figure 6.7: The values of $\rho_{\mathrm{IL}}$ and $\rho_{\mathrm{TL}_k}$ of Figures 6.1, 6.2, 6.3 and 6.4. Each point represents a local minimum point, with colours distinguishing the four levels (Figure 9 [92]).

## 6.2 Symmetry group $G = D_6$

Following a setup analogous to the previous case, we now consider a Mountain Pass solution under a different symmetry constraint, namely $G = D_6$. As starting configurations for Algorithm 17, we use two periodic trajectories in $\mathbb{R}^2$ obtained via Algorithm 12. These correspond to the 8-shaped orbit with an action value of 5.858, shown in Figure 6.8, and the 8-shaped orbit with an action value of 9.3, depicted in Figure 6.9.

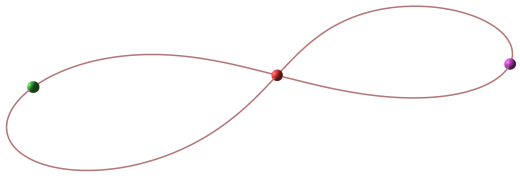


Figure 6.8: 8-shape orbit of action level of 5.858 (Figure 10 [92])

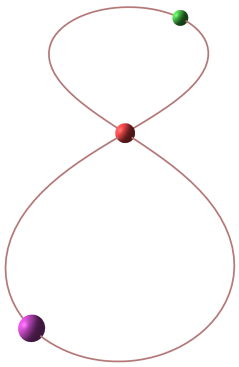


Figure 6.9: 8-shape orbit of action level of 9.298 (Figure 11 [92])

As in the preceding example, only a limited number of generations was required in Algorithm 12. Setting NoG = 10 proved sufficient to detect two distinct periodic orbits, which were the only solutions within this configuration exhibiting a small gradient norm of the action functional.

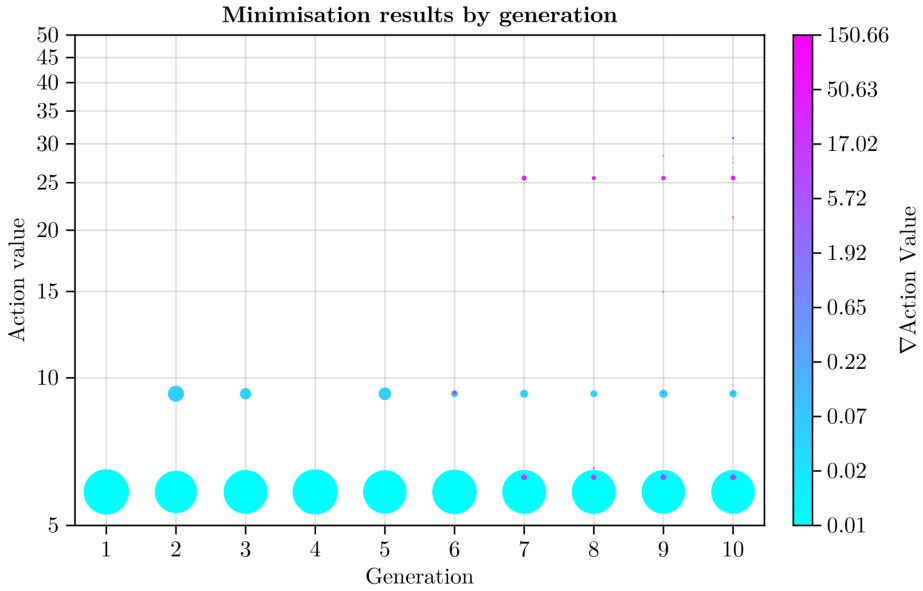



Figure 6.10: Minimisation results by generation. The plot represents the action values of identified solutions across generations in logarithmic scale. Larger dots indicate a higher number of solutions found at that action value, while the colour gradient represents the magnitude of the action gradient ($\nabla$ Action Value), with blue indicating smaller gradients and magenta representing higher gradients (Figure 12 [92]).

Figure 6.10 shows the evolution of the action values obtained across the optimisation process. A dominant minimum with an action value of 5.858 was consistently identified throughout the generations, whereas the secondary minimum at 9.3 appeared only from the second generation onward. This behaviour is consistent with the observations from the previous case. Furthermore, beginning with the eighth generation, several configurations exhibiting a large gradient norm were detected. This occurred because Algorithm 12 did not converge in certain regions of the search domain—a phenomenon that is expected when the optimisation procedure decomposes the space into smaller subdomains, some of which may not contain minimising solutions.
Using the orbits shown in Figures 6.8 and 6.9 as initial configurations, the Mountain Pass Algorithm 17 produced a saddle-type periodic solution with an action value of 9.60.

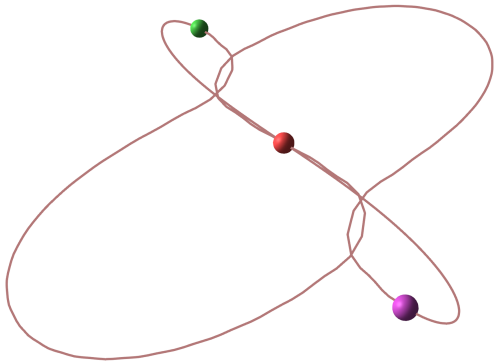


Figure 6.11: Mountain Pass solution (Figure 13 [92])

**Morse index.** The periodic orbits shown in Figures 6.8 and 6.9 exhibit a discrete Morse index of 0 on $\mathbb{I}$, classifying them as local minimum points of the action functional. Conversely, the orbit depicted in Figure 6.11 has a discrete Morse index $> 0$ on $\mathbb{I}$, indicating that it corresponds to a non-minimising, saddle-type configuration.

**Intra- and trans-level distances.** The intra-level ($\rho_{\mathrm{IL}}$) and trans-level ($\rho_{\mathrm{TL}_k}$) distances associated with the orbits in Figures 6.8 and 6.9 are displayed in Figure 6.12. The blue and yellow markers correspond to the respective orbits. These points summarise the distinct periodic solutions detected by Algorithm 12 after NoG = 10 generations.

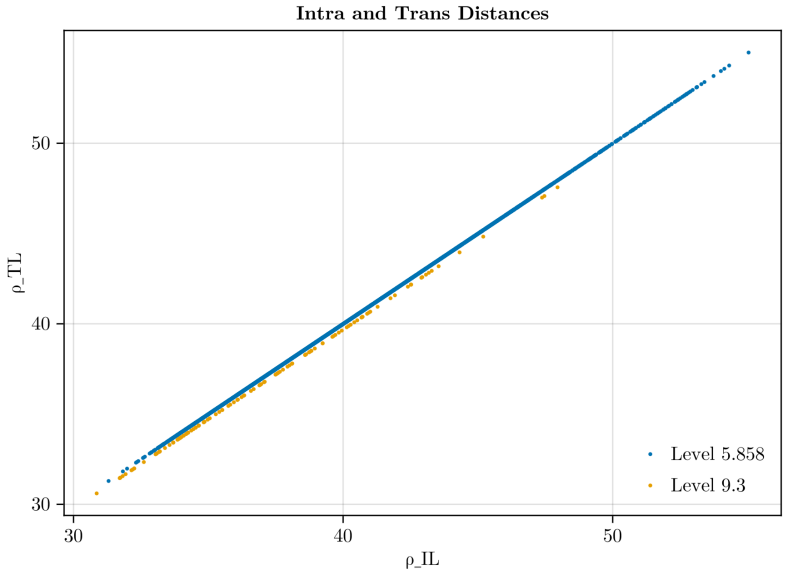



Figure 6.12: The values of $\rho_{\mathrm{IL}}$ and $\rho_{\mathrm{TL}_k}$ of Figures 6.8 and 6.9. Each point represents a local minimum point, with colours distinguishing the two levels (Figure 14 [92]).

# Chapter 7

# Computer-assisted proofs

We now describe the concrete implementation of the algebraic, analytic, and operator-theoretic constructions on Fourier series that are required for the computer-assisted validation framework developed in this Chapter. The underlying mathematical ingredients (such as Fourier multiplication, integer and fractional powers, reciprocals, square roots, antiderivatives, vector-valued extensions, and Banach-space fixed-point arguments) are classical and well established in the literature on analytic function spaces and rigorous numerics. No claim of originality is made for these theoretical components when considered in isolation.

The contribution of the present work lies instead in the systematic integration of these standard constructions into a single, coherent, and fully rigorous computational framework, and in its realisation as a reusable software infrastructure. While each of the validated operations employed here exists in some form in the literature, the present work brings them together within a single, unified setting in which nonlinear maps, vector-valued structures, compact operators, and Newton-like fixed-point schemes can be combined consistently within the same Banach space. In particular, the entire framework is implemented in the `Julia` programming language [70]. To the best of our knowledge, `Julia` has not previously been used to develop a computer-assisted proof infrastructure of comparable scope for analytic problems based on Fourier series. Importantly, however, the framework is not intrinsically tied to the use of Fourier coefficients: alternative functional representations, such as Chebyshev polynomial expansions, can be incorporated with minimal modifications, as the code is designed to accommodate different families of basis functions in a modular and extensible manner. Nevertheless, the contribution is not tied to the choice of programming language: the essential novelty lies in the systematic organisation of the underlying mathematical and algorithmic structures, which could be reproduced in other programming environments without conceptual difficulty.

From a methodological standpoint, the key contribution consists in organising these classical constructions so that they can be combined automatically and safely. Target spaces are generated consistently, parity and truncation are handled transparently, analytic tail bounds are propagated through all operations, and interval arithmetic ensures rigorous enclosure at every stage. This design allows high-level proof strategies (such as contraction-mapping arguments and a posteriori validation of numerical approximations) to be expressed directly in code without sacrificing mathematical rigour. Building on this infrastructure, the Chapter develops a Newton-like fixed-point framework in which defect bounds and operator norms are computed and verified automatically, enabling the certification of existence and uniqueness of solutions to nonlinear functional equations. Together, these components form the operational core that makes it possible to turn numerical approximations of periodic solutions into fully validated mathematical results, within a systematic, extensible, and reusable implementation.

The Chapter is organised as follows. Section 7.1 introduces the motivation, problem formulation, and overall methodology of the computer-assisted proof framework. Section 7.2 presents the principles of interval arithmetic and its algebraic foundations, which underpin the rigorous numerical

computations used throughout. Section 7.3 discusses the representation of analytic functions in spaces of Fourier series and outlines their implementation. Section 7.4 details the validated operations on these series, including addition, multiplication, division, and the application of methods such as Newton-Raphson and DFFT. The framework is then extended in Section 7.5 to vector-valued series, and in Section 7.6 to linear and nonlinear operators, together with the definition of norms and antiderivatives. Finally, Section 7.7 presents the fixed-point validation strategy used to rigorously establish the existence of periodic solutions, supported by the implemented algorithms. Altogether, this Chapter bridges the gap between numerical computation and analytical rigour by providing a computer-assisted methodology for the certification of solutions to nonlinear functional equations.

## 7.1 Introduction to CAPs

Computer-assisted proofs (CAPs) are mathematical arguments in which computers play an essential and irreplaceable role in establishing the correctness of a result. These are not simply automated calculations performed as part of a traditional proof, but rather, they are proofs where some parts, often vast, highly complex, or numerically delicate, are verified entirely by computational means. The justification provided by a CAP typically consists of a combination of traditional analytical reasoning and rigorously controlled numerical verifications carried out by a computer. In many cases, especially in the study of differential equations, dynamical systems, or combinatorially large problems, such proofs allow mathematicians to rigorously settle questions that would be unreachable by human effort alone.

The motivation for introducing computer-assisted methods stems from the intrinsic limitations of human computation when facing mathematical problems that involve enormous case-checking, intricate inequality verifications, or analysis in infinite-dimensional spaces. While these issues are often tractable in principle, they may be in practice too tedious or error-prone to handle manually. Computers offer a means to tame such complexity, but only if they are employed with care: approximate numerics or floating-point arithmetic alone are insufficient for mathematical proof. CAPs thus rely on rigorous numerics, methods like interval arithmetic that can account for and control rounding errors, and on the formal verification of algorithms to ensure correctness across all steps. The resulting proofs, when executed carefully, carry the same level of mathematical rigor as any traditional analytic proof.

The concept gained widespread attention in 1976 with the controversial but groundbreaking proof of the Four Color Theorem by Kenneth Appel and Wolfgang Haken (see [113]). Their argument reduced the general case to $1,936$ configurations that were each verified using extensive computer programs. The sheer size and opacity of the computer-generated portion raised questions within the mathematical community about the acceptability and epistemic status of such proofs. Nevertheless, this milestone marked the beginning of a broader shift in how computers could be incorporated into the fabric of mathematical practice.

The significance of CAPs was further solidified in the late 1990s with the proof of Kepler's conjecture on the densest packing of spheres in three-dimensional space. The conjecture, first posed in 1611, was proven by Thomas Hales (see [114]) through a combination of analytical arguments and an exhaustive computer-assisted check of thousands of geometric configurations. Due to the complexity and scale of the computations, the proof initially faced difficulties in peer review. This prompted Hales to initiate the Flyspeck Project (see [115]), which culminated in a fully formalised and machine-verified version of the proof in 2014. This development helped to bridge the gap between informal computer-assisted proofs and fully formalised, logically complete ones.

In the realm of dynamical systems and differential equations, CAPs have been particularly fruitful. Warwick Tucker, in a landmark result from 2002 [116], provided a rigorous proof of the existence of the Lorenz attractor and verified its chaotic properties. His work combined deep insights from dynamical systems theory with validated numerics, specifically, the use of interval arithmetic to

rigorously control the error bounds in numerical integration. Tucker's method demonstrated the viability of combining computational power with theoretical tools to obtain mathematically sound results about complex, nonlinear systems.
Historically, these advances build on foundational work in numerical analysis and validated computation by researchers such as Robert E. Moore, who developed interval arithmetic in the 1960s to provide guaranteed bounds for numerical computations (see [117]). Over time, a vibrant community has developed around the tools and techniques required for CAPs, contributing algorithms for rigorous ODE integration, verified global optimisation, and computer-assisted bifurcation analysis.
The tools employed in computer-assisted proofs vary widely depending on the domain. In some cases, symbolic computation software (such as Mathematica [118] or Maple [119]) is used to handle algebraic simplifications and case splitting. In more delicate contexts, particularly those involving floating-point operations, rigorous frameworks such as the CAPD library [16], INTLAB [120], or Arb [121] are employed to manage rounding errors and provide provably correct enclosures of solutions. Meanwhile, formal verification tools such as Coq [122], Isabelle/HOL [123], and Lean [124] are being increasingly adopted to ensure logical soundness from the ground up, allowing for fully formalised mathematics that can be mechanically verified with no gaps in reasoning.
As mathematical research continues to push into domains of higher complexity, the role of computer-assisted proofs is only expected to grow. They are now viewed not as a last resort but as a powerful complement to human reasoning, an extension of the mathematician's toolkit. CAPs open the door to solving problems that combine deep structure with heavy computation, where neither symbolic manipulation nor numerical approximation alone would suffice. Moreover, they encourage a more rigorous approach to numerical mathematics itself, where error bounds and correctness guarantees are treated with the same seriousness as logical deduction.

In this Chapter, we delve into the detailed workings of computer-assisted proofs in the context of periodic solutions for differential equations arising from the classical $n$-body problem. Specifically, we demonstrate how such proofs can be constructed and validated using a carefully designed numerical framework that combines functional analytic estimates with interval arithmetic. Central to our approach is a software package implemented in the `Julia` programming language [70], which we have developed to rigorously verify the existence of periodic orbits discovered through numerical simulation. Our methodology draws from the work of G. Arioli and collaborators (see for instance [54–57]), whose pioneering research has established a robust and flexible foundation for applying CAPs to nonlinear problems in mathematical physics. Building on their framework, we adapt and extend these techniques to the setting of our problem, providing a fully self-contained computational pipeline for orbit validation. The complete source code implementing the methods described in this Chapter is available in our repository [125].

**Problem formulation.** Our main objective is to establish a rigorous framework for studying periodic solutions of period $2\pi$ to nonlinear ordinary differential equations of the form

$$\ddot{q} = f(q), \qquad q \in \mathbb{R}^n.$$

A natural reformulation expresses the problem as a fixed-point equation

$$q = \partial^{-2} f(q), \tag{7.1}$$

where $\partial^{-2}$ denotes the second antiderivative operator on the space of continuous $2\pi$-periodic functions with average zero. This fixed-point perspective will be our central focus: by developing a computational framework for rigorously representing, manipulating, and bounding periodic functions, we aim to construct a pathway toward validated proofs of existence for such solutions.
To achieve this, three ingredients are indispensable. First, we must introduce *interval arithmetic*, which provides a way to control rounding and truncation errors arising from finite precision. Second,

we must define a suitable *functional space*, equipped with a norm and a basis, in which analytic periodic functions can be represented and studied. Third, we must design a computational framework to *handle functions*, enabling algebraic and analytic operations to be carried out rigorously on their representations. We now describe these components in turn, beginning with interval arithmetic, the numerical backbone of all subsequent constructions.

## 7.2 Interval arithmetic

Interval arithmetic provides the language in which all our computations are carried out. In essence, it replaces real numbers by sets (intervals) that enclose them, and then extends the elementary operations of arithmetic to these sets in such a way that all possible round-off errors and uncertainties are rigorously controlled. This ensures that the true mathematical values are always contained within the computed results, making it an indispensable tool for computer-assisted proofs.
In interval arithmetic, a real number $s \in \mathbb{R}$ is represented on a computer by an interval

$$S = [S^-, S^+],$$

where $S^- \leq s \leq S^+$ and where both endpoints $S^-$ and $S^+$ belong to the set of machine-representable numbers (e.g. floating-point numbers following the IEEE 754 standard). For instance, the fraction $1/3$, which cannot be exactly represented in binary arithmetic, may be represented by the interval $[0.333, 0.334]$. The true value is then guaranteed to lie within the enclosure.
By performing computations with such intervals rather than with single floating-point numbers, one can ensure that all intermediate and final results are rigorously enclosed. For example, if $x \in [a_1, b_1]$ and $y \in [a_2, b_2]$, then interval addition is defined as

$$x + y \in [a_1 + a_2,\, b_1 + b_2],$$

and interval multiplication as

$$x \cdot y \in [\min(a_1a_2, a_1b_2, b_1a_2, b_1b_2),\, \max(a_1a_2, a_1b_2, b_1a_2, b_1b_2)].$$

In this way, every operation produces a new interval guaranteed to contain the true result. Thus, rounding errors and uncertainties do not accumulate in an uncontrolled way, but are rigorously propagated and tracked throughout the computation.
Interval arithmetic is therefore not only a numerical technique but a rigorous algebraic framework for enclosing real quantities. It is especially important in computer-assisted proofs, where uncontrolled numerical errors could otherwise invalidate the proof. By ensuring that each step of the computation remains within explicitly bounded enclosures, interval arithmetic preserves the correctness of the overall argument.
In practice, implementing interval arithmetic from scratch is unnecessary, as robust and efficient libraries already exist. In our work, we rely on the `Julia` ecosystem, which provides high-performance tools for validated numerics. In particular, we use the packages `IntervalArithmetic.jl` and `IntervalLinearAlgebra.jl`. The first provides the core functionality of interval arithmetic, fully compliant with the IEEE 1788-2015 standard for interval arithmetic [126], while the second offers interval-based routines for linear algebra, including verified solutions of linear systems and rigorous enclosures of eigenvalues [127]. These packages allow us to seamlessly integrate rigorous numerics into our `Julia` implementation of computer-assisted proofs.
To formalise this notion, we introduce the concept of a *standard set*. A real number $s \in \mathbb{R}$ is represented by a standard set

$$S = [S^-, S^+] \in \mathrm{std}(\mathbb{R}),$$

that encloses $s$, where $S^-$ and $S^+$ are machine-representable numbers. The collection $\mathrm{std}(\mathbb{R})$ of all such standard sets forms the surrogate of $\mathbb{R}$ used in rigorous computations. The same notation

extends naturally to higher-dimensional spaces: if $X_1, X_2$ are spaces with standard sets, then

$$\mathrm{std}(X_1 \times X_2) = \{S_1 \times S_2 : S_1 \in \mathrm{std}(X_1),\ S_2 \in \mathrm{std}(X_2)\}.$$

In this way, vector-valued and matrix-valued quantities can be represented and manipulated within the same rigorous framework.
The philosophy of interval arithmetic is therefore clear: by replacing real numbers with rigorously enclosing sets, every computation remains mathematically valid and numerically certified. Before extending this philosophy from numbers to functions, we first formalise the algebraic structure underlying interval arithmetic. This algebraic framework makes explicit how interval operations rigorously enclose their real counterparts and will serve as the basis for later extensions to functional spaces.

### 7.2.1 Interval arithmetic algebra

The algebra of interval arithmetic is based on the principle that operations on intervals should enclose all possible results of the corresponding real operation applied to any pair of values from the operand intervals. Formally, given a binary operation $\star$ and two intervals

$$X = [x_1, x_2], \quad Y = [y_1, y_2],$$

the interval extension of $\star$ is defined as

$$X \star Y = \{\, x \star y \mid x \in [x_1, x_2],\ y \in [y_1, y_2] \,\}.$$

In words, $X \star Y$ is the set of all possible values of $x \star y$, where $x$ and $y$ range independently over their respective intervals.
If $\star$ is monotone in each argument over the given intervals (as is the case for addition, subtraction, and multiplication, and for division whenever $0 \notin Y$), then the extreme values of $X \star Y$ occur at the endpoints of $X$ and $Y$. Writing out all combinations, one obtains

$$[x_1, x_2] \star [y_1, y_2] = \big[\min\{x_1 \star y_1, x_1 \star y_2, x_2 \star y_1, x_2 \star y_2\},\ \max\{x_1 \star y_1, x_1 \star y_2, x_2 \star y_1, x_2 \star y_2\}\big],$$

provided the operation $x \star y$ is well defined for all $x \in [x_1, x_2]$ and $y \in [y_1, y_2]$.
From this general definition, the four basic arithmetic operations can be stated explicitly:

$$\begin{aligned}
[x_1, x_2] + [y_1, y_2] &= [\, x_1 + y_1,\ x_2 + y_2 \,], \\
[x_1, x_2] - [y_1, y_2] &= [\, x_1 - y_2,\ x_2 - y_1 \,], \\
[x_1, x_2] \cdot [y_1, y_2] &= \big[\min(x_1y_1, x_1y_2, x_2y_1, x_2y_2),\ \max(x_1y_1, x_1y_2, x_2y_1, x_2y_2)\big], \\
\frac{[x_1, x_2]}{[y_1, y_2]} &= [x_1, x_2] \cdot \frac{1}{[y_1, y_2]}, \quad \text{with } \frac{1}{[y_1, y_2]} = \Big[\tfrac{1}{y_2}, \tfrac{1}{y_1}\Big], \ \text{if } 0 \notin [y_1, y_2].
\end{aligned}$$

Division requires extra care if the denominator interval contains zero. In such cases, the reciprocal $1/[y_1, y_2]$ may consist of two disjoint unbounded intervals, and it is often convenient to represent the result as a union of intervals (a so-called *multi-interval*).

**Example.** Consider the affine function

$$f(a, b, x) = a \cdot x + b,$$

with $a = [1, 2]$, $b = [5, 7]$, and $x = [2, 3]$. Then

$$f(a, b, x) = ([1, 2] \cdot [2, 3]) + [5, 7] = [\, 1 \cdot 2,\ 2 \cdot 3 \,] + [5, 7] = [2, 6] + [5, 7] = [7, 13].$$

Thus, for any choice of $a \in [1,2]$, $b \in [5,7]$, and $x \in [2,3]$, the exact value $f(a,b,x)$ is guaranteed to lie inside the computed enclosure $[7,13]$.

**Elementary functions.** Interval arithmetic extends naturally beyond the four basic operations to elementary functions such as powers, exponentials, logarithms, and trigonometric functions. If $f : \mathbb{R} \to \mathbb{R}$ is monotone on the interval $[x_1, x_2]$, then the range of $f$ on that interval can be computed directly by evaluating the function at its endpoints:

$$f([x_1, x_2]) = \big[\min\{f(x_1), f(x_2)\},\ \max\{f(x_1), f(x_2)\}\big].$$

From this principle one obtains, for example:

$$\begin{aligned} a^{[x_1,x_2]} &= [a^{x_1}, a^{x_2}], \qquad a > 1, \\ \log_a[x_1, x_2] &= [\log_a(x_1), \log_a(x_2)], \qquad a > 1,\ 0 < x_1 < x_2, \\ [x_1, x_2]^n &= [x_1^n, x_2^n], \qquad n \in \mathbb{N} \text{ odd}. \end{aligned}$$

For even powers, care must be taken. For instance, if $n = 2$ and $x \in [-1,1]$, then $x^2 \in [0,1]$, even though repeated interval multiplication $[-1,1] \cdot [-1,1]$ would give the looser enclosure $[-1,1]$.
For functions that are only *piecewise monotone*, the range over an interval cannot be determined solely from the endpoints. In such cases, one must also consider the *critical points*, that is, points where the derivative vanishes or the function changes monotonicity. For example, the function $\sin(x)$ alternates between increasing and decreasing, with critical points at $n\pi$ and $\frac{\pi}{2} + n\pi$ for all $n \in \mathbb{Z}$. To compute the enclosure of $\sin([x_1, x_2])$, one evaluates sin at the interval endpoints $x_1$, $x_2$, and at any critical points lying inside $[x_1, x_2]$. The smallest and largest of these values provide the tightest possible enclosure. If the interval $[x_1, x_2]$ contains more than one full oscillation of $\sin(x)$, that is, at least two extrema, then the function attains all values between $-1$ and 1 within that range. In this case, the best possible enclosure is simply the trivial bound $[-1,1]$. The same reasoning applies to other periodic or oscillatory functions, such as $\cos(x)$.
This algebra of intervals, encompassing both elementary and transcendental operations, provides the rigorous backbone of all our numerical computations: every operation produces a guaranteed enclosure of the true mathematical result. The next step is to carry this philosophy from numbers to functions. In complete analogy with $\mathrm{std}(\mathbb{R})$, we will construct $\mathrm{std}(\mathcal{F})$, a surrogate functional space where analytic periodic functions are represented by interval-enclosed Fourier coefficients. This extension allows us to manipulate functions, through addition, multiplication, integration, or differentiation, with the same rigour and transparency as real numbers.

## 7.3 Representation of functions

One of the central challenges in computer-assisted proofs is to design a representation of analytic functions that is at the same time faithful to the underlying mathematics and amenable to rigorous computation. A purely symbolic description is too abstract to be implemented, while a naive numerical approximation fails to provide the error control required for proofs. The right compromise is to work in a carefully chosen Banach space of analytic functions, which captures the relevant structure, and then to construct a computable surrogate of this space inside interval arithmetic. This two-step approach separates the analytic and computational aspects: first we specify the model space, which encodes the characteristics of the problem, such as periodicity, analyticity, and quantitative bounds, then we build its surrogate, where exact coefficients are replaced by certified interval enclosures and high-frequency contributions are grouped into rigorously controlled tails. In this way, every computational object corresponds to a genuine analytic function, and every operation carries a rigorous error bound.
In the present context, our goal is to study periodic solutions of the $n$-body problem. The functions under consideration are continuous $2\pi$-periodic trajectories with average zero, as explained in

Equation (7.1). For such functions, it is natural to adopt a Fourier representation: trigonometric polynomials form a dense subset of continuous periodic functions by the Stone-Weierstrass theorem, ensuring that any such function can be approximated arbitrarily well by a finite Fourier expansion. This property makes Fourier series not only mathematically justified but also computationally advantageous, providing a complete and orthogonal basis for encoding periodic trajectories. Hence, Fourier coefficients serve as the natural coordinates for our Banach space and for its computable surrogate within interval arithmetic.

### 7.3.1 A functional space

To rigorously represent analytic periodic functions, we introduce a Banach space of $2\pi$-periodic functions that admit analytic continuation to a complex strip. This space will provide the analytic setting for the computer-assisted proofs developed in the following sections.

**Definition 7.1.** Given $\rho > 0$, set

$$\mathcal{D}_\rho = \{t \in \mathbb{C} : |\Im(t)| < \rho\},$$

where $\Im(t)$ denotes the imaginary part of $t$. Denote by $\mathcal{F}_\rho$ the vector space of all $2\pi$-periodic analytic functions $f : \mathcal{D}_\rho \to \mathcal{C}$ of the form

$$f(t) = \sum_{k=1}^{\infty} f_k \sin(kt) + \sum_{k=1}^{\infty} f'_k \cos(kt), \qquad t \in \mathcal{D}_\rho,$$

which take real values for real arguments, and for which the norm

$$\|f\|_\rho = \sum_{k=1}^{\infty} e^{\rho k}|f_k| + \sum_{k=1}^{\infty} e^{\rho k}|f'_k| \tag{7.2}$$

is finite.

The space $(\mathcal{F}_\rho, \|\cdot\|_\rho)$ provides the mathematical backbone for our construction: it encodes analyticity and controls the growth of Fourier coefficients. The parameter $\rho$ plays the role of a regularity scale: the larger $\rho$, the wider the complex strip of analyticity allowed for the functions, and the faster the Fourier coefficients are required to decay. In particular, $\|f\|_\rho < \infty$ ensures exponential decay of the coefficients, so that analytic properties of $f$ are captured in a quantitative way.
This setting is well adapted to rigorous computation. On the one hand, $\mathcal{F}_\rho$ is a Banach algebra under pointwise multiplication, which guarantees that products of functions remain within the same space. On the other hand, the norm $\|\cdot\|_\rho$ is compatible with the natural operations used in dynamical systems and PDE analysis: differentiation corresponds to multiplying Fourier coefficients by $k$, while integration divides them by $k$. These operations are continuous on $\mathcal{F}_\rho$, and their effect on the norm can be tracked explicitly.
Altogether, $\mathcal{F}_\rho$ provides a precise analytic framework in which the qualitative properties of functions (periodicity, analyticity, even/odd decomposition) and their quantitative features (decay of Fourier coefficients, algebraic closure under multiplication) are simultaneously controlled. It is this combination of structure and stability that makes $\mathcal{F}_\rho$ the right space to approximate by a computable surrogate.

**A computable surrogate.** The space $\mathcal{F}_\rho$ is not directly accessible to computation. Our goal is to build a surrogate $\mathrm{std}(\mathcal{F}_\rho)$ inside interval arithmetic, mirroring the way $\mathrm{std}(\mathbb{R})$ replaces real numbers by interval enclosures. The guiding idea is simple:

- each Fourier coefficient $c_n$ is replaced by an interval $\tilde{c}_n \in \mathrm{std}(\mathbb{R})$ guaranteed to contain it;

- higher-frequency contributions, beyond a truncation index, are grouped into structured "tail sets," again represented by intervals.

Concretely, fix $n \geq 1$ and consider odd $2\pi$-periodic functions. Let

$$U = (U_1, \dots, U_n) \in \mathrm{std}(\mathbb{R}^n), \qquad V = (V_n, V_{n+1}, \dots) \in \mathrm{std}(\mathbb{R}^{\mathbb{N}}),$$

where $U_k$ and $V_m$ are intervals. The surrogate set determined by $(U, V)$ consists of all functions of the form

$$f(t) = \sum_{k=1}^{n} u_k \sin(kt) + \sum_{m=n}^{\infty} v_m \sin(mt), \qquad u_k \in U_k,\ v_m \in V_m.$$

Even functions are treated analogously with cosines, and general functions decompose into even and odd parts. This construction yields a computationally effective Banach surrogate: functions can be added, multiplied, differentiated, or integrated by acting directly on their interval coefficients, with all operations tracked rigorously inside $\mathrm{std}(\mathcal{F}_\rho)$.

**Proposition 7.2.** *When equipped with this norm, $\mathcal{F}_\rho$ as defined in Definition 7.1 is a Banach space.*

*Proof.* To prove that the given space $\mathcal{F}_\rho$ is a Banach space with the given norm, we need to verify two main things:

1. The space $\mathcal{F}_\rho$ is complete with respect to the given norm.
2. The norm satisfies the properties of a norm.

*Step 1: $\mathcal{F}_\rho$ is a normed space.* A space is a normed space if it is equipped with a norm that satisfies the following properties:

- *Positivity*: $\|f\| \geq 0$, and $\|f\| = 0$ if and only if $f = 0$.
- *Scalability*: $\|\alpha f\| = |\alpha| \|f\|$ for any scalar $\alpha$.
- *Triangle inequality*: $\|f + g\| \leq \|f\| + \|g\|$ for any functions $f, g \in \mathcal{F}_\rho$.

For the given norm

$$\|f\|_\rho = \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |f_k| + \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |f_k'|,$$

we check each condition:

- *Positivity*: The terms $\mathrm{e}^{\rho^k} |f_k|$ and $\mathrm{e}^{\rho^k} |f_k'|$ are both non-negative because $f_k$ and $f_k'$ are real-valued. Therefore, $\|f\|_\rho \geq 0$, and if $\|f\|_\rho = 0$, then each Fourier coefficient $f_k = 0$ and $f_k' = 0$, implying $f(t) = 0$ for all $t$.
- *Scalability*: If $f(t)$ is scaled by a constant $\alpha$, then the Fourier coefficients scale as $f_k \to \alpha f_k$ and $f_k' \to \alpha f_k'$. Hence,

$$\|\alpha f\|_\rho = \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |\alpha f_k| + \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |\alpha f_k'| = |\alpha| \left( \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |f_k| + \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |f_k'| \right) = |\alpha| \|f\|_\rho.$$

- *Triangle inequality*: For functions $f(t)$ and $g(t)$, their Fourier coefficients $f_k, f_k'$ and $g_k, g_k'$ satisfy

$$\|f + g\|_\rho = \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |(f_k + g_k)| + \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |(f_k' + g_k')|.$$

By the triangle inequality for real numbers, we have

$$|f_k + g_k| \le |f_k| + |g_k| \quad \text{and} \quad |f_k' + g_k'| \le |f_k'| + |g_k'|.$$

Hence,

$$\|f + g\|_\rho \le \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} (|f_k| + |g_k|) + \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} (|f_k'| + |g_k'|).$$

This is equal to

$$\|f\|_\rho + \|g\|_\rho,$$

showing that the triangle inequality holds.

Thus, $\mathcal{F}_\rho$ is indeed a normed space with the given norm.

*Step 2: $\mathcal{F}_\rho$ is a Banach space.* Therefore, we need to show that it is complete, i.e., every Cauchy sequence in $\mathcal{F}_\rho$ converges to a limit in $\mathcal{F}_\rho$.
Let $\{f_n\}$ be a Cauchy sequence in $\mathcal{F}_\rho$ with respect to the norm $\|\cdot\|_\rho$. This means that for any $\epsilon > 0$, there exists an $N$ such that for all $m, n \ge N$,

$$\|f_n - f_m\|_\rho < \epsilon.$$

Writing this out explicitly:

$$\sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |f_{n,k} - f_{m,k}| + \sum_{k=1}^{\infty} \mathrm{e}^{\rho^k} |f_{n,k}' - f_{m,k}'| < \epsilon.$$

This implies that for each $k$, the sequences $\{f_{n,k}\}$ and $\{f_{n,k}'\}$ are Cauchy sequences of real numbers. Since the real numbers are complete, each sequence $\{f_{n,k}\}$ and $\{f_{n,k}'\}$ converges to some limit $f_k$ and $f_k'$, respectively. Let $f(t)$ be the function defined by

$$f(t) = \sum_{k=1}^{\infty} f_k \sin(kt) + \sum_{k=1}^{\infty} f_k' \cos(kt).$$

We now check that $f(t)$ belongs to $\mathcal{F}_\rho$ and that $f_n \to f$ in the norm $\|\cdot\|_\rho$.

- *Analyticity*: Since each $f_n(t)$ is analytic on $\mathcal{D}_\rho$ and the limit function $f(t)$ is defined by a pointwise limit of analytic functions, $f(t)$ is also analytic on $\mathcal{D}_\rho$.
- *Fourier series*: Since $f_k$ and $f_k'$ are the limits of the corresponding sequences $f_{n,k}$ and $f_{n,k}'$, we have $f(t)$ as the limit of the Fourier series of $f_n(t)$.
- *Convergence in the norm*: Since $\|f_n - f\|_\rho \to 0$ as $n \to \infty$ (which follows from the Cauchy property of the sequence $\{f_n\}$ in the norm $\|\cdot\|_\rho$), the function $f(t)$ is the limit of the sequence $\{f_n\}$ in $\mathcal{F}_\rho$.

Thus, every Cauchy sequence in $\mathcal{F}_\rho$ converges to an element in $\mathcal{F}_\rho$, proving that $\mathcal{F}_\rho$ is a Banach space. We have shown that $\mathcal{F}_\rho$ with the given norm is a normed space, and we have demonstrated that it is complete. Therefore, $\mathcal{F}_\rho$ is a Banach space.

□

**Proposition 7.3.** *When equipped with this norm, $\mathcal{F}_\rho$ as defined in Definition 7.1 is a Banach algebra.*

*Proof.* To prove that $\mathcal{F}_\rho$ is a Banach algebra with respect to the given norm, we need to show that the norm satisfies the sub-multiplicative property, i.e., that for all $f, g \in \mathcal{F}_\rho$,

$$\|fg\|_\rho \le \|f\|_\rho \|g\|_\rho.$$

*Step 1: Expression for $fg$.* We start by noting the form of the functions in $\mathcal{F}_\rho$. A function $f \in \mathcal{F}_\rho$ can be written as

$$f(t) = \sum_{k=1}^{\infty} f_k \sin(kt) + \sum_{k=1}^{\infty} f'_k \cos(kt),$$

where the Fourier coefficients $f_k$ and $f'_k$ are real, and the norm of $f$ is given by

$$\|f\|_\rho = \sum_{k=1}^{\infty} e^{\rho^k} |f_k| + \sum_{k=1}^{\infty} e^{\rho^k} |f'_k|.$$

For two functions $f, g \in \mathcal{F}_\rho$, we can express their product $fg$ as

$$(fg)(t) = \left( \sum_{k=1}^{\infty} f_k \sin(kt) + \sum_{k=1}^{\infty} f'_k \cos(kt) \right) \cdot \left( \sum_{k=1}^{\infty} g_k \sin(kt) + \sum_{k=1}^{\infty} g'_k \cos(kt) \right).$$

Using the product of trigonometric functions, we expand this as:

$$(fg)(t) = \sum_{k=1}^{\infty} (f_k g_k) \sin^2(kt) + \sum_{k=1}^{\infty} (f_k g'_k) \sin(kt)\cos(kt) + \sum_{k=1}^{\infty} (f'_k g_k) \sin(kt)\cos(kt) + \sum_{k=1}^{\infty} (f'_k g'_k) \cos^2(kt).$$

The terms in the expansion will lead to new Fourier coefficients for the product $fg$, say $(h_k, h'_k)$. The key point is that each of the Fourier coefficients for the product $fg$ will be a combination of products of the coefficients $f_k, f'_k, g_k, g'_k$.

*Step 2: Estimating the Fourier coefficients of $fg$.* Let $h_k$ and $h'_k$ be the Fourier coefficients of $fg$. We will estimate the magnitude of these coefficients. Using the fact that products of sines and cosines can be rewritten as sums of sines and cosines of different frequencies (via trigonometric identities), we obtain the following estimates for the Fourier coefficients:

$$|h_k| \le \sum_{k=1}^{\infty} |f_k||g_k|, \quad |h'_k| \le \sum_{k=1}^{\infty} |f'_k||g'_k|.$$

Thus, the Fourier coefficients of $fg$ satisfy the following bounds:

$$|h_k| \le \sum_{k=1}^{\infty} |f_k||g_k|, \quad |h'_k| \le \sum_{k=1}^{\infty} |f'_k||g'_k|.$$

*Step 3: Applying the norm to the product.* The norm of the product $fg$ is then given by

$$\|fg\|_\rho = \sum_{k=1}^{\infty} e^{\rho^k} |h_k| + \sum_{k=1}^{\infty} e^{\rho^k} |h'_k|.$$

Using the estimates for $|h_k|$ and $|h'_k|$, we have:

$$\|fg\|_\rho \le \sum_{k=1}^{\infty} e^{\rho^k} \left( \sum_{k=1}^{\infty} |f_k||g_k| \right) + \sum_{k=1}^{\infty} e^{\rho^k} \left( \sum_{k=1}^{\infty} |f'_k||g'_k| \right).$$

Now, applying the Cauchy-Schwarz inequality to each sum:

$$\|fg\|_\rho \le \left( \sum_{k=1}^{\infty} e^{\rho^k} |f_k| \right) \left( \sum_{k=1}^{\infty} e^{\rho^k} |g_k| \right) + \left( \sum_{k=1}^{\infty} e^{\rho^k} |f'_k| \right) \left( \sum_{k=1}^{\infty} e^{\rho^k} |g'_k| \right).$$

This gives us:

$$\|fg\|_\rho \le \|f\|_\rho \|g\|_\rho.$$

Thus, $\mathcal{F}_\rho$ is a Banach algebra with respect to the given norm. □

We now introduce the odd and even subspaces of $\mathcal{F}_\rho$ and the corresponding norm on their direct sum.

**Definition 7.4.** The subspaces of odd and even functions in $\mathcal{F}_\rho$ are denoted by $\mathcal{A}_\rho$ and $\mathcal{B}_\rho$, respectively. On the direct sum $\mathcal{F}_\rho^P$, we define the norm

$$\|q\|_\rho = \max_{1\le i\le P} \|q_i\|. \tag{7.3}$$

### 7.3.2 Implementation

We now describe how the surrogate space $\mathrm{std}(\mathcal{F})$ is realised concretely in code. The implementation is modular: first we define the notion of a series space (capturing metadata such as parity, truncation, and precision parameters), then the indexing and size helpers, and finally the series objects themselves, which carry coefficients and validated error enclosures.

#### 7.3.2.1 Series spaces and parameters

The first building block of the implementation is the definition of a *series space*. A series space encodes the metadata that determines how a Fourier series is stored and manipulated: the type of its coefficients, whether the series is truncated, its parity (even, odd, or complete), and the numerical parameters governing its operations.
In details, the abstract type `SeriesSpace{CT,ET,T}` carries two pieces of information:

- the coefficient type `CT`, typically a floating-point type or an interval type,
- a Boolean flag `T` indicating whether the space is truncated.

Helper functions such as `coefftype` and `istruncated` allow these parameters to be queried generically. Algorithm 18 summarises the structure and basic query functions associated with `SeriesSpace{CT,ET,T}`.

**Algorithm 18** Abstract interface for series spaces

**Type:** `SeriesSpace{CT,ET,T}`
Parameters:

- `CT`: coefficient type (e.g. floating-point, interval),
- `ET`: error type (e.g. floating-point, interval),
- `T`: Boolean flag indicating truncation.

```
function CoeffType(S)
    return coefficient type CT of S
end function
function ErrType(S)
    return error type ET of S
end function
function IsTruncated(S)
    return truncation flag T of S
end function
```

We now introduce the notion of a **Fourier space**, which encodes the analytic and computational parameters required to represent Fourier series rigorously. The type `FourierSpace{CT,ET,T,P}` carries three qualifiers:

- `CT`: the coefficient type (e.g. `Float64` or `Interval{BigFloat}`),
- `ET`: the error type (e.g. `Float64` or `Interval{BigFloat}`),

- `T`: a Boolean flag indicating whether the space is truncated,
- `P`: the parity of the basis (`Even`, `Odd`, or `Complete`).

Each instance of a Fourier space contains the following fields:

- $N$: the number of Fourier modes retained,
- $R$: the analyticity radius of the function space,
- `rational_power`: an algorithmic object specifying the method and precision parameters (such as the truncation order and the number of FFT points) to be used for computing rational powers of series,
- `inverse_power`: an algorithmic object specifying the method and precision parameters to be used for computing series reciprocals.

The constructor enforces the natural conditions $N \geq 1$ and $R > 0$, ensuring that only meaningful parameter sets can be created. The structure is summarised in Algorithm 19.

**Algorithm 19** Fourier space structure

**Type:** `FourierSpace{CT,ET,T,P}`

Parameters:

- `CT`: coefficient type(e.g. `Float64` or `Interval{BigFloat}`),
- `ET`: the error type (e.g. `Float64` or `Interval{BigFloat}`),
- `T`: truncation flag (Boolean),
- `P`: parity (`Even`, `Odd`, `Complete`).

Fields:

- $N$ `::` `Int` (number of Fourier modes),
- $R$ `::` `Float64` (analyticity radius),
- `rational_power :: Algorithm` (algorithmic object specifying the method and precision parameters),
- `inverse_power :: Algorithm` (algorithmic object specifying the method and precision parameters).

**function** FourierSpace($N, R,$ `rational_power`, `inverse_power`)
  **if** $N < 1$ **then**
    **error:** "Number of coefficients must be positive."
  **end if**
  **if** $R \leq 0$ **then**
    **error:** "Radius must be strictly positive."
  **end if**
  **return** new Fourier space with given parameters
**end function**

The parity parameter determines whether the basis functions are cosines (`Even`), sines (`Odd`), or both (`Complete`). In addition, the enumeration `SqrtAlgorithm` selects the method used for computing square roots of series: Taylor expansion (`TaylorAlgorithm`), Newton iteration (`NewtonAlgorithm`), or FFT-based evaluation (`FFTAlgorithm`). The mathematical justification and practical trade-offs between these three approaches will be discussed in Section 7.4.

**Specialisations and utilities.** For convenience, we introduce type aliases that specialise the parametric type `FourierSpace{CT,ET,T,P}` by fixing the parity parameter $P$. This yields the three concrete families `FourierEvenSpace{CT,ET,T}`, `FourierOddSpace{CT,ET,T}`, and `FourierCompleteSpace{CT,ET,T}`, together with truncated variants (`TruncatedFourierEvenSpace`, `TruncatedFourierOddSpace`, and `TruncatedFourierCompleteSpace`).
The shorthand `FourierEOSpace{CT,ET,T}` is also provided to denote a space of either even or odd type. From a practical point of view, two additional utilities simplify manipulation of spaces. First, the function `Base.copy` is overloaded so that duplicating a space produces a new `FourierSpace` with identical parameters. Second, the helper `truncate_space` converts a non-truncated Fourier space into its truncated counterpart, preserving all parameters which are N,R,`rational_power`, `inverse_power`,P, while setting the truncation flag $T$ = `true` and converting the coefficients type from interval to floating point, if needed.

#### 7.3.2.2 Size and indexing helpers

All series data (`C` for coefficients and `E` for error enclosures) are stored in a *single flat vector*. To address, for instance, "the 5th odd coefficient of the second variable" or to convert between *frequency* indices and *storage* indices, we centralise layout rules in a small set of helpers. This avoids hard-coded offsets and prevents off-by-one errors across arithmetic kernels.
In details, for a Fourier space with resolution $N$ one stores for each space:

- *Even* space: $N + 1$ cosine coefficients;
- *Odd* space: $N$ sine coefficients;
- *Complete* space: $2N + 1$ coefficients, stored contiguously as [even ‖ odd].

Error arrays (`E`) mirror this parity layout with lengths $2N + 1$ (even), $2N$ (odd), and $4N + 1$ (complete). Algorithms 20 and 21 summarise the only rules one really needs to remember during implementation.
To translate between stored indices and Fourier frequencies, we use parity–aware shifts:

- *Coefficient shifts.* For even spaces, the constant mode corresponding to frequency $k = 0$ sits at index 1, so $\textsc{cidx\_shift}(\texttt{Even}, :C) = 1$. For odd spaces, modes start at $k = 1$, so $\textsc{cidx\_shift}(\texttt{Odd}, :C) = 0$. In complete spaces, the even block uses shift 1; the odd block is placed after the even block, i.e. its indices are offset by $N + 1$.
- *Error shifts.* The same idea applies to the enclosure vector, with the odd error block in the complete case starting after the even error block, i.e. offset $2N + 1$.

Hence the frequency ranges are recovered by

$$\begin{aligned}\textsc{freq\_coeff}(\cdot, :C) &= \textsc{cidx}(\cdot, :C) - \textsc{idx\_shift}(\cdot, :C),\\ \textsc{freq\_error}(\cdot, :E) &= \textsc{eidx}(\cdot, :E) - \textsc{idx\_shift}(\cdot, :E).\end{aligned}$$

For complete spaces, this yields $\textsc{freq\_coeff}(\texttt{Complete}, :C) = 0{:}N$ and $\textsc{freq\_error}(\texttt{Complete}, :E) = 0{:}2N$ and the coefficient indices split as

$$\text{even block } = 1{:}(N+1), \qquad \text{odd block } = (N+2){:}(2N+1),$$

and analogously for the error vector with block lengths $2N + 1$ and $2N$. These slices correspond to the index windows accessed in high-level operations as $[\,\cdot\,\texttt{[Even]}]$ and $[\,\cdot\,\texttt{[Odd]}]$.
For routines that work on a finite subset of modes (e.g. matrix-based compact operators), we expose parity-aware selectors that return the storage index windows for the first $N$ (odd) or $N$+1 (even) coefficient modes of each parity block. These respect the layout and offsets in Algorithm 21.

**Algorithm 20** Sizes of coefficient and error blocks

**function** N_COEFFS($S$)
  **if** $S$ even **then return** $N + 1$
  **else if** $S$ odd **then return** $N$
  **else return** $2N + 1$ ▷ complete
  **end if**
**end function**
**function** N_COEFFS($S$, $P$) ▷ complete space, specific parity $P$
  **if** $P$ = `Even` **then return** $N + 1$
  **else return** $N$
  **end if**
**end function**
**function** N_ERRORS($S$)
  **if** $S$ even **then return** $2N + 1$
  **else if** $S$ odd **then return** $2N$
  **else return** $4N + 1$
  **end if**
**end function**
**function** N_ERRORS($S$, $P$) ▷ complete space, specific parity $P$
  **if** $P$ = `Even` **then return** $2N + 1$
  **else return** $2N$
  **end if**
**end function**

**Algorithm 21** Storage windows for coefficients and errors

**function** FIRST_CIDX($S$) **return** 1
**end function**
**function** FIRST_CIDX($S$, $P$) ▷ complete space
  **if** $P$ = `Even` **then return** 1
  **else return** $N + 2$
  **end if**
**end function**
**function** LAST_CIDX($\cdot$) **return** FIRST_CIDX($\cdot$) + N_COEFFS($\cdot$) $-1$
**end function**
**function** CIDX($\cdot$) **return** FIRST_CIDX($\cdot$) : LAST_CIDX($\cdot$)
**end function**
**function** FIRST_EIDX($S$) **return** 1
**end function**
**function** FIRST_EIDX($S$, $P$) ▷ complete space
  **if** $P$ = `Even` **then return** 1
  **else return** $2N + 2$
  **end if**
**end function**
**function** LAST_EIDX($\cdot$) **return** FIRST_EIDX($\cdot$) + N_ERRORS($\cdot$) $-1$
**end function**
**function** EIDX($\cdot$) **return** FIRST_EIDX($\cdot$) : LAST_EIDX($\cdot$)
**end function**

Moreover, Algorithm 22 concretely implements the parity-aware index mapping described above: it returns the storage index windows for the even and odd coefficient blocks, consistent with the

parity-dependent shifts and block offsets defined earlier, so that each subspace correctly corresponds to its frequency range in the complete Fourier layout.

**Algorithm 22** Subspace index windows (Fourier spaces)

**function** GET_SUBSPACE_INDICES($S$=`FourierEvenSpace`, $N$)
    **return** [ 1:($N$+1) ] ▷ even block: $k = 0, \ldots, N$
**end function**
**function** GET_SUBSPACE_INDICES($S$=`FourierOddSpace`, $N$)
    **return** [ 1:$N$ ] ▷ odd block: $k = 1, \ldots, N$
**end function**
**function** GET_SUBSPACE_INDICES($S$=`FourierCompleteSpace`, $N$)
    **return** [ 1:($N$+1), ($S.N$+2):($S.N$+1+$N$) ] ▷ concatenate even and odd index ranges
**end function**

*Remark.* Centralising the indexing rules serves a dual purpose. From a mathematical point of view, it guarantees consistency: every high-level operation (addition, product, tail bookkeeping, FFT projection, antiderivatives) relies on knowing (i) the size of each block, (ii) where each block starts in storage, and (iii) how to translate between storage indices and frequencies. Encoding this logic once keeps arithmetic kernels simple and eliminates the risk of layout bugs.
From a computational point of view, the same design is highly efficient. By storing all coefficients and error enclosures in a single flat vector rather than in nested structures, memory usage is reduced and data locality is improved. This lowers allocation overhead, enables vectorised operations, and allows direct use of optimised BLAS routines. As a result, both memory footprint and runtime performance scale favourably, which is essential when working with Fourier series containing thousands of modes.

#### 7.3.2.3 Series objects and invariants

While `FourierSpace` encodes the *parameters* of a functional space, the actual data of a Fourier series are stored in a `Series{T,V}` object. It couples three components:

- `space`: a `SeriesSpace` describing the resolution, parity, and other parameters,
- `C`: the coefficient vector, storing Fourier coefficients,
- `E`: the error vector, storing rigorous tail bounds.

As in the case of Fourier spaces, the constructor is responsible for enforcing the natural invariants of the structure: type consistency between `C` and the space, correct lengths for coefficient and error vectors, and proper handling of truncated versus non-truncated spaces (see Section 7.3.2.1). These checks are summarised in Algorithm 23.
In addition, the element types of the two arrays follow a strict convention. The coefficient vector `C` may contain either standard floating-point numbers or interval-valued entries (in our specific case `Interval{BigFloat}`) when rigorous enclosures are required. The error vector `E`, on the other hand, stores only upper bounds on the corresponding coefficient norms and therefore never consists of intervals. Type relations are enforced automatically, using Algorithm 18:

$$\begin{cases} \text{ELTYPE}(\texttt{E}) = T, & \text{if ELTYPE}(\texttt{C}) = \texttt{Interval\{T\}}, \\ \text{ELTYPE}(\texttt{E}) = \text{ELTYPE}(\texttt{C}), & \text{otherwise.} \end{cases}$$

Here, ELTYPE$(\cdot)$ refers to the standard `Julia` function returning the element type of an array, while CoeffType$(\cdot)$ and ErrType$(\cdot)$ are user-defined methods that specify the expected coefficient type and error type for a given Fourier space, as explain in Algorithm 18. This guarantees that

both arrays are numerically compatible while respecting the semantic distinction between interval coefficients and scalar error bounds.

Moreover, for ease of use, two outer constructors are provided:

- `Series(space,C,E)`: calls the inner constructor directly, taking both coefficients and errors as explicit arguments,
- `Series(space,C)`: requires only the coefficients; the error vector is then created automatically, filled with zeros in the non-truncated case, or left uninitialised when the space is truncated.

Finally, the function `Base.copy` is overloaded so that duplicating a series produces a new object with identical space, coefficients, and errors. This ensures that series objects can be safely duplicated without sharing memory between the original and the copy. In addition, the function `Base.zero` is overloaded for series spaces: calling `zero(space)` constructs a new `Series` object in the given space, with all coefficients and errors set to zero. This provides a convenient and type-stable way to initialise series for subsequent computations. Similarly, the function `Base.one` is overloaded so that `one(y::AbstractSeries)` returns the multiplicative identity in the same space, i.e. `one(y.space)`. This ensures that generic code relying on neutral elements (e.g. initialising products, scaling series, or setting up iterative algorithms) remains consistent and type-safe. e schemes) works seamlessly with Fourier series objects.

**Algorithm 23** Construction and invariants of a Series

**Type:** `Series{T,V}` with fields `space`, `C`, `E`.

**function** SERIES($space, C, E$)
  **if** ELTYPE($C$) ≠ COEFFTYPE($space$) **then** ▷ Algorithms 18-19
    attempt conversion; if impossible, raise error
  **end if**
  **if** $|C|$ ≠ N_COEFFS($space$) **then** ▷ Algorithm 20
    raise error
  **end if**
  **if** space is non-truncated **then** ▷ Algorithm 18
    **if** $E$ = `nothing` **then**
      raise error
    **end if**
    **if** $|E|$ ≠ N_ERRORS($space$) **then** ▷ Algorithm 20
      raise error
    **end if**
    **if** COEFFTYPE($E$) ≠ COEFFTYPE($space$) **then** ▷ Algorithms 18-19
      raise error
    **end if**
  **end if**
  **return** new `Series(space,C,E)`
**end function**
**function** SERIES($space, C$)
  **if** space is truncated **then** ▷ Algorithm 18
    initialise $E$ as uninitialised vector of length N_ERRORS($space$) ▷ Algorithm 20
  **else**
    initialise $E$ as zero vector of length N_ERRORS($space$) ▷ Algorithm 20
  **end if**
  **return** `Series(space,C,E)`
**end function**

## 7.4 Handling functions: operations

Having introduced the framework of interval arithmetic (Section 7.2) and the representation of functions through Fourier expansions with interval coefficients (Section 7.3), we now turn to the operations that allow us to manipulate such functions rigorously. The fundamental requirement is that every operation, such as addition, subtraction, scalar multiplication, division, or composition with nonlinear maps, must propagate interval bounds, so that the result always remains a verified enclosure of the true function.

From the theoretical side, this follows from the Banach algebra structure of $\mathcal{F}\rho$, which is stable under these operations. The surrogate space $\mathrm{std}(\mathcal{F})$ mirrors this stability by acting on families of interval coefficients and structured tails. This makes it possible to treat functions as algebraic objects: operations are defined directly on the coefficient intervals, but they reflect exactly the analytic manipulations of $\mathcal{F}\rho$.

This abstract viewpoint is naturally realised in an object-oriented setting. One may regard a Fourier expansion as an abstract "object" equipped with methods for all the standard algebraic operations. By overloading the familiar operators `+`, `-`, `*`, and `/`, these objects can be manipulated as transparently as scalars, while rigour is automatically preserved by interval arithmetic. In practice, we build on existing validated numerical libraries (specifically, the `Julia` packages `IntervalArithmetic.jl` [126] and `IntervalLinearAlgebra.jl` [127]), which implement the IEEE 1788-2015 standard—so that our contribution lies not in reimplementing interval arithmetic from scratch, but in constructing a functional layer tailored to Fourier series and analytic function spaces.

The advantage of this design is twofold. First, it mirrors the underlying mathematics: the algebraic rules of $\mathcal{F}_\rho$ are faithfully reproduced at the computational level. Second, it provides a modular foundation: once the primitive operations are carefully defined, they can be reused seamlessly in more advanced routines, such as Newton's method, contraction mappings, and rigorous orbit validation.

Finally, the technique is not confined to Fourier series. In principle, any analytic function space (e.g. Fourier, Chebyshev, Legendre, or others) can be handled in the same way.

In the remainder of this Section we detail how these principles are put into practice. We begin with the basic algebra of series (addition, subtraction, scalar multiplication), then turn to nonlinear operations such as multiplication and integer powers, and finally address more delicate procedures including reciprocals, square roots, and validated iterative methods.

### 7.4.1 Basic operations

Fix $N \in \mathbb{N}$. We work with *series objects*, each represented by a family of interval coefficients $(c_n)_{n=1}^N$ with $c_n \in \mathrm{std}(\mathbb{R})$. A series object belongs to a specified series space, which fixes parameters such as parity, resolution, and truncation. Linear operations are defined *componentwise* on these interval coefficients, and are therefore independent of the underlying representation. In practice, our implementation specialises to Fourier series objects, but the same formulae apply verbatim to other choices such as Chebyshev or Legendre series, provided their storage layouts are compatible.

#### 7.4.1.1 Addition and subtraction

Given two series objects

$$s_1 \sim (c_n)_{n=1}^N, \qquad s_2 \sim (d_n)_{n=1}^N,$$

defined in the same space, their sum and difference are

$$(s_1 \pm s_2) \;\sim\; (\, c_n \pm d_n \,)_{n=1}^N.$$

When interval coefficients are used, the updates $c_n \pm d_n$ are carried out with interval arithmetic (see Section 7.2), ensuring that the resulting series object rigorously encloses the true sum or difference.

**Implementation.** In the code, addition and subtraction are defined by overloading `Base.:+` and `Base.:-` for the `Series` type. The implementation is split into two main stages: (i) verifying that the input series live in a compatible space, and (ii) carrying out the coefficientwise updates. Compatibility is checked by a dedicated function, and the result space is constructed by a helper constructor. Both procedures are summarised in Algorithm 24.

**Algorithm 24** Check compatibility of two Fourier spaces and construct sum space

**function** CHECK_COMPATIBLE_SPACES_SUM($s_1, s_2$)
    **return true** if and only if the following hold:
    - same parity: PARITY($s_1$) = PARITY($s_2$) ▷ Algorithm 19
    - same radius: $s_1.R = s_2.R$
    - same resolution: $s_1.N = s_2.N$
    - same coefficient type: COEFFTYPE($s_1$) = COEFFTYPE($s_2$) ▷ Algorithm 19
    - same truncation flag: ISTRUNCATED($s_1$) = ISTRUNCATED($s_2$) ▷ Algorithm 19

    Otherwise, return **false**.
**end function**
**function** GET_SPACE_SUM($s_1, s_2$)
    **if** CHECK_COMPATIBLE_SPACES_SUM($s_1, s_2$) = **false then**
        **error:** "Spaces not compatible for sum."
    **end if**
    **return** new Fourier space with:
    - coefficient type = `coefftype`($s_1$),
    - truncation flag = `istruncated`($s_1$),
    - parity = `parity`($s_1$),
    - $N = s_1.N$, $R = s_1.R$, inheriting $k_{\text{power}}$, $k_{\text{inv}}$, `SQRT`, and `FFT_points`.

**end function**

Once the result space is prepared, the addition or subtraction proceeds coefficientwise, as it is summarised in Algorithm 25.

**Algorithm 25** Addition and subtraction of series

**function** ADD/SUB($s_1, s_2$)
    $s_{\text{sum}} \leftarrow$ GET_SPACE_SUM($s_1.space, s_2.space$) ▷ Algorithm 24
    $s \leftarrow$ zero series in $s_{\text{sum}}$ ▷ see Section 7.3.2.3
    **for** each coefficient index $n$ **do**
        $s.C[n] \leftarrow s_1.C[n] \pm s_1.C[n]$
    **end for**
    **if** $s_{\text{sum}}$ is non-truncated **then** ▷ Algorithm 19
        **for** each error index $k$ **do**
            $s.E[n] \leftarrow s_1.E[n] + s_2.E[n]$
        **end for**
    **end if**
    **return** $s$
**end function**

*Remark.* Addition and subtraction are implemented componentwise on the coefficient family, independent of the chosen basis. Interval arithmetic is used whenever coefficients are intervals, ensuring validated results. In-place variants overwrite a preallocated series to reduce allocations, while pre-

serving the same rigorous semantics.

#### 7.4.1.2 Negation

For a series object $s \sim (c_n)_{n=1}^{N}$ its negation is defined coefficientwise by

$$(-s) \sim \big((-c_n)\big)_{n=1}^{N}.$$

Interval negation is used when coefficients are intervals, ensuring that the result encloses the true negated coefficients. If error enclosures are present, they are copied unchanged, since they represent nonnegative bounds.

**Implementation.** Negation is implemented by overloading `Base.:-` for the `Series` type. The procedure has two steps: (i) allocate a new result series $S$ in the same space as $s$, (ii) set $S.C = -s.C$ while copying $s.E$ (if present). Both allocating and in-place forms are provided: `Base.:-` creates a fresh series, `neg!(s,S)` writes into a preallocated target $S$, and `neg!(s)` negates in place. The algorithm is summarised in Algorithm 26.

**Algorithm 26** Negation of a series

```
1:  function NEGATE(s)
2:      S ← zero series in a copy of s.space                ▷ see Section 7.3.2.3
3:      for each coefficient index n do
4:          S.C[n] ← −s.C[n]
5:      end for
6:      if s.space is non-truncated then                    ▷ Algorithm 19
7:          for each error index n do
8:              S.E[n] ← s.E[n]                             (errors are preserved)
9:          end for
10:     end if
11:     return S
12: end function
```

*Remark.* The negation operation is basis-agnostic: it acts only on the coefficient family and therefore applies equally to Fourier, Chebyshev, or Legendre series. The interval arithmetic framework ensures that, if coefficients are intervals, their negation is also rigorous. In practice, the in-place variants reduce memory allocations while preserving identical semantics.

#### 7.4.1.3 Scalar multiplication and division

For a scalar $\alpha \in \mathbb{R}$ and a series object $s \sim (c_n)_{n=1}^{N}$, the scalar product is defined coefficientwise by

$$\alpha s \sim \big(\alpha c_n\big)_{n=1}^{N}.$$

Division by a nonzero scalar is realised as multiplication by its reciprocal:

$$\tfrac{1}{\alpha} s \sim \left(\tfrac{1}{\alpha} c_n\right)_{n=1}^{N}, \qquad \alpha \neq 0.$$

**Implementation.** In the code, scalar multiplication and scalar division are defined by overloading `Base.:*` and `Base.:/` for the `Series` type. The implementation is split into two main stages: (i) allocating a new result series in the same space as the input, and (ii) updating its coefficient and error arrays. Coefficients are scaled by $\alpha$, while errors (if present) are scaled by $|\alpha|$ to preserve

nonnegativity. Division is implemented as multiplication with the reciprocal. Both allocating and in-place variants are provided. The procedure is summarised in Algorithm 27.

**Algorithm 27** Scalar multiplication and division of series

```
function ScalarMultiply(α, s)
    S ← zero(s.space)                                   ▷ see Section 7.3.2.3
    for each coefficient index n do
        S.C[n] ← α · s.C[n]
    end for
    if s.space is non-truncated then                    ▷ Algorithm 19
        for each error index n do
            S.E[n] ← supabs(α) · s.E[n]
        end for
    end if
    return S
end function
function ScalarDivide(s, α)
    if 0 ∈ α then
        error: "Division by zero."
    end if
    return ScalarMultiply(α^{-1}, s)
end function
```

*Remark* 7.5. When $\alpha \in \mathbb{R}$, scaling by a scalar is basis-independent: it acts only on the coefficients and enclosures, and therefore applies identically across Fourier, Chebyshev, and Legendre representations. In the error updates, the use of $\operatorname{supabs}(\alpha)$ is essential, since enclosures represent nonnegative bounds and must be scaled by the absolute value of $\alpha$ to preserve their validity. For real scalars, we simply have

$$\operatorname{supabs}(\alpha) = |\alpha|.$$

Both allocating and in-place variants of the scaling operation are supported, with the in-place version avoiding unnecessary memory allocations. For division by a real scalar $\alpha \neq 0$, the coefficients are scaled by $\alpha^{-1}$ and the enclosures by $|\alpha^{-1}|$ through ScalarMultiply.

#### 7.4.1.4 Norm of a Fourier series

The next fundamental operation is the computation of the norm. Recall from Section 7.3 that the Banach space $\mathcal{F}_\rho$ is equipped with the weighted $\ell^1$-norm (see Subsection 7.3.1, in particular Equation (7.2))

$$\|f\|_\rho = \sum_{k\geq 1} e^{\rho k}\big(|f_k| + |f'_k|\big),$$

which measures analytic size through exponentially weighted Fourier coefficients. In the surrogate space $\operatorname{std}(\mathcal{F})$, this definition is translated into a norm on interval coefficients: we take the supremum of each interval, multiply by the corresponding weight, and sum over all modes. If the function carries tail enclosures, the error bounds are included in the norm as well. For complete functions (with both even and odd parts), the norm is defined as the maximum of the norms of its components, consistent with Equation (7.3).

**Implementation.** In the code, the norm is implemented by overloading `LinearAlgebra.norm` for the `Series` type. The procedure has three stages: (i) compute the weighted sum of absolute values of the stored coefficients, (ii) if the space is non-truncated, add the contributions from the

error bounds, and (iii) if the space is complete, take the maximum of its even and odd components. The full procedure is summarised in Algorithm 28.

**Algorithm 28** Norm of a Fourier series $s_F$

```
1: function NORM(s_F)
2:     Norm ← 0
3:     for each coefficient index n in s_F.space do                ▷ coefficient part
4:         Norm ← Norm + s_F.space.R^n · SUPABS(s_F[: C, n])       ▷ Remark 7.6 and Algorithm 19
5:     end for
6:     if s_F.space is non-truncated then                          ▷ Algorithm 19
7:         for each error index n in s_F.space do
8:             Norm ← Norm + s_F[: E, n]
9:         end for
10:    end if
11:    if s_F.space is complete then                               ▷ complete series
12:        return max(NORM(s_F[Even]), NORM(s_F[Odd]))
13:    else
14:        return Norm
15:    end if
16: end function
```

*Remark* 7.6. In Algorithm 28, the function SUPABS$(\cdot)$ denotes the *supremum of the absolute value* of a real coefficient or interval, as introduced in Remark 7.5. For a real interval $I = [a, b] \subset \mathbb{R}$, this is given by

$$\text{SUPABS}(I) \;=\; \sup\{\,|x| : x \in I\,\} \;=\; \max\{|a|, |b|\}.$$

This definition ensures that, for any coefficient $c_n \in I$, the value SUPABS$(I)$ provides a rigorous upper bound on its absolute value. When applied to families of coefficients, it guarantees that the computed numerical norm rigorously dominates the analytic norm defined in Section 7.3.

*Remark.* The norm plays a central role both analytically and computationally: it realises the Banach structure of $\mathcal{F}_\rho$ at the level of interval surrogates. Every subsequent operation, whether linear or nonlinear, can be rigorously controlled in terms of this norm, ensuring that computations remain enclosed within $\text{std}(\mathcal{F})$.

#### 7.4.1.5 Bases of the functional space $\mathcal{F}_\rho$

From a theoretical point of view, a basis provides the canonical way to describe the modes of a Fourier series, and more generally of any analytic series space. Each mode corresponds to a fundamental building block (e.g. a sine or cosine in the Fourier setting), and arbitrary series objects are expressed as linear combinations of these blocks with suitable coefficients. A basis thus provides the structure needed for decomposition, coefficientwise operations, norms, and projections.
In our framework, this abstract idea is realised concretely through the iterator type `Basis`, which exposes the admissible modes of a given space one by one. Each iteration step yields a canonical series object supported on a single coefficient or, when available, on a tail slot. Iterating over all indices therefore traverses the complete generating set of the space, covering both explicit modes and error reservoirs.
This construction bridges analytic theory with computer-assisted practice. Analytically, the basis captures the geometry of $\mathcal{F}_\rho$; computationally, the iterator provides a uniform mechanism that can be used in projections, operator representations, and rigorous error control.
Two refinements are important in this setting:

- *Normalisation.* Basis elements may be used in raw form, supported on a single mode or tail

slot, or rescaled to unit norm in the weighted $\ell^1$ structure induced by the analyticity radius $R$. Normalisation is useful for comparing modes or constructing orthonormal families.

- *Parity and completeness.* Fourier spaces decompose into even, odd, or complete parts. A complete basis element naturally splits into an even and an odd component. In addition to coefficient modes, we include *tail slots* representing the error reservoirs, so that the basis spans both the explicit finite modes and the controlled tails.

Formally, a (possibly normalised) basis for a given series space is an ordered family $\{B_n\}_{n\geq 1}$ of series objects, each having exactly one unit entry in either a coefficient position or a tail slot, with all other entries zero. When the normalised option is active, each element is rescaled as $B_n/\|B_n\|$, using Algorithm 28 to reckon the $\|B_n\|$. For complete Fourier spaces, indices first traverse even coefficients, then odd coefficients, and finally the tail slots for each parity, in accordance with the direct sum norm of Equation (7.3).

**Implementation.** As previously explained, a basis in our setting is realised by the iterator type `Basis`. It records the underlying series space together with an optional cutoff `subspace_dim`, limiting the number of explicit coefficient modes before tail slots are reached. Two flags further refine its behaviour: `normalised`, which rescales each element by its norm, and `infinite`, which allows iteration into tail slots (forbidden if the space is truncated). The corresponding constructor and invariants are summarised in Algorithm 29.

**Algorithm 29** Construction and invariants of a `Basis`

**Type:** `Basis{S,N,F}` with fields:

- `space :: Space` (the underlying series space),
- `subspace_dim :: Int` (maximum number of coefficient modes).

Parameters:

- $S$: space type,
- $N$: normalised (`true`/`false`),
- $F$: infinite part included (`true`/`false`).

1: **function** BASIS(*space*; *subspace_dim* = 0, *normalised* = `true`, *infinite* = `false`)
2: **if** `istruncated`(*space*) **and** *infinite* = `true` **then**
3: raise error: "The space is truncated, infinite basis cannot be created."
4: **end if**
5: **return** new `Basis{S, normalised, infinite}` with given *space* and *subspace_dim*
6: **end function**

To implement iteration over a basis, we must first determine when the process should terminate. Since a basis consists of both explicit coefficient modes and, when available, tail (error) slots, the iteration must stop once all admissible modes have been visited. The exact stopping index depends on the parity of the Fourier space (even, odd, or complete), which dictates the number of tail slots, and on the optional cut-off `subspace_dim`. This logic is encapsulated in the stopping rule of Algorithm 30.

Once the stopping rule is defined, the next step is to *materialise* individual basis elements. Given an index $k$, the routine `basis!` constructs the corresponding vector in the functional space: a series with all entries zero except for a single coefficient or tail slot set to 1. For complete spaces, the index is split between even and odd parts before considering tail contributions. This procedure, summarised in Algorithm 31, ensures that every admissible index produces a valid basis element consistent with the structure of the space.

**Algorithm 30** Stopping rule for basis iteration

**function** N_ERRORS_BASIS(*space*)
    **return** number of tail slots depending on the parity of $s$:
    - 1 if *space* is even,
    - 1 if *space* is odd,
    - 2 if *space* is complete.
**end function**
**function** STOP_BASIS_ITERATOR(*space*, $k$, *max_k*)
    **if** *max_k* = 0 **then**
        **return** true if $k >$ N_COEFFS(*space*) + N_ERRORS_BASIS(*space*) ▷ Alg. 20
    **else**
        **return** true if $k >$ N_COEFFS(*space*, *max_k*) + N_ERRORS_BASIS(*space*) ▷ Alg. 20
    **end if**
**end function**

**Algorithm 31** Materialise the $k$-th basis element

**function** BASIS!($B$, $k$, *max_k*)
    *space* ← $B$.space ▷ Algorithm 29
    **if** STOP_BASIS_ITERATOR(*space*, $k$, *max_k*) **then** ▷ Algorithm 30
        **error:** "index $k$ out of range for basis"
    **end if**
    *cap* ← (*max_k* = 0) ? N_COEFFS(*space*) : N_COEFFS(*space*, *max_k*) ▷ Algorithm 20
    $B$ ← zero series in *space* ▷ see Section 7.3.2.3
    **if** *space* is even/odd **then**
        **if** $k \le$ *cap* **then**
            set coefficient $k$ of $B$ to 1
        **else**
            **if** ISTRUNCATED(space) **then** ▷ Algorithm 18
                **error:** "tails not stored"
            **end if**
            set tail slot $k -$ *cap* + N_COEFFS(*space*) of $B$ to 1
        **end if**
    **else if** *space* is complete **then**
        compute $cap_{\text{even}}$ and $cap_{\text{all}}$
        **if** $k \le cap_{\text{even}}$ **then**
            BASIS!($B$[Even], $k$, *max_k*)
        **else if** $cap_{\text{even}} < k \le cap_{\text{all}}$ **then**
            BASIS!($B$[Odd], $k - cap_{\text{even}}$, *max_k*)
        **else if** $k = cap_{\text{all}} + 1$ **then**
            activate the even tail slot
        **else if** $k = cap_{\text{all}} + 2$ **then**
            activate the odd tail slot
        **else**
            **error:** "invalid tail index"
        **end if**
    **end if**
    **return** $B$
**end function**

With the construction, stopping rule, and materialisation procedures in place, we can now define the iteration itself. The iterator traverses the basis of a given space by successively producing each admissible element: at every step, a new series is materialised, optionally normalised by its norm, and returned together with the updated state. The process halts when the stopping rule is triggered. This full iteration mechanism is summarised in Algorithm 32.

**Algorithm 32** Iteration over a basis

```
1: function ITERATE(b, state = 1)                          ▷ b is a Basis, Algorithm 29
2:     if STOP_BASIS_ITERATOR(b.space, state, b.subspace_dim) then   ▷ Algorithm 30
3:         return nothing
4:     end if
5:     B ← zero series in b.space                          ▷ see Section 7.3.2.3
6:     BASIS!(B, state, b.subspace_dim)                    ▷ Algorithm 31
7:     if b is normalised then
8:         return (B/‖B‖, state + 1)                       ▷ Algorithm 28
9:     else
10:        return (B, state + 1)
11:    end if
12: end function
```

*Remark.* The construction above shows how the abstract idea of a basis is translated into a concrete, computable object within our framework. The combination of Algorithms 29, 30 and 31, and 32 ensures that every admissible mode of a Fourier space (coefficients and, when present, tail slots) can be accessed in a systematic and rigorous way. This design has several advantages:

1. *Uniformity across spaces.* Even, odd, and complete Fourier spaces are handled seamlessly, with parity bookkeeping managed internally by the iterator.

2. *Rigorous coverage.* Both explicit coefficients and error slots are included, reflecting that our surrogate stores the entire function, not only a truncated part.

3. *Flexibility.* Optional normalisation and cutoffs allow the same mechanism to serve multiple purposes, from constructing orthonormal families to probing finite-dimensional subspaces.

In practice, iterating over a basis is an essential building block for higher-level routines: it enables projections, stability analysis, and rigorous operator representations in the functional framework developed here.

### 7.4.2 Multiplication between two series

We now turn to the multiplication of two elements of $\mathrm{std}(\mathcal{F})$. Recall that each element is represented by a truncated Fourier polynomial of degree $n$ with interval coefficients, together with interval sets $V_m$ that bound the contribution of the higher-order coefficients beyond $n$. For instance, in the odd case

$$f(t) = \sum_{k=1}^{n} u_k \sin(kt) \; + \; \sum_{m=0}^{2n} v_m(t), \qquad u_k \in U_k,\; v_{mk} \in V_m,$$

and similarly for even functions with cosines. In practice, the error sets $V_m$ are often initialised to zero, but they provide the structure needed to rigorously account for all contributions outside the truncation window.

**Polynomial interactions.** Multiplying two truncated parts corresponds to multiplying two

Fourier polynomials. By the product-to-sum identities

$$\begin{aligned}\sin(kt)\sin(ht) &= \tfrac{1}{2}(\cos((k-h)t) - \cos((k+h)t)),\\ \cos(kt)\cos(ht) &= \tfrac{1}{2}(\cos((k-h)t) + \cos((k+h)t)),\\ \sin(kt)\cos(ht) &= \tfrac{1}{2}(\sin((k+h)t) \pm \sin(|k-h|t)),\end{aligned}$$

each pair of modes $(k, h)$ produces a *sum frequency* $k+h$ and a *difference frequency* $|k-h|$. When these lie within the truncation window ($\le n$), their contributions are added directly to the stored coefficients of the product.

**Tail contributions.** The main difficulty comes from interactions that produce frequencies larger than $n$. To treat them rigorously, we enlarge the error sets $V_m$ using a systematic bookkeeping scheme:

- *Coefficient-coefficient (C-C).* Products of two stored coefficients may generate modes above $n$. Rather than discarding them, these contributions are inserted into the corresponding error slots $V_M$.

- *Coefficient-error (C-E).* When a stored coefficient multiplies an error term of the other factor, arbitrarily high indices are involved. To control them uniformly, we introduce analytic weights

  $$w_m = R^m, \qquad R > 1,$$

  which satisfy $w_{k+h} = w_k w_h$. The contribution of a pair $(k, h)$ is bounded symmetrically by

  $$C_{k,h} = \tfrac{1}{2}\Big(|u_k|\, E_h^G + |v_h|\, E_k^F\Big),$$

  with $E_h^F, E_h^G$ the error enclosures. These bounds are then assigned to the tail slots $V_{k+h}$ and $V_{|k-h|}$, scaled appropriately by $w_k$ or $1/w_k$ so that the resulting sequence fits the weighted norm. If a difference index does not correspond to a positive frequency (e.g. $h < k$), the contribution is collected into a small *base reservoir* determined by the parity of the product space.

- *Error-error (E-E).* Finally, multiplying two error terms yields a contribution

  $$S_{k,h} = E_k^F\, E_h^G,$$

  which is then distributed into the error sets: one part goes to $V_{k+h}$, and another part to the base reservoir. A square-root factor $1/\sqrt{w_{\min(k,h)}}$ is applied so that the resulting bounds remain valid near low modes and the global tail stays summable.

Because the weights $w_m$ grow exponentially, the total tail contribution is dominated by a rapidly convergent series. For example, if $|u_k| \le C_1 w_k^{-1}$ and $|v_h| \le C_2 w_h^{-1}$, then

$$\sum_{k+h>n} |u_k v_h| \;\le\; \sum_{k+h>n} \frac{C_1 C_2}{w_k w_h} = C_1 C_2 \sum_{m>n} \frac{1}{w_m},$$

showing that the cumulative tail is rigorously bounded. Thus every neglected mode is safely enclosed in some $V_m$, ensuring that the product remains inside $\mathrm{std}(\mathcal{F})$.

**Parity bookkeeping.** The parity of the product follows directly from the trigonometric identities:

$$\mathcal{F}^{\mathrm{even}} \cdot \mathcal{F}^{\mathrm{even}} \subset \mathcal{F}^{\mathrm{even}}, \qquad \mathcal{F}^{\mathrm{odd}} \cdot \mathcal{F}^{\mathrm{odd}} \subset \mathcal{F}^{\mathrm{even}}, \qquad \mathcal{F}^{\mathrm{even}} \cdot \mathcal{F}^{\mathrm{odd}} \subset \mathcal{F}^{\mathrm{odd}}.$$

If one factor is complete, then the product is complete as well.

**Implementation.** In the code, multiplication of two Fourier series is implemented by overloading `Base.:*` for the `Series` type. The design splits into three main layers:

1. *Result space construction.* A helper function `get_space_product` determines the parity of the product space (even · even = even, odd · odd = even, mixed = odd, complete = complete) and checks that the two input spaces are compatible (same resolution $N$, radius $R$, coefficient type, truncation flag). If the check fails, an error is raised. Otherwise, a new `FourierSpace` object is constructed with the inherited parameters. This procedure is summarised in Algorithm 33.

2. *Tail weights.* If either input is non-truncated, the multiplication must propagate tails. The function `compute_weights` generates analytic weights $(w_m)$ such that $w_{k+h} = w_k w_h$. These are used to factor contributions into tail slots while ensuring the weighted norm remains bounded. The procedure is given in Algorithm 34.

3. *Coefficient and error updates.* Multiplication produces a new series object in the target space, whose coefficients and error slots are filled according to three types of interactions:

   - coefficient-coefficient interactions (handled by direct convolution, with excess terms truncated or accumulated into tails);
   - coefficient-error interactions (bounded symmetrically using the prescribed weights);
   - error-error interactions (quadratic contributions collected into the tail slots).

   Parity signs arising from trigonometric product-to-sum identities are tracked explicitly, so that each contribution is placed with the correct sign into the appropriate frequency. The complete procedure is summarised in Algorithm 35.

**Algorithm 33** Product space construction

**function** GET_SPACE_PRODUCT($s_1, s_2$)
  Check that $s_1, s_2$ have the same $N, R$, coefficient type, truncation flag. ▷ Algorithm 19
  **if** either is complete **then**
    **return** complete Fourier space with inherited parameters
  **else**
    $P \leftarrow \text{parity}(s_1)+\text{parity}(s_2) \pmod 2$ ▷ Algorithm 19
    **return** Fourier space with parity $P$ and inherited parameters
  **end if**
**end function**

**Algorithm 34** Tail weight generation

**function** COMPUTE_WEIGHTS($R, N_E$)
  $w[0] \leftarrow 1$, $w[1] \leftarrow R$, block size $M \leftarrow 1$
  **for** $k = 2, \ldots, N_E - 1$ **do**
    $w[k] \leftarrow w[k - M] \cdot w[M]$
    **if** $k = 2M$ **then** set $M \leftarrow k$
    **end if**
  **end for**
  **return** $w$
**end function**

*Remark.* The multiplication kernel treats three types of interactions separately (C-C, C-E, E-E). When tails are active, analytic weights ensure multiplicative structure $w_{k+h} = w_k w_h$, which makes the error bookkeeping consistent with the weighted $\ell^1$ norm. Signs are handled explicitly via parity checks, so even/odd/complete spaces behave exactly as in the theoretical product-to-sum rules.

**Algorithm 35** Multiplication of two Fourier series $s_{1F}, s_{2F}$

**function** MULTIPLY($s_{1F}, s_{2F}$)
  $s_{\text{prod}} \leftarrow$ GET_SPACE_PRODUCT($s_{1F}.space, s_{2F}.space$) ▷ Algorithm 33
  $S_F \leftarrow$ zero series in $s_{\text{prod}}$ ▷ see Section 7.3.2.3
  **if** any input is non-truncated **then** ▷ Algorithm 19
    $w \leftarrow$ COMPUTE_WEIGHTS($s_{\text{prod}}.R$, $n\_errors(s_{\text{prod}})$)
  **end if**
  ▷ Coefficient-coefficient block
  **for** frequencies $k$ of $s_{1F}$, $h$ of $s_{2F}$ **do**
    $C \leftarrow s_{1F}[:C, k] \cdot s_{2F}[:C, h]$
    Add contribution of sum frequency $M = k + h$ to $S.C$ (or $S.E$ if $M > N$)
    Add contribution of difference frequency $M = |k - h|$ to $S.C$ if admissible
  **end for**
  ▷ Coefficient-error block (if non-truncated)
  **for** coefficients $k$ of one input, errors $h$ of the other **do**
    Bound $B_{k,h} \leftarrow \frac{1}{2}(|s_{1F}[:C, k]| \cdot s_{2F}[:E, h] + |s_{2F}[:C, k]| \cdot s_{1F}[:E, h])$
    Assign $B_{k,h}$ to error slots $S.E$ using weights $w_k$
  **end for**
  ▷ Error-error block (if non-truncated)
  **for** errors $k$ of $s_{1F}$, $h$ of $s_{2F}$ **do**
    $D \leftarrow s_{1F}[:E, k] \cdot s_{2F}[:E, h]$
    Add $\frac{1}{2}D$ to $S[:E, k + h]$
    Add $\frac{1}{2}D/\sqrt{w_{\min(k,h)}}$ to base error slot
  **end for**
  **return** $S$
**end function**

#### 7.4.2.1 Integer powers

Once multiplication has been defined, it is straightforward to extend the construction to integer powers of a Fourier series. Given $s_F \in \text{std}(\mathcal{F})$ and an integer $n \geq 1$, we define

$$s_F^n = \underbrace{s_F \cdot s_F \cdot \ldots \cdot s_F}_{n \text{ times}},$$

with the product taken in the sense described above. In practice, repeated multiplication is implemented recursively using the standard *exponentiation by squaring* algorithm: for $n = 1$ we return $s_F$, for $n = 2$ we compute $s_F \cdot s_F$, for even $n$ we write $s_F^n = (s_F^2)^{n/2}$, and for odd $n$ we write $s_F^n = s_F \cdot (s_F^2)^{(n-1)/2}$. This approach minimises the number of multiplications required (reducing the complexity from linear to logarithmic in $n$) while preserving the same rigorous control of coefficients and tails at each step. Thus, integer powers of Fourier series are handled within the same framework, with every intermediate product remaining in $\text{std}(\mathcal{F})$.

**Implementation.** In the code, integer powers are defined by overloading `Base.:^` for the `Series` type with an integer exponent. The operation relies on the standard exponentiation–by–squaring algorithm, which reduces the number of multiplications from linear to logarithmic in $n$. Each multiplication step invokes the validated product routine of Section 7.4.2, so coefficient and tail enclosures are propagated rigorously throughout. All intermediate results remain in the same Fourier space, ensuring that the final series stays within $\text{std}(\mathcal{F})$. The procedure is summarised in Algorithm 36.

**Algorithm 36** Exponentiation by squaring for Fourier series $s_F$

```
1: function Power(s_F, n)
2:     if n = 1 then
3:         return s_F
4:     else if n = 2 then
5:         return s_F * s_F                                  ▷ Algorithm 35
6:     else if n is even then
7:         return Power(s_F * s_F, n/2)
8:     else
9:         return s_F * Power(s_F * s_F, (n − 1)/2)
10:    end if
11: end function
```

### 7.4.3 Square root and division

Operations such as taking the reciprocal of a series or extracting its square root are considerably more delicate than addition or multiplication. The difficulty is twofold. First, these operations are nonlinear and cannot be expressed by a simple convolution of Fourier coefficients. Second, the rigorous framework of $\mathrm{std}(\mathcal{F})$ requires that every step comes with explicit error bounds: it is not enough to compute an approximate reciprocal or square root, one must also prove that the true function lies within the prescribed interval enclosure.
In practice, the solution is to reformulate these operations in a way that is compatible with rigorous numerics. This often means turning the nonlinear operation into a *fixed-point problem* that can be solved iteratively, with carefully controlled error propagation. Depending on the context, different strategies are possible:

- **Taylor approximation.** The reciprocal or square root of a series can be expanded into a binomial series around its constant mode, and then truncated to a polynomial. This provides an explicit and easily computable approximation, with the remainder bounded by a geometric estimate. The method is efficient when the perturbation from the constant term is small, but becomes less effective as the relative size of the higher modes grows.

- **Newton–Raphson method.** A more systematic approach is to apply the Newton–Raphson iteration to the defining equation (e.g. $G \cdot F = 1$ for the reciprocal, or $H^2 = F$ for the square root). This produces quadratic convergence to the desired solution, and at each step the multiplication is handled within $\mathrm{std}(\mathcal{F})$ with rigorous tail control. The challenge is to ensure that the iteration remains valid and the error enclosures shrink as expected.

- **DFFT-based methods.** In some settings, discrete Fourier transform techniques (DFFT) can be used to accelerate the computation of nonlinear operations. By moving between coefficient space and physical space, products and nonlinearities can be evaluated efficiently. However, adapting this approach to a rigorous framework is nontrivial, since one must account for aliasing, discretisation, and rounding errors.

Each of these methods provides a different balance between efficiency and rigour. In some cases (for example, low-degree approximations), Newton's approximation is sufficient. In others, especially when high accuracy is needed, Newton–Raphson is preferred. DFFT-based methods offer speed, but require more elaborate error analysis. For this reason, all three approaches deserve careful consideration, and we now describe them in turn.

**Implementation.** In the code, the choice of algorithm for computing the square root of a Fourier series is delegated to the Fourier space's `rational_power` algorithmic object: the square root is

treated as the rational power with exponent 1/2. The object specifies both the computational method and any precision parameters (e.g., truncation order, number of FFT points). Available strategies include:

- `TaylorAlgorithm`: truncated binomial–Taylor expansion (see Section 7.4.3.1);
- `NewtonAlgorithm`: Newton–Raphson fixed-point iteration (see Section 7.4.3.2);
- `FFTAlgorithm`: discrete FFT-based computation in physical space (see Section 7.4.3.3).

The `Julia` method `Base.:sqrt` is overloaded for the `Series` type and dispatches automatically to the corresponding routine, according to the algorithmic specification stored in the Fourier space. This process is summarised in Algorithm 37.

**Algorithm 37** Dispatching square-root computation via `rational_power`

```
function Sqrt(s_F)
    A ← s_F.space.rational_power                          ▷ Algorithm 19
    p ← 1/2
    if A = TaylorAlgorithm then
        return RationalPower_Taylor(s_F, p)
    else if A = NewtonAlgorithm then
        return RationalPower_Newton(s_F, p)
    else if A = FFTAlgorithm then
        return RationalPower_FFT(s_F, p)
    else
        error: "Unknown algorithm in rational_power."
    end if
end function
```

*Remark.* This design keeps all fractional powers, including the square root, under a single `rational_power` mechanism, ensuring a uniform interface and centralised control of method and precision parameters. Switching the algorithm in the Fourier space automatically changes the implementation used by `sqrt` (by setting $p = 1/2$).

As previously discussed, each nonlinear operation requires its own dedicated algorithmic strategy: a method that is efficient and rigorous for one case (e.g. square roots) may be unsuitable for another (e.g. reciprocals). The Newton–Raphson method, being an iterative refinement scheme, could in principle lead to non-termination or excessive recursion if implemented naively. In our framework, Newton's method is therefore reserved exclusively for the computation of rational powers with exponent 1/2, i.e. for square roots, where the iteration count is explicitly capped and monotonicity safeguards are enforced.
Whenever the `rational_power` algorithmic object is set to `NewtonAlgorithm`, the reciprocal $1/F$ is instead computed using the `inverse_power` strategy, which defaults to a Taylor-based approximation. This separation ensures that all nonlinear operations terminate after a finite number of steps while remaining rigorously enclosed. Conversely, when Newton's method is not selected, the reciprocal of a series can be computed using either of the remaining methods (binomial expansion or FFT-based evaluation), as determined by the `inverse_power` specification in the Fourier-space parameters.

#### 7.4.3.1 Taylor approximation

Perhaps the most direct way to define fractional powers of a Fourier series (such as square roots or reciprocals) is via a binomial expansion. Instead of reformulating the problem as an auxiliary functional equation, one expands $(1 - x)^m$ into a power series around a well-chosen reference point

and truncates after finitely many terms. This yields an explicit polynomial approximation whose accuracy is determined by the truncation order and whose remainder can be rigorously bounded by interval arithmetic.

Let $s_F \in \mathrm{std}(\mathcal{F})$ be an even Fourier series with constant term $c_0 > 0$. To simplify the expansion, we first normalise so that the constant coefficient is close to 1:

$$\tilde{s}_F = \frac{1}{c_0}\, s_F, \qquad X = 1 - \tilde{s}_F.$$

In this form, the deviation from the constant mode is captured by $X$, which is small in norm whenever $s_F$ is close to its constant coefficient. In such cases the binomial expansion converges rapidly, and truncating at a moderate order $h$ suffices to obtain a rigorous enclosure. The approach is computationally efficient, though it requires the series to be even (since odd Fourier series vanish at the origin) and the normalisation to ensure a well-conditioned expansion point.

For a rational exponent $m = \frac{1}{n} > 0$, the binomial theorem gives

$$(1-x)^r \;=\; \sum_{i=0}^{\infty} \binom{m}{i} (-x)^i.$$

Truncating after $h$ terms yields the approximation

$$P_h(x) = \sum_{i=0}^{h} \binom{m}{i} (-x)^i.$$

The fractional power of the original series is then approximated by

$$s_F^m \;\approx\; \frac{1}{c_0^m}\, P_h(X).$$

The truncation error is of order $O(\|X\|^{h+1})$, and can be rigorously bounded using interval arithmetic. In this way, the binomial expansion provides both an explicit constructive formula and a certified enclosure for $s_F^m$.

**Convergence proof.** The validity of the Taylor approximation rests on the convergence of the binomial series

$$(1-X)^m = \sum_{i=0}^{\infty} \binom{m}{i} (-X)^i, \qquad m = \tfrac{1}{n} > 0.$$

It is a classical result that this expansion converges whenever $\|X\| < 1$, with the remainder after $h$ terms given by

$$R_{h+1}(X) \;=\; \binom{m}{h+1} (-X)^{h+1} + \binom{m}{h+2} (-X)^{h+2} + \cdots.$$

In norm, this remainder can be bounded as

$$\|R_{h+1}(X)\| \;\le\; C_m \, \|X\|^{h+1},$$

for some constant $C_m$ depending only on $m$. Thus the truncated polynomial

$$P_h(X) = \sum_{i=0}^{h} \binom{m}{i} (-X)^i$$

satisfies

$$\left\| s_F^m - \tfrac{1}{c_0^m} P_h(X) \right\| \;\le\; \frac{C_m}{c_0^m} \, \|X\|^{h+1}.$$

In other words, the approximation error decreases geometrically with the order $h$, provided the

normalised deviation $X$ remains small. By rigorously enclosing $\|X\|$ using interval arithmetic, one obtains certified error bounds for the truncated Taylor expansion, ensuring that the computed result is a valid enclosure of the true value.

**Example** (Convergence example)**.** Consider the even Fourier series

$$F(t) = 1 + \varepsilon \cos t, \qquad |\varepsilon| < 1,$$

and compute $F^{1/2} = \sqrt{1 + \varepsilon \cos t}$ by the binomial expansion $(1+X)^{1/2} = \sum_{j\geq 0} \binom{1/2}{j} X^j$ with $X = \varepsilon \cos t$. Truncating at order $k = 3$ gives

$$\sqrt{1 + \varepsilon \cos t} \;\approx\; 1 + \tfrac{1}{2}\varepsilon \cos t - \tfrac{1}{8}\varepsilon^2 \cos^2 t + \tfrac{1}{16}\varepsilon^3 \cos^3 t.$$

Using the identities

$$\cos^2 t = \tfrac{1}{2}\big(1 + \cos 2t\big), \qquad \cos^3 t = \tfrac{1}{4}\big(3\cos t + \cos 3t\big),$$

the truncated approximation can be written explicitly as a short Fourier series:

$$\begin{aligned}\sqrt{1 + \varepsilon \cos t} \;\approx\; &\Big(1 - \tfrac{1}{8}\varepsilon^2 \cdot \tfrac{1}{2}\Big) + \Big(\tfrac{1}{2}\varepsilon + \tfrac{1}{16}\varepsilon^3 \cdot \tfrac{3}{4}\Big)\cos t \\ &- \Big(\tfrac{1}{8}\varepsilon^2 \cdot \tfrac{1}{2}\Big)\cos 2t + \Big(\tfrac{1}{16}\varepsilon^3 \cdot \tfrac{1}{4}\Big)\cos 3t,\end{aligned}$$

i.e.

$$\sqrt{1 + \varepsilon \cos t} \;\approx\; \big(1 - \tfrac{1}{16}\varepsilon^2\big) + \big(\tfrac{1}{2}\varepsilon + \tfrac{3}{64}\varepsilon^3\big)\cos t - \tfrac{1}{16}\varepsilon^2 \cos 2t + \tfrac{1}{64}\varepsilon^3 \cos 3t.$$

If for example we take $\varepsilon = 0.1$. Then

$$\sqrt{1 + 0.1 \cos t} \;\approx\; 0.999375 + 0.050046875\,\cos t - 0.000625\,\cos 2t + 0.000015625\,\cos 3t.$$

A crude remainder bound for the neglected terms ($j \geq 4$) is

$$\sum_{j=4}^{\infty} \Big|\tbinom{1/2}{j}\Big|\,|\varepsilon|^j \;\leq\; \frac{\big|\binom{1/2}{4}\big|\,|\varepsilon|^4}{1 - |\varepsilon|} \;=\; \frac{0.0390625 \times 10^{-4}}{0.9} \;\approx\; 4.4 \times 10^{-6},$$

showing that the third-order truncation already yields a micro-level enclosure when $|\varepsilon| = 0.1$. More generally, if $\|X\| < 1$ in a Banach-algebra norm, the Taylor remainder decays geometrically like $O(\|X\|^{h+1})$, providing a certified bound that tightens rapidly as the truncation order $h$ increases.

**Convergence rate.** Let $s_F$ be even with $c_0 := s_F(0) > 0$, and set

$$\tilde{s}_F = \frac{s_F}{c_0}, \qquad X = 1 - \tilde{s}_F.$$

For a rational exponent $m = \frac{1}{n} > 0$ the binomial series

$$(1 - X)^m = \sum_{j=0}^{\infty} \binom{m}{j} (-X)^j$$

converges whenever $\|X\| < 1$ (in any Banach–algebra norm compatible with multiplication). Writing $t := \|X\| < 1$, the remainder after truncation at degree $h$ is

$$R_{h+1}(X) \;=\; \sum_{j=h+1}^{\infty} \binom{m}{j} (-X)^j, \qquad \|R_{h+1}(X)\| \;\leq\; \sum_{j=h+1}^{\infty} \Big|\tbinom{m}{j}\Big|\, t^j.$$

For $j \geq h$ the successive ratio satisfies

$$\frac{|\binom{m}{j+1}|\, t^{j+1}}{|\binom{m}{j}|\, t^{j}} = \frac{|m-j|}{j+1}\, t \;\leq\; t,$$

hence the tail is dominated by a geometric series. In particular,

$$\|R_{h+1}(X)\| \leq \frac{|\binom{m}{h+1}|}{1-t}\, t^{h+1}.$$

Consequently,

$$\Big\| s_F^m - \frac{1}{c_0^m} \sum_{j=0}^{h} \binom{m}{j} (-X)^j \Big\| \;\leq\; \frac{|\binom{m}{h+1}|}{(1-t)\, c_0^m}\, t^{h+1}.$$

*Asymptotics.* For $0 < m < 1$ one has $|\binom{m}{j}| \sim \dfrac{C_m}{j^{1+m}}$ as $j \to \infty$ (with $C_m = 1/|\Gamma(-m)|$), so a sharper form is

$$\|R_{h+1}(X)\| \;\lesssim\; \frac{C_m}{1-t}\, \frac{t^{h+1}}{(h+1)^{1+m}}.$$

Thus the error decays essentially geometrically in $h$ with ratio $t = \|X\| < 1$, augmented by a polynomial factor $(h+1)^{-(1+m)}$. In terms of digits, the number of correct significant figures grows approximately linearly with $h$ at slope $-\log_{10} t$, improving further due to the $j^{-(1+m)}$ decay of the coefficients.

**Implementation.** Fractional powers with rational exponent $m = \frac{1}{n} > 0$ are defined by overloading `Base.:^` for the `Series` type. The routine is enabled for even or complete Fourier spaces, under the assumption that the constant mode is positive so that the normalisation step is well posed. The workflow is as follows:

- check that the exponent has numerator 1 (otherwise raise a domain error);
- normalise by the constant coefficient $c_0$, setting $\tilde{s}_F = s_F/c_0$ and $X = 1 - \tilde{s}_F$;
- construct the truncated binomial sum $P_h(X) = \sum_{i=0}^{h} \binom{m}{i}(-X)^i$, using the space parameter `k_power` for the truncation order $h$;
- if the space is non-truncated, add a validated enclosure to the first error slot to account for the tail of order $O(\|X\|^{h+1})$;
- return the rescaled result $s_F^m = (1/c_0^m)\, P_h(X)$.

This procedure ensures that fractional powers are computed with rigorous error control while remaining within the surrogate space $\mathrm{std}(\mathcal{F})$ and it is explained in Algorithm 38.

*Remark* 7.7. The update of $g$ in Algorithm 38 uses a *Horner-like accumulation*: instead of recomputing $X^i$ from scratch, the powers are built incrementally by repeatedly multiplying the previous $g$ by $X$. This reduces the number of multiplications from quadratic to linear in the truncation order, and ensures that each step uses the rigorous multiplication routine (Algorithm 35) to propagate tail enclosures.

*Remark.* The restriction to even (or complete) spaces reflects that odd Fourier series vanish at the origin, making the expansion around the constant mode $c_0$ ill-posed. The enclosure update uses interval arithmetic for all scalar and coefficient operations, so the returned series rigorously bounds the true value $s_F^m$.

**Algorithm 38** Binomial Taylor fractional power of a Fourier series $s_F$

**function** TaylorPower($s_F$, $m = \frac{1}{n}$)
  **preconditions:** $s_F$ even or complete; $m > 0$ and $\mathrm{num}(m) = 1$.
  **if** $m < 0$ **then**
    **error:** "Exponent must be positive."
  **end if**
  $h \leftarrow s_F$.`space.rational_power`; $n \leftarrow \mathrm{den}(m)$ ▷ Algorithm 19
  $c_0 \leftarrow s_F.C[\mathrm{FIRST\_CIDX}(s_F.space)]$ ▷ normalise by constant mode and Algorithm 21
  $\tilde{s}_F \leftarrow (1/c_0)\, s_F$; $X \leftarrow 1 - \tilde{s}_F$
  $S \leftarrow 1$; $g \leftarrow 1$ ▷ Horner-like accumulation of powers of $X$ and Remark 7.7
  **for** $i = 1, \ldots, h$ **do**
    $S \leftarrow S + \binom{1/n}{i}(-1)^i\, g$
    $g \leftarrow g * X$ ▷ Algorithm 35
  **end for**
  **if** space is non-truncated **then** ▷ Algorithm 19
    $M \leftarrow \|X\|$
    $S.E[\mathrm{FIRST\_EIDX}(s_F.space)]$ += $\left|\binom{1/n}{h+1}\right| M^{h+1}$ ▷ Algorithm 21
  **end if**
  **return** $S/c_0^m$
**end function**

**Implementation of reciprocal.** The reciprocal $a/s_F$ is exposed by overloading `Base.:/` for a scalar $a$ and a `Series`. It is supported for even and complete Fourier spaces, under the assumption that the constant coefficient $c_0$ in the even component is nonzero, so that the normalisation is well conditioned. The method is based on the classical geometric expansion

$$\frac{1}{s_F} = \frac{1}{c_0}\,\frac{1}{1-(1-\tilde{s}_F)} = \frac{1}{c_0}\sum_{i=0}^{\infty}(1-\tilde{s}_F)^i, \qquad \tilde{s}_F = \tfrac{1}{c_0}\, s_F,$$

which is truncated after $h$ terms with a validated remainder bound.
The procedure begins by reading the truncation order $h \coloneqq$ `k_inv` from the space. The constant mode $c_0$ is then extracted (from the even component if $s_F$ is complete, otherwise directly), and used to define $\alpha = 1/c_0$, the normalised series $\tilde{s}_F = \alpha s_F$, and the deviation $X = 1 - \tilde{s}_F$ (see Algorithm 21 for indexing details). The truncated geometric sum

$$P_h = \sum_{i=0}^{h} X^i$$

is built incrementally using a Horner-like accumulation (Remark 7.7): starting with $S = 1$ and $g = X$, the loop repeatedly updates $S \leftarrow S + g$ and $g \leftarrow g \cdot X$ for $h$ steps. If the space is non-truncated, the procedure then appends a validated enclosure to the first error slot of the even component, namely

$$\frac{\|X\|^{h+1}}{1-\|X\|},$$

where $\|X\|$ is the series norm (see Algorithm 28). Finally, the reciprocal is returned in the form

$$\frac{a}{s_F} \;=\; a\,\alpha\,S_h.$$

This procedure is explained in Algorithm 39.

*Remark* 7.8. The remainder estimate in Algorithm 39 requires $\|X\| < 1$, i.e. the normalised deviation $X = 1 - \alpha s_F$ is a contraction in the series norm. This is enforced and bounded via interval arithmetic;

**Algorithm 39** Reciprocal of a Fourier series $s_F$ via geometric expansion

**function** RECIPROCAL_TAYLOR($a$, $s_F$)
  **preconditions:** $s_F$ even or complete; even constant mode $c_0 \neq 0$.
  $h \leftarrow s_F$.`space.inverse_power` ▷ Algorithm 19
  $c_0 \leftarrow \begin{cases} s_F[\texttt{Even}, :C, 0], & \text{if } s_F \text{ is complete} \\ s_F[:C, 0], & \text{if } s_F \text{ is even} \end{cases}$
  $\alpha \leftarrow 1/c_0$;  $\tilde{s}_F \leftarrow \alpha\, s_F$;  $X \leftarrow 1 - \tilde{s}_F$
  $S \leftarrow 1$;  $g \leftarrow X$ ▷ Horner-like accumulation; see Remark 7.7
  **for** $i = 1, \ldots, h$ **do**
    $S \leftarrow S + g$
    $g \leftarrow g * X$ ▷ Algorithm 35
  **end for**
  **if** space is non-truncated **then** ▷ Algorithm 19
    $M \leftarrow \|X\|$ ▷ Algorithm 28
    **if** $s_F$ is complete **then**
      $S[\texttt{Even}, :E, 0]$ += $\dfrac{M^{h+1}}{1-M}$
    **else**
      $S[:E, 0]$ += $\dfrac{M^{h+1}}{1-M}$
    **end if**
  **end if**
  **return** $(a\,\alpha)\, S$
**end function**

when $\|X\|$ is not small enough, one should either increase the truncation order $h$ or switch to an alternative method (e.g. FFT-based, Section 7.4.3.3).

#### 7.4.3.2 Newton–Raphson method

While the Taylor expansion provides a direct polynomial approximation for fractional powers of Fourier series, it is not always the most efficient way to obtain high accuracy. An alternative is to reformulate the problem as a nonlinear equation and apply an iterative root-finding method. For instance, to compute square roots one seeks $x = \sqrt{a}$ for $a > 0$, that is, to solve

$$x^2 = a.$$

This leads naturally to Newton's method. Remarkably, iterative algorithms equivalent to Newton's iteration for this problem were already known to the Babylonians around 1000~BCE: in modern terms, the so-called "Babylonian method" is precisely the Newton–Raphson scheme applied to $f(x) = x^2 - a = 0$.

The iteration begins with an arbitrary positive initial guess $x_1 > 0$, and then constructs a sequence via the update rule

$$x_{n+1} \;=\; \tfrac{1}{2}\left(x_n + \frac{a}{x_n}\right).$$

The mechanism of this update is intuitive. If the current guess $x_n$ is too large, so that $x_n > \sqrt{a}$, then the reciprocal term $a/x_n$ is too small; taking their arithmetic mean pulls the next iterate $x_{n+1}$ closer to $\sqrt{a}$. Conversely, if $x_n$ is too small, then $a/x_n$ is too large, and once again their mean lies between $x_n$ and the true root. In both cases, the iteration drives the sequence toward the square root.

The connection with Newton's method can also be seen explicitly. Setting $f(x) = x^2 - a$, the

Newton–Raphson iteration for solving $f(x) = 0$ is

$$x_{n+1} \;=\; x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - a}{2x_n} = \tfrac{1}{2}\left(x_n + \frac{a}{x_n}\right).$$

Thus the Babylonian update is exactly Newton's method specialised to the square-root problem, and therefore enjoys its characteristic property of quadratic convergence: once the iterates are sufficiently close to the true root, the number of correct digits essentially doubles at each step.

**Convergence proof.** A standard analysis (see e.g. [128]) shows that the Newton iteration for $\sqrt{a}$ converges. The argument has three parts.

1. *Monotonicity.* Suppose $x_n > \sqrt{a}$. Then one checks directly that

   $$a < x_{n+1} < x_n.$$

   Indeed:

   $$x_{n+1} - x_n = \tfrac{1}{2}\left(\frac{a}{x_n} - x_n\right) = \frac{a - x_n^2}{2x_n} < 0,$$

   so the sequence decreases, and

   $$x_{n+1}^2 - a = \tfrac{1}{4}\,(x_n - a/x_n)^2 \;>\; 0,$$

   so each iterate remains strictly larger than $\sqrt{a}$.

2. *Boundedness.* A monotone decreasing sequence that is bounded below must converge (Rudin, Theorem 3.14). If the initial guess happens to satisfy $x_1 < \sqrt{a}$, then the first update $x_2$ lies above $\sqrt{a}$, after which the argument above applies. Hence for all $n \geq 2$, the sequence $\{x_n\}$ is decreasing and bounded below by $\sqrt{a}$.

3. *Identification of the limit.* Let $x = \lim_{n\to\infty} x_n$. Passing to the limit in the iteration

   $$x_{n+1} = \tfrac{1}{2}\left(x_n + \frac{a}{x_n}\right),$$

   we find

   $$x = \tfrac{1}{2}\left(x + \frac{a}{x}\right),$$

   which rearranges to $x^2 = a$. Thus the sequence converges to $\sqrt{a}$.

This argument establishes existence and correctness of the limit, but it does not yet address *how fast* the convergence occurs. The rate of convergence, which turns out to be quadratic, will be analysed separately.

**Example** (Convergence example)**.** To illustrate the method in practice, let us compute the most famous square root of all, $\sqrt{2}$. (According to legend, the Greek who first proved its irrationality was thrown from a cliff by his fellow Pythagoreans.)
Starting with the initial guess $x_1 = 1$, the Newton iteration

$$x_{n+1} = \tfrac{1}{2}\left(x_n + \frac{2}{x_n}\right)$$

produces the following sequence, here displayed with about 60 digits of precision. Correct digits are shown in boldface:

$x_1 = 1,$

$x_2 = 1.5,$

$x_3 = 1.41666666666666666666666666666666666666666666666666666666666675,$

$x_4 = 1.41421568627450980392156862745098039215686274509803921568627450980392156862745,$

$x_5 = 1.41421356237468991062629557889013491011655962211574404458490570,$

$x_6 = 1.41421356237309504880168962350253024361498192577619742849828900,$

$x_7 = 1.41421356237309504880168872420969807856967187537723400156101250,$

$x_8 = 1.41421356237309504880168872420969807856967187537694807317667960.$

Looking carefully, we see that the number of correct digits roughly doubles at each iteration. This remarkable phenomenon reflects the *quadratic convergence* of Newton's method: once the iterates are sufficiently close to the true root, the error essentially squares at each step. Consequently, in only seven iterations we obtain more than 60 correct digits—accuracy which is then limited not by the iteration itself, but by the finite precision of floating-point arithmetic.

**Convergence rate.** We now analyse the convergence quantitatively. Suppose the true solution is $x = \sqrt{a}$, and let $x_n$ be the $n$-th Newton iterate. Define the *relative error*

$$\varepsilon_n \;=\; \frac{x_n - x}{x},$$

so that $x_n = x(1+\varepsilon_n)$. Relative error is convenient because it is dimensionless and directly measures the loss of significant digits: roughly speaking, $\log_{10}(1/|\varepsilon_n|)$ is the number of correct digits in $x_n$.

Substituting $x_n = x(1+\varepsilon_n)$ into the Newton iteration

$$x_{n+1} = \tfrac{1}{2}\left(x_n + \frac{a}{x_n}\right), \qquad a = x^2,$$

and dividing through by $x$, we obtain

$$1 + \varepsilon_{n+1} = \tfrac{1}{2}\left((1+\varepsilon_n) + \frac{1}{1+\varepsilon_n}\right).$$

Expanding $(1+\varepsilon_n)^{-1}$ as a Taylor series,

$$\frac{1}{1+\varepsilon_n} = 1 - \varepsilon_n + \varepsilon_n^2 - \varepsilon_n^3 + \cdots,$$

we find

$$1 + \varepsilon_{n+1} = 1 + \tfrac{1}{2}\,\varepsilon_n^2 + O(\varepsilon_n^3).$$

Thus the errors satisfy

$$\varepsilon_{n+1} = \tfrac{1}{2}\,\varepsilon_n^2 + O(\varepsilon_n^3).$$

This relation shows that once $\varepsilon_n$ is small, the next error is essentially proportional to $\varepsilon_n^2$. In other words, the number of correct digits approximately doubles with each step. This phenomenon is called *quadratic convergence*. It explains the rapid accuracy growth observed in the numerical example: in only a handful of iterations, Newton's method delivers results accurate to machine precision.

**Implementation.** We implement the square root of an even or complete Fourier series $s_F$ by applying Newton's method to the functional equation $S^2 = s_F$. The iteration reads

$$S^{(i+1)} \;=\; \tfrac{1}{2}\big(S^{(i)} + s_F/S^{(i)}\big), \qquad i = 0, 1, \ldots$$

where the division $s_F/S^{(i)}$ is computed with the validated reciprocal routine described in Section 39. The procedure:

- starts from the neutral guess $S^{(0)} = \mathbf{1}$ in the same Fourier space as $s_F$;
- limits the number of Newton steps by the space parameter `k_power`;
- monitors the residual $\|(S^{(i+1)})^2 - s_F\|$ and stops early if its midpoint ceases to decrease;
- in non-truncated spaces, records the final residual as a rigorous enclosure in the first error slot $E[\textsc{first\_eidx}(s_F.space)]$ (see Section 7.3.2.2).

Each multiplication and reciprocal inside the loop is carried out using the validated series arithmetic developed earlier (interval coefficients and analytic tail handling). Thus every iterate $S^{(i)}$ is certified, and the final output is a rigorously validated enclosure of $\sqrt{s_F}$. This procedure is explained in Algorithm 40.

**Algorithm 40** Newton-Raphson square root of a Fourier series $s_F$

**function** SQRT_NEWTON($s_F$)
  **preconditions:** $s_F$ even/complete with a positive constant mode ($\Rightarrow \sqrt{s_F}$ is well-defined).
  $k_{\max} \leftarrow s_F$`.space.rational_power`; $S \leftarrow \mathbf{1}$ ▷ initial guess in the space of $s_F$; Alg. 19
  $\rho_{\text{last}} \leftarrow +\infty$ ▷ tracks residual norm
  **for** $i = 1, \ldots, k_{\max}$ **do**
    $S^{-1} \leftarrow 1/S$ ▷ Algorithm 39
    $S_{\text{new}} \leftarrow \frac{1}{2}\big(S + s_F \cdot S^{-1}\big)$ ▷ Newton update
    $\rho \leftarrow \| S_{\text{new}}^2 - s_F \|$ ▷ Algorithm 28
    **if** $\text{mid}(\rho) > \text{mid}(\rho_{\text{last}})$ **then**
      **break**
    **end if**
    $S \leftarrow S_{\text{new}}$; $\rho_{\text{last}} \leftarrow \rho$
  **end for**
  **if** space is non-truncated **then** ▷ Algorithm 19
    $SE[\textsc{first\_eidx}(s_F.space)]$ += $\rho_{\text{last}}$ ▷ Algorithm 21
  **end if**
  **return** $S$
**end function**

*Remark.* (i) Each multiplication/division in Algorithm 40 invokes the validated Fourier product with analytic tails (Algorithm 35) and the certified series arithmetic, so all iterates remain in $\text{std}(\mathcal{F})$. (ii) The early-stop condition uses midpoint comparison (a heuristic to avoid overshooting); rigour is preserved because the final residual is enclosed and added to the error slot. (iii) An analogous Newton scheme computes a validated reciprocal: starting from $S^{(0)}$, iterate $S^{(i+1)} = S^{(i)}\,(2 - s_F\,S^{(i)})$, then store the terminal residual in $E[\textsc{first\_eidx}(s_F.space)]$.

#### 7.4.3.3 DFFT method

For nonlinear operations on Fourier series, such as $s_F \mapsto \sqrt{s_F}$ or $s_F \mapsto 1/s_F$, it is often advantageous to move back and forth between *coefficient space* and *physical space*. The discrete Fourier transform (DFT) and its inverse provide the bridge: given the first $N$ Fourier coefficients of a $2\pi$-periodic

series object $s_F$, one can compute its values on a uniform grid $\{t_j\}_{j=0}^{M-1}$, with $M$ chosen sufficiently large (typically $M \gtrsim 2N$ to reduce aliasing). The DFFT approach exploits this fact. Starting from the coefficients of $s_F$, an inverse FFT (or DCT in the even case) produces the sampled values $\{s_F(t_j)\}$ on the grid. At this stage the nonlinearity becomes trivial: instead of handling infinite convolutions in coefficient space, one simply applies the analytic map $\Phi$ pointwise to each sample, obtaining $\{\Phi(s_F(t_j))\}$. Finally, a forward FFT/DCT maps these values back to Fourier space, and the result is truncated to the first $N$ modes, yielding a series in the original coefficient format. Symbolically, this amounts to the pipeline

$$s_F \xrightarrow{I_M} \{s_F(t_j)\} \xrightarrow{\Phi} \{\Phi(s_F(t_j))\} \xrightarrow{P_N} \text{truncated coefficients.}$$

The essential intuition is that in coefficient space, nonlinear maps correspond to convolutions of arbitrarily many modes, making rigorous control of aliasing and tails delicate. In contrast, on a grid the map $\Phi$ acts *pointwise*, which is conceptually simpler and computationally efficient. The price to pay is twofold: first, one must oversample sufficiently to control aliasing; second, after applying the nonlinearity, one has to project back to $N$ modes, thereby discarding high frequencies. For analytic data this truncation error is exponentially small in $N$, so the loss is negligible. What remains is to certify that the truncated coefficient vector returned by the DFFT pipeline is close to a true solution of the nonlinear problem (e.g. verifying $S^2 = s_F$ when computing a square root). This final step is achieved by an *a posteriori* validation argument in the spirit of Banach's fixed-point theorem.

**Convergence proof (a posteriori validation).** We illustrate the case of the square root; the reciprocal is analogous. The nonlinear equation of interest is

$$\mathcal{N}(S) = S^2 - s_F = 0.$$

Let $R$ denote the DFFT approximation,

$$R = P_N\big(\sqrt{I_M s_F}\big),$$

where $I_M$ denotes interpolation to an $M$-point grid and $P_N$ is the projection onto the first $N$ Fourier modes (i.e. truncation back to the original series space).
To validate $R$, we introduce the Newton-like operator

$$G(x) = x - A\mathcal{N}(x),$$

where $A$ is a bounded linear operator chosen as an approximate inverse of $D\mathcal{N}(R)$.
The quality of the approximation is quantified by the *defect*

$$r = \frac{N+1}{N}\,\|A\mathcal{N}(R)\|,$$

which defines a candidate ball $B(R, r)$. On this ball, the Lipschitz constant of $DG$ is bounded by

$$K = \sup_{x \in B(R,r)} \|I - A\,D\mathcal{N}(x)\|.$$

If $K \cdot \frac{N+1}{N} < 1$ and $\varepsilon := \|A\mathcal{N}(R)\|$ satisfies $\varepsilon + Kr < r$, then Banach's fixed-point Theorem 7.9 ensures that a unique exact solution $S$ exists inside $B(R, r)$. In other words, the DFFT approximation $R$ is certifiably close to a true square root, with $r$ giving an explicit enclosure radius.

**Theorem 7.9** (Banach fixed point Theorem, [129])**.** *Let $x_0 \in \mathcal{X}$ be an approximate solution of $\mathcal{N}(x) = 0$ in a Banach space $\mathcal{X}$. Assume there exists a bounded linear operator $A : \mathcal{X} \to \mathcal{X}$ such that:*

1. *the defect is small:*
$$r \;=\; \frac{N+1}{N}\,\|A\mathcal{N}(x_0)\| \quad \textit{is finite};$$

2. *the Lipschitz bound holds: for all* $x \in B(x_0, r)$,
$$K = \sup_{h\neq 0} \frac{\|h - A(D\mathcal{N}(x)h)\|}{\|h\|} \quad \textit{satisfies} \quad K \cdot \tfrac{N+1}{N} < 1;$$

*then there exists a unique* $x \in B(x_0, r)$ *with* $\mathcal{N}(x) = 0$.

Thus, in our setting the DFFT output $R$ plays the role of the approximate solution $x_0$, and the theorem guarantees that $R$ can be rigorously validated as being close to a true root of $\mathcal{N}(S) = 0$, with uniqueness ensured within the enclosing ball $B(R, r)$.

**Example** (Convergence example)**.** Take
$$s_F(t) \;=\; 1 + \varepsilon\cos t, \qquad 0 < \varepsilon < 1,$$
and consider the nonlinear map $\Phi(z) = \sqrt{z}$. Choose $N$ and $M$ with $M \geq 2N + 1$. The DFFT pipeline produces the approximation
$$R \;=\; P_N\Big(\sqrt{I_M s_F}\Big).$$
Since $s_F$ is analytic on a strip and bounded away from 0 (indeed $1 - \varepsilon > 0$), the composition $\Phi \circ s_F$ is also analytic on a slightly narrower strip. Its Fourier coefficients therefore decay exponentially, so the $N$-term projection $P_N$ incurs an error of order $O(e^{-\sigma N})$ for some $\sigma > 0$.
The *residual*
$$\|\mathcal{N}(R)\| \;=\; \|R^2 - s_F\|$$
is accordingly small. With $A \approx (2R)^{-1}$ one obtains a defect $\varepsilon = \|A\mathcal{N}(R)\| = O(e^{-\sigma N})$. A bound $K < 1$ follows from continuity of $D\mathcal{N}$ on a small ball around $R$. Hence the Banach fixed-point test succeeds, yielding an explicit enclosure radius $r = O(e^{-\sigma N})$ and a validated solution $S$ of $\mathcal{N}(S) = 0$.

**Convergence rate.** The DFFT approximation $R$ inherits the exponential convergence typical of spectral methods applied to analytic data. Indeed, if $s_F$ extends analytically to a complex strip $\{\, t \in \mathbb{C} : |\Im t| < \sigma \,\}$ and is bounded away from 0, then $\Phi \circ s_F$ is also analytic on a (slightly smaller) strip. By classical results on Fourier approximation, the Fourier coefficients of $\Phi \circ s_F$ decay like $O(e^{-\sigma' n})$ for some $\sigma' > 0$. Consequently the projection error
$$\|\Phi \circ s_F - P_N(\Phi \circ s_F)\| \;=\; O(e^{-\sigma' N}).$$
Since the DFFT pipeline computes $R = P_N(\Phi(I_M s_F))$, the oversampling step with $M \gtrsim 2N$ ensures that aliasing errors are also exponentially small. Therefore the total error of the DFFT method satisfies
$$\|R - \Phi \circ s_F\| \;=\; O(e^{-\sigma' N}),$$
with $\sigma'$ determined by the analyticity strip width of $s_F$. In turn, the a posteriori validation argument shows that the exact solution $S$ lies within an explicit ball $B(R, r)$ of radius $r = O(e^{-\sigma' N})$, so the validated convergence rate is exponential in $N$.

*Remark.* In the DFFT approach, several practical and theoretical aspects must be taken into account. First, the choice of transform depends on the symmetry of the series: cosine transforms (DCT) are used for even series, while complete series are treated with FFT on complex modes. In both cases the mathematics is identical, amounting to a unitary (up to normalisation) mapping to grid space, pointwise application of the nonlinearity, and projection back to the coefficient domain.

Second, nonlinear maps inevitably generate higher frequencies. To avoid contamination of the retained modes, one employs oversampling (typically $M \gtrsim 2N$) and, in the case of polynomial nonlinearities, the standard 2/3-dealiasing rule. This ensures that aliasing errors are suppressed to a level below the intrinsic exponential truncation error of analytic series. Third, for nonlinearities such as $\Phi(z) = \sqrt{z}$ or $\Phi(z) = 1/z$, it is essential to guarantee that the range of $s_F$ remains within a compact subset avoiding the branch cut or the origin. This condition is not assumed a priori but is instead checked *a posteriori* during the validation stage. Finally, unlike Taylor or Newton expansions where truncation errors can be controlled directly from the size of $\|S\|$, the DFFT strategy relies on an *a posteriori* argument: the defect $\varepsilon = \|A\mathcal{N}(R)\|$ together with a Lipschitz bound $K$ for $DG$ certify, via Theorem 7.9, the existence and uniqueness of a true solution near the numerical approximation $R$.

**Implementation.** We describe the four routines used by the DFFT method: coefficient→grid described in TO_POINTS_FFT$(\cdot)$ (Algorithm 41), grid→coefficient described in TO_SERIES_FFT!$(\cdot)$ (Algorithm 42), a posteriori validation described in COMPUTE_FFT_ERROR!$(\cdot)$ (Algorithm 44) and the end-user square root described in SQRT_FFT$(\cdot)$ (Algorithm 45). Two cases are handled uniformly but with different transforms and normalisations: *even* series use a real cosine transform (DCT–type), while *complete* seriesuse a real/complex FFT that mixes cosine/sine parts.

*Remark.* If the scalar type is an interval, grid evaluation uses the midpoint of the stored coefficients to produce a single floating grid signal for the nonlinear pointwise operation; the missing rigour is recovered by the *a posteriori* validation step (Section 7.4.3.3) which attaches a certified enclosure to the returned series.

**Algorithm 41** Coefficients → grid values

**function** TO_POINTS_FFT_COMPLETE($s_F$)
  **Input:** $s_F = (s_F[\texttt{Even}], s_F[\texttt{Odd}])$ on a complete Fourier space ($N$ modes, grid $M$).
  Build a complex frequency vector $c \in \mathbb{C}^M$:
  - fill $c_0, \dots, c_N$ from the even coefficients;
  - add $i$ times the odd coefficients into $c_1, \dots, c_N$;
  - adjust the zero mode according to the transform normalisation.

  Apply the inverse real FFT to $c$ and rescale to obtain the grid signal

$$\{t_j\}_{j=0}^{M-1} \quad \longmapsto \quad S_j \approx s_F(t_j).$$

  **return** $S = (S_0, \dots, S_{M-1})$.
**end function**
**function** TO_POINTS_FFT_EVEN($s_F$)
  **Input:** $s_F$ on an even Fourier space with $N$ modes and grid size $M$.
  Form the real cosine spectrum from $s_F$, scaling the constant term per DCT convention.
  Apply the inverse cosine transform and rescale to obtain $S_j \approx s_F(t_j)$.
  **return** $S = (S_0, \dots, S_{M-1})$.
**end function**

*Remark.* (i) For interval coefficients, use midpoints when building $c$ or the cosine spectrum. (ii) The exact multiplicative constants on the constant term and on the global scaling are those dictated by the chosen FFT/DCT conventions; these are matched consistently in the inverse routine below. (iii) In practice, the forward and inverse FFT/DCT transforms are implemented using the `FFTW.jl` package in `Julia`, which provides bindings to the highly optimised FFTW library [130].

*Remark.* The pair TO_POINTS_FFT / TO_SERIES_FFT! uses consistent normalisations so that, for a bandlimited trigonometric polynomial $p$ with at most $N$ modes, the round-trip $p \mapsto s_F \mapsto p$ is

**Algorithm 42** Grid values → coefficients

**function** To_Series_FFT_Complete($S$, $s_F$)
  **Input:** empty target series $S$ on a complete space; grid $s_F \in \mathbb{S}^M$.
  Compute the real FFT of $s_F$ and rescale by $1/M$.
  Extract the first $N$ cosine modes into $S[\texttt{Even}].C$, scaling the constant term accordingly.
  Extract the first $N$ sine modes into $S[\texttt{Odd}].C$, scaling the constant term as in Algorithm 41.
  **return** $S$.
**end function**
**function** To_Series_FFT_Even($S$, $s_F$)
  **Input:** empty target series $S$ on an even space; grid $s_F \in \mathbb{S}^M$.
  Apply the forward cosine transform to $s_F$, rescaling by the inverse factor from Algorithm 41.
  Store the first $N{+}1$ cosine coefficients in $S.C$, scaling the constant term.
  **return** $S$.
**end function**

exact up to floating error and truncation to $N$. In practice, the transforms are implemented via the real-to-complex routines of the `FFTW.jl` package in `Julia`, which provide bindings to the highly optimised FFTW library [130].

**A posteriori validation.** Following the Banach fixed-point Theorem 7.9, given a nonlinear map $\mathcal{N}$ on the series space (e.g. $\mathcal{N}(S) = S^2 - s_F$ for a square root) and an *a priori* approximation $R$, we certify the existence of a true solution $S$ near $R$ by constructing a Newton-like fixed-point operator

$$G(x) = x - A\mathcal{N}(x),$$

where $A$ is a bounded linear preconditioner approximating $(D\mathcal{N}(R))^{-1}$ (for the square root, $A \approx (2R)^{-1}$). We then bound

$$\varepsilon := \|A\mathcal{N}(R)\|, \qquad K := \sup_{x \in B(R,r)} \|I - A\,D\mathcal{N}(x)\|.$$

If there exists $r > 0$ with $K < 1$ and $\varepsilon + Kr < r$, the Banach fixed-point theorem guarantees a unique solution $S \in B(R, r)$ with $\mathcal{N}(S) = 0$. The ball $B(R, r)$ is represented concretely in Algorithm 43.

**Algorithm 43** Ball around a Fourier series $s_F$

**function** Ball($s_F$, $r$)
  **Input:** Fourier series $s_F$, radius $r > 0$.
  Copy the error vector $E \leftarrow s_F.E$.
  Increase the leading error entry:
  $E[\text{first\_eidx}(s_F.space)] \leftarrow E[\text{first\_eidx}(s_F.space)] + r$ ▷ Algorithm 21
  **return** new series with space $s_F.space$, coefficients $s_F.C$, and updated error vector $E$.
**end function**

In other words, $B(s_F, r)$ is encoded by enlarging the first error slot of $s_F$, which uniformly encloses a ball of radius $r$ around the approximation. This compact representation integrates seamlessly with the validated arithmetic on Fourier series.
We can now explain how the DFFT error is certified. Algorithm 44 computes a concrete radius $r$ and Lipschitz constant $K$, and records $r$ in the first error slot of the output.

*Remark.* (i) The $(N{+}1)/N$ factor accounts for the projection/truncation from grid back to $N$ modes. (ii) For square roots, $\mathcal{N}(S) = S^2 - s_F$ and $D\mathcal{N}(S)[v] = 2Sv$. (iii) The resulting $r$ gives an explicit, machine-checkable enclosure certifying that $R$ is within distance $r$ of a true solution.

**Algorithm 44** Computing the error for the DFFT

**function** COMPUTE_FFT_ERROR!($R$, $s_F$, $\mathcal{N}$, $D\mathcal{N}$)
  **Input:** provisional solution $R$, data $s_F$, residual map $\mathcal{N}$, derivative $D\mathcal{N}$.
  Build a bounded preconditioner $A \approx (D\mathcal{N}(R))^{-1}$ on the working space.
  Define the Newton operator $G(x) = x - A\mathcal{N}(x)$ and its derivative $DG(x) = I - A\,D\mathcal{N}(x)$.
  Set the residual radius $r \leftarrow \|A\mathcal{N}(R)\| \cdot \frac{N+1}{N}$ ▷ projection factor, Algorithm 28
  Construct the BALL($B(R,r)$) around $R$ ▷ Algorithm 43
  Compute $K \leftarrow \sup_{x \in B(R,r)} \|DG(x)\|$. ▷ Algorithm 28
  **if** $K \dfrac{N+1}{N} > 1$ **then**
    **error:** "Validation failed (insufficient contraction)".
  **end if**
  Store $r$ in the first error slot of $R$ (according to parity layout).
  **return** $R$
**end function**

Using the Banach-based validation scheme and the error estimation routine of Algorithm 44, Algorithm 45 implements a complete DFFT-based procedure that computes the square root of a Fourier series and embeds the corresponding rigorous validation directly into the transform pipeline.

**Algorithm 45** SQRT_FFT: DFFT square root with validation

**function** SQRT_FFT($s_F$)
  **Input:** series $s_F$ in an even or complete Fourier space with grid size $M$.
  $p \leftarrow$ TO_POINTS_FFT($s_F$) ▷ Algorithm 41
  $p \leftarrow \sqrt{p}$ ▷ apply nonlinearity pointwise on the grid
  $R \leftarrow$ zero series in the same space as $s_F$ ▷ Section 7.3.2.3
  TO_SERIES_FFT!($R$, $p$) ▷ Algorithm 42
  **if** space is non-truncated **then** ▷ Algorithm 19
    Define $\mathcal{N}(S) = S^2 - s_F$ and $D\mathcal{N}(S)[v] = 2Sv$
    COMPUTE_FFT_ERROR!($R$, $s_F$, $\mathcal{N}$, $D\mathcal{N}$) ▷ Algorithm 44
  **end if**
  **return** $R$
**end function**

*Remark.* (i) For even series, the grid transforms use cosine (DCT) conventions with appropriate scaling of the constant coefficient; for complete series, real FFTs split naturally into cosine/sine blocks. (ii) Nonlinearities generate high frequencies; choosing $M \gtrsim 2N$ (and standard 2/3 de-aliasing for polynomial maps) suppresses aliasing below the truncation tail. (iii) For $\sqrt{\cdot}$ or $1/\cdot$, one must avoid branch cuts and zeros; this is enforced *a posteriori* by the validation step, which fails fast if the contraction conditions are not met.

**Conclusion.** The three strategies (Taylor expansion, Newton-Raphson iteration, and DFFT-based projection) offer complementary tools for handling nonlinear operations such as square roots and reciprocals of Fourier series. The Taylor method is explicit, lightweight, and effective when the series is well normalised around its constant term, with the binomial truncation providing immediate error bounds. The Newton-Raphson method, by contrast, delivers rapid (quadratic) convergence once an accurate initial guess is available, but requires a careful safeguard on the number of iterations to guarantee termination and validated bounds. The DFFT approach is the most flexible, allowing nonlinearities to be imposed pointwise in physical space, with residual validation ensuring rigour; however, it comes with the overhead of oversampling, projection, and the need for *a posteriori* analysis. In practice, no single technique is sufficient for all problems.

Depending on the function under study and the analytic estimates available, some methods yield convergent enclosures while others may fail to do so. For this reason it is important to develop, implement, and combine multiple approaches, so that a suitable tool is available in each context of a computer-assisted proof.

### 7.4.4 Antiderivative

From a mathematical point of view, taking antiderivatives of Fourier series is straightforward. Given a Fourier expansion $F(t) = \sum_{k \in K} c_k \, \phi_k(t)$, where $\phi_k$ denotes the standard trigonometric modes (sines in the odd case, cosines in the even case, or both in the complete case). Its $n$-th antiderivative is obtained by dividing each Fourier mode by $k^n$ and switching between sine/cosine according to the usual parity rules. Concretely,

$$\int \sin(kt)\, dt = -\frac{\cos(kt)}{k}, \qquad \int \cos(kt)\, dt = \frac{\sin(kt)}{k},$$

and iterating this $n$ times alternates parity with a sign that repeats with period 4. In our implementation this global sign is encoded as

$$s = \begin{cases} -1, & n \bmod 4 \in \{1, 2\}, \\ 1, & n \bmod 4 \in \{0, 3\}, \end{cases}$$

and each stored coefficient is mapped to its antiderivative by

$$c_k \longmapsto \frac{s\, c_k}{k^n}.$$

In the rigorous setting, enclosure entries are transformed in exactly the same way: each error slot is divided by $k^n$, so that tail bounds remain valid under integration.
If $F$ is *even* or *odd*, the rule above applies directly. If $F$ is *complete* (i.e. it carries both even and odd parts), we integrate the two parity components separately and then recombine them into a complete result, placing each integrated component into the even/odd block dictated by its *output* parity after $n$ integrations.

**Result-space constructor for antiderivatives.** Taking $n$ antiderivatives flips parity $n$ times: an even (resp. odd) series object becomes odd (resp. even) when $n$ is odd, and returns to its original parity when $n$ is even. The helper routine creates the appropriate target Fourier space by preserving all numeric and algorithmic parameters ($N$, $R$, coefficient type, truncation flag, `rational_power`, `inverse_power` and updating only the parity (Algorithm 19):

$$\text{parity}_{\text{out}} = \text{Parity}\big(\,(\,\text{Integer}(\text{parity}_{\text{in}}) + n\,) \bmod 2\big).$$

Algorithm 46 implements the parity rule above by constructing the Fourier space corresponding to the $n$-th antiderivative, preserving all numerical parameters while updating only the parity.
When the input series object $s_F$ lives in an *even* or *odd* Fourier space, the antiderivative is computed directly by dividing coefficients and errors by $k^n$ and applying the appropriate parity-dependent sign. Algorithm 47 performs the actual coefficient-wise integration when the input series lives in an even or odd Fourier space, applying the appropriate sign and $k^{-n}$ scaling to both coefficients and error bounds.
If the series object is *complete* (containing both even and odd parts), then each component is integrated separately using the parity routine above, and the results are recombined into a complete Fourier series. Algorithm 48 extends the above routine to complete Fourier spaces by integrating the even and odd components separately and then recombining them into a validated complete series.

**Algorithm 46** Construct target space for the $n$-th antiderivative

**function** GET_SPACE_ANTIDERIVATIVE($s$, $n$)
  **preconditions:** $n \geq 1$; $s$ a Fourier space (even/odd/complete).
  **if** $n < 1$ **then**
    **error:** "Order of antiderivative must be positive."
  **end if**
  $P_{\text{in}} \leftarrow \text{PARITY}(s)$
  $P_{\text{out}} \leftarrow \text{Parity}\big((\text{Integer}(P_{\text{in}}) + n) \bmod 2\big)$
  **return** Fourier space with same parameters but parity $P_{\text{out}}$.
**end function**

**Algorithm 47** $n$-th antiderivative on an even/odd Fourier space $s_F$

**function** ANTIDERIVATIVE_PARITY($s_F$, $n$)
  **preconditions:** $s_F$ lives in an even *or* odd Fourier space; $n \geq 1$.
  $S \leftarrow$ zero series in GET_SPACE_ANTIDERIVATIVE($s_F.space$, $n$) ▷ Alg. 46, Sec. 7.3.2.3
  $s \leftarrow \begin{cases} -1 & \text{if } n \bmod 4 \in \{1, 2\}, \\ 1 & \text{if } n \bmod 4 \in \{0, 3\} \end{cases}$
  **for** each coefficient frequency $k \geq 1$ **do**
    $S[:C, k] \leftarrow s \cdot s_F[:C, k]/k^n$
  **end for**
  **if** $s_F.space$ is non-truncated **then** ▷ Algorithm 19
    **for** each error index $k \geq 1$ **do**
      $S[:E, k] \leftarrow s_F[:E, k]/k^n$
    **end for**
  **end if**
  **return** $S$
**end function**

**Algorithm 48** $n$-th antiderivative on a complete Fourier space $s_F$

**function** ANTIDERIVATIVE_COMPLETE($s_F$, $n$)
  **preconditions:** $s_F$ lives in a complete Fourier space; $n \geq 1$.
  $r_{\text{even}} \leftarrow$ ANTIDERIVATIVE_PARITY($s_F$[`Even`], $n$) ▷ Algorithm 47
  $r_{\text{odd}} \leftarrow$ ANTIDERIVATIVE_PARITY($s_F$[`Odd`], $n$) ▷ Algorithm 47
  $S \leftarrow$ zero series in $s_F.space$ ▷ Section 7.3.2.3
  **if** $r_{\text{even}}.space$ is odd **then**
    $S$[`Odd`] $\leftarrow r_{\text{even}}$
  **else**
    $S$[`Even`] $\leftarrow r_{\text{even}}$
  **end if**
  **if** $r_{\text{odd}}.space$ is odd **then**
    $S$[`Odd`] += $r_{\text{odd}}$
  **else**
    $S$[`Even`] += $r_{\text{odd}}$
  **end if**
  **return** $S$
**end function**

*Remark.* The sign pattern is periodic with period 4 in $n$, reflecting the sine/cosine alternation under repeated integration. For complete series, the two independently integrated components are placed into the even/odd blocks according to their *output* parity. Error enclosures are transformed

in lockstep with coefficients by division by $k^n$, thus preserving rigorous tail bounds.

## 7.5 Vector-valued series

**Definition 7.10.** Given $\rho > 0$, define the vector space

$$\mathcal{F}_\rho^d = \left\{ \mathbf{s_F} = (s_F^{(1)}, \dots, s_F^{(d)}) \ : \ s_F^{(i)} \in \mathcal{F}_\rho \right\}.$$

For $\mathbf{s_F} \in \mathcal{F}_\rho^d$, the norm is given by

$$\|\mathbf{s_F}\|_\rho \ = \ \max_{1 \le i \le d} \|s_F^{(i)}\|_\rho. \tag{7.4}$$

The space $(\mathcal{F}_\rho^d, \|\cdot\|_\rho)$ inherits all the properties of the scalar space: each component $s_F^{(i)}$ is a $2\pi$-periodic analytic function with exponentially decaying Fourier coefficients, and the vector norm controls the analytic regularity uniformly across components. In particular, $\mathcal{F}_\rho^d$ is a Banach algebra under componentwise multiplication. Differentiation and integration act componentwise as well, and their effect on the norm is tracked exactly as in the scalar case.

**A computable surrogate.** As before, $\mathcal{F}_\rho^d$ is not directly accessible to computation. We therefore introduce a surrogate $\mathrm{std}(\mathcal{F}_\rho^d)$ defined by replacing Fourier coefficients with interval enclosures. Concretely, for odd vector-valued functions truncated at index $n$, we set

$$U^{(i)} = (U_1^{(i)}, \dots, U_n^{(i)}) \in \mathrm{std}(\mathbb{R}^n), \qquad V^{(i)} = (V_n^{(i)}, V_{n+1}^{(i)}, \dots) \in \mathrm{std}(\mathbb{R}^{\mathbb{N}}),$$

for each component $i = 1, \dots, d$. The surrogate set determined by $(U^{(i)}, V^{(i)})_{i=1}^d$ consists of all functions

$$s_F^{(i)}(t) = \sum_{k=1}^{n} u_k^{(i)} \sin(kt) + \sum_{m=n}^{\infty} v_m^{(i)} \sin(mt), \qquad u_k^{(i)} \in U_k^{(i)}, \ \ v_m^{(i)} \in V_m^{(i)}.$$

Even components are treated analogously with cosine modes, and general functions decompose into even and odd parts. This yields a computationally effective Banach surrogate $\mathrm{std}(\mathcal{F}_\rho^d)$ in which addition, multiplication, differentiation, and integration are all rigorously tracked by interval arithmetic at the vector level.

### 7.5.1 Implementation

Vector-valued series are realised by extending the scalar infrastructure to tuples of series spaces. Concretely, a *vector series space* of dimension $d$ is defined as an ordered $d$-tuple of scalar series spaces, all sharing the same coefficient type and truncation flag. This allows one to represent $\mathbf{s_F}(t) \in \mathcal{F}_\rho^d$ as a single object while preserving access to each component $s_F^{(i)}$.

**Vector series spaces.** The type `VectorSeriesSpace{T,d,CT,ET}` encodes the structure of such a space. Before introducing the full structure, we recall that the length of a *scalar* series space depends on its Fourier parity:

$$\textsc{get\_component\_length}(s) \ = \ \begin{cases} 1 & \text{if } s \text{ is even,} \\ 1 & \text{if } s \text{ is odd,} \\ 2 & \text{if } s \text{ is complete.} \end{cases}$$

Intuitively, this counts how many independent coefficient blocks are carried by the space. For a vector-valued series $\mathbf{s_F} \in \mathcal{F}_\rho^d$, the overall size is obtained by summing the scalar contributions from each component, as summarised in Algorithm 49.

**Algorithm 49** Vector components

1: **function** GET__COMPONENT__LENGTH($S$)
2: **return** $\sum_{i=1}^{d}$ GET__COMPONENT__LENGTH($S_i$)
3: **end function**

With this helper in place, the full structure of a vector series space is summarised in Algorithm 50.

**Algorithm 50** Vector series space structure

**Type:** `VectorSeriesSpace{T,d,CT,ET}`
Parameters:

- $T$: truncation flag (Boolean),
- $d$: dimension of the vector space,
- $CT$: coefficient type,
- $ET$: error type.

Fields:

- `space :: Space` (tuple of $d$ component series spaces),
- `d :: Int` (number of components),
- `Cstart :: Vector{Int}` (cumulative offsets for coefficients),
- `Estart :: Vector{Int}` (cumulative offsets for errors),
- `Cmap :: Vector{Tuple{Int, Int}}` (mapping from global coefficient indices to component-local indices).

1: **function** VECTORSERIESSPACE($S_1, \ldots, S_d$)
2: **if** $d < 1$ **then**
3: **error:** "Dimension must be positive."
4: **end if**
5: Set `Cstart`, `Estart` as cumulative sums of N__COEFFS($S_i$) and N__ERRORS($S_i$). ▷ Alg. 20
6: Build `Cmap` to record, for each global coefficient index, the pair (component, local index).
7: Compute overall size using GET__COMPONENT__LENGTH($S$). ▷ Alg. 49
8: **return** new `VectorSeriesSpace` with fields $(S, d, Cstart, Estart, Cmap)$
9: **end function**

The arrays `Cstart` and `Estart` provide fast access to the block of coefficients and errors associated with each component $s_F^{(i)}$. The mapping `Cmap` is used to recover, from a global index, both the component number $i$ and the local index within $s_F^{(i)}$.

**Size helpers.** Global sizes are obtained by summing the contributions of the component spaces, consistently with Algorithm 20:

$$\text{N\_\_COEFFS}(\mathbf{s_F}) = \sum_{i=1}^{d} \text{N\_\_COEFFS}(s_F^{(i)}), \qquad \text{N\_\_ERRORS}(\mathbf{s_F}) = \sum_{i=1}^{d} \text{N\_\_ERRORS}(s_F^{(i)}).$$

More generally, each function defined for scalar spaces (such as GET__SUBSPACE__INDICES($\cdot$) in Algorithm 22) is lifted to vector spaces by applying it componentwise and shifting indices by the appropriate offsets stored in `Cstart` and `Estart`.

**Truncation.** The helper TRUNCATE_SPACE$(\cdot)$ applies truncation componentwise using Algorithm 18, producing a truncated vector space with the same structure but where each component $s_F^{(i)}$ is truncated. This ensures that vector spaces inherit the same rigorous semantics as their scalar counterparts.

**Utilities.** Two generic functions are overloaded for convenience. First, `Base.copy(S::ST)` ensures that vector series spaces can be safely duplicated: the new object shares no memory with the original, but preserves all structural information (`S`, `d`, `Cstart`, `Estart`, `Cmap`). Second, the length of a vector space is defined as its dimension, so that `Base.length(space::VectorSeriesSpace)` simply returns $d$. These overloads allow vector series spaces to integrate naturally with `Julia`'s standard library, supporting generic algorithms that rely on copying or dimension queries.

*Remark.* The design of `VectorSeriesSpace` mirrors that of scalar Fourier spaces, but with additional bookkeeping to manage offsets across components $s_F^{(i)}$. This ensures that:

1. All scalar operations (addition, scalar multiplication, norms) lift transparently to the vector case by applying them componentwise.
2. Indexing into global coefficient or error arrays is unambiguous, thanks to the cumulative offset vectors `Cstart` and `Estart`.
3. Compatibility checks enforce structural consistency, which is crucial for rigorous computer-assisted proofs involving systems of functional equations.

**Canonical basis.** Let $\{e_k^{(i)}\}_{i=1,\dots,d;\, k\in K}$ denote the canonical basis of $\mathcal{F}_\rho^d$, defined by

$$e_k^{(i)}(t) \;=\; \big(0,\dots,0,\ \phi_k(t),\ 0,\dots,0\big),$$

where $\phi_k$ occupies the $i$-th component and all other components are zero. Every $\mathbf{s_F} \in \mathcal{F}_\rho^d$ admits the (unique) expansion

$$\mathbf{s_F}(t) \;=\; \sum_{i=1}^{d} \sum_{k\in K} c_k^{(i)}\, e_k^{(i)}(t).$$

For finite index sets $K_n \subset K$ (e.g. a truncation level), the family $\{e_k^{(i)} : i = 1,\dots,d,\ k \in K_n\}$ spans the finite-dimensional subspace $\mathcal{F}_{\rho,n}^d$.
Before materialising vector basis elements, we extend the scalar stopping logic to the vector case by summing, across components, the admissible coefficient slots (with optional cap $max\_k$) and the tail slots determined by parity. This yields the vector-level stopping rule of Algorithm 51.

**Algorithm 51** Stopping rule for vector series bases

**function** STOP_BASIS_ITERATOR$(S, k, max_k)$
  **if** $max\_k = 0$ **then**
    **return** $k >$ N_COEFFS$(\mathbf{s_F}) + \sum_{i=1}^{d}$ N_ERRORS_BASIS$(s_F^{(i)})$ ▷ Alg. 20, 30
  **else**
    **return** $k > \sum_{i=1}^{d}$ N_COEFFS$(s_F^{(i)}, max_k) + \sum_{i=1}^{d}$ N_ERRORS_BASIS$(s_F^{(i)})$ ▷ Alg. 20, 30
  **end if**
**end function**

Given the stopping rule, we materialise the $k$-th vector basis element by scanning components until the relevant block (coefficients or tails) is reached, then delegating to the scalar `basis!` inside that component $s_F^{(i)}$.
With the stopping rule (Algorithm 51) and the materialisation routine (Algorithm 52) in place, we can expose a full iterator over the canonical basis of a vector space. At each step, the iterator

**Algorithm 52** Materialise the $k$-th vector basis element

```
1:  function BASIS!(B, k, max_k)                          ▷ B zero in s_F ← B.space
2:      c ← 0                                              ▷ component counter
3:      s ← 0                                              ▷ cumulative slot counter
4:      errors ← false
5:      while s < k do
6:          c ← c + 1;
7:          if c > d then
8:              c ← 1; errors ← true
9:          end if
10:         if ¬ errors then
11:             s ← s + N_COEFFS(s_F^(c), max_k)           ▷ Algorithm 20
12:         else
13:             s ← s + N_ERRORS_BASIS(s_F^(c))            ▷ Algorithm 30
14:         end if
15:     end while
16:     if ¬ errors then
17:         ℓ ← N_COEFFS(s_F^(c), max_k)                   ▷ Algorithm 20
18:         BASIS!(B[c], k − (s − ℓ), max_k)               ▷ delegate to scalar basis in s_F^(c)
19:     else
20:         ℓ ← N_ERRORS_BASIS(s_F^(c))                    ▷ Algorithm 30
21:         BASIS!(B[c], k − (s − ℓ), max_k)               ▷ activate appropriate tail slot in s_F^(c)
22:     end if
23:     return B
24: end function
```

returns the next basis element and advances the state counter, until all admissible indices for $\mathbf{s_F} \in \mathcal{F}_\rho^d$ have been exhausted.

**Algorithm 53** Iteration over a vector basis

```
1:  function ITERATE(S, state = 1)                        ▷ S is a series in a vector space F_ρ^d
2:      if state > LENGTH(S.space) then
3:          return nothing                                 ▷ all components s_F^(i) exhausted
4:      end if
5:      F ← S[state]                                       ▷ Algorithm 52
6:      next_state ← state + 1
7:      return (F, next_state)
8:  end function
```

### 7.5.2 Basic operations in a vector series space

For $\alpha \in \mathbb{R}$ and $\mathbf{s_{1,F}}, \mathbf{s_{2,F}} \in \mathcal{F}_\rho^d$,

$$(\mathbf{s_{1,F}} \pm \mathbf{s_{2,F}})^{(i)} = s_{1,F}^{(i)} \pm s_{2,F}^{(i)}, \qquad (\alpha\, \mathbf{s_{1,F}})^{(i)} = \alpha\, s_{1,F}^{(i)}.$$

All coefficient operations are performed with interval arithmetic, so enclosures are preserved. Operators defined on $\mathcal{F}_\rho$ (e.g. integration, differentiation) lift to $\mathcal{F}_\rho^d$ by acting on each component $s_{1,F}^{(i)}$ independently.

Addition of vector-valued series requires that their ambient spaces be compatible, i.e. they must have the same dimension and their corresponding component spaces must be compatible. This

is checked by the routine CHECK_COMPATIBLE_SPACES_SUM$(\cdot)$, and the resulting sum space is constructed componentwise, as summarised in Algorithm 54.

**Algorithm 54** Compatibility and sum of vector series spaces

```
function CHECK_COMPATIBLE_SPACES_SUM(S_1, S_2)
    if length(S_1) ≠ length(S_2) then return false
    end if
    for i = 1, ..., d do
        if CHECK_COMPATIBLE_SPACES_SUM(S_1.S[i], S_2.S[i]) = false then
            return false
        end if
    end for
    return true
end function
function GET_SPACE_SUM(S_1, S_2)
    if CHECK_COMPATIBLE_SPACES_SUM(S_1, S_2) = false then
        error: "Vector spaces not compatible for sum."
    end if
    if S_1 = S_2 then
        return S_1
    end if
    Construct componentwise sums S_i^* = GET_SPACE_SUM(S_1.S[i], S_2.S[i])
    return new vector series space from (S_1^*, ..., S_d^*).
end function
```

Once the target space has been constructed by Algorithm 54, the actual addition (or subtraction) proceeds independently on each component $s_{1,F}^{(i)}$ and $s_{2,F}^{(i)}$, using Algorithm 25. At the scalar level, this is precisely the coefficientwise procedure of Algorithm 27, already introduced for single series. Thus, vector addition is nothing more than applying the scalar algorithm componentwise, with all interval bounds propagated as before.

#### 7.5.2.1 Norm of a vector series space

Recall from Definition 7.10 that $(\mathcal{F}_\rho^d, \|\cdot\|_\rho)$ is equipped with the product (max) norm

$$\|\mathbf{s_F}\|_\rho \;=\; \max_{1\le i\le d} \|s_F^{(i)}\|_\rho,$$

where $\|\cdot\|_\rho$ on each component is the scalar norm of Equation (7.2). This choice is natural for validated numerics: it preserves the componentwise enclosure philosophy and satisfies the standard properties (positivity, homogeneity, triangle inequality). Moreover, the max-norm is compatible with the canonical basis in the sense that

$$\|e_k^{(i)}\|_\rho \;=\; \|\phi_k\|_\rho,$$

where $\phi_k$ denotes the $k$-th scalar trigonometric basis element of $\mathcal{F}_\rho$ (either $\sin(kt)$ or $\cos(kt)$, depending on parity).

**Implementation.** At the code level, the norm of a vector-valued series $\mathbf{s_F}$ is obtained by computing the norm of each scalar component $s_F^{(i)}$ (using Algorithm 28) and returning their maximum, as it can be seen in Algorithm 55. This directly realises Definition 7.10.

**Algorithm 55** Norm of a vector-valued series $\mathbf{s_F}$

```
1: function NORM(s_F)
2:     n ← 0
3:     for i = 1, ..., s_F.space.d do
4:         n ← max(n, NORM(s_F^(i)))                    ▷ Algorithm 28
5:     end for
6:     return n
7: end function
```

#### 7.5.2.2 Balls of vector series spaces

For $\mathbf{s_{0,F}} \in \mathcal{F}_\rho^d$ and $r > 0$, we define the closed ball

$$B(\mathbf{s_{0,F}}, r) \;=\; \left\{ \mathbf{s_F} \in \mathcal{F}_\rho^d : \|\mathbf{s_F} - \mathbf{s_{0,F}}\|_\rho \le r \right\},$$

where $\|\cdot\|_\rho$ is the max-norm from Definition 7.10. Such balls are the natural neighbourhoods used in the Banach fixed point framework: if a map is contractive on $B(\mathbf{s_{0,F}}, r)$, then a unique fixed point lies inside.

**Implementation.** In addition to the norm, a common operation is to form the *ball* of radius $r > 0$ centred at a vector-valued series $\mathbf{s_F}$. This operation is performed componentwise: each scalar component $s_F^{(i)}$ is replaced by its ball of radius $r$, as defined in Algorithm 43 for scalar series. The result is a vector-valued series whose components are enclosed in validated balls of radius $r$, as it can be seen in Algorithm 56.

**Algorithm 56** Ball around a vector-valued series $\mathbf{s_F}$

```
1: function BALL(s_F, r)
2:     R ← zero series in s_F.space
3:     for i = 1, ..., s_F.space.d do
4:         R[i] ← BALL(s_F^(i), r)                      ▷ Algorithm 43
5:     end for
6:     return R
7: end function
```

#### 7.5.2.3 Antiderivatives of a vector series space

For $\mathbf{s_F} = (s_F^{(1)}, \ldots, s_F^{(d)}) \in \mathcal{F}_\rho^d$, the $n$-th antiderivative is obtained by applying the scalar procedure from Section 7.4.4 to each component independently:

$$\text{Antiderivative}^n(\mathbf{s_F}) \;=\; \big(\text{Antiderivative}^n(s_F^{(1)}), \ldots, \text{Antiderivative}^n(s_F^{(d)})\big).$$

**Implementation.** At the implementation level, each component $s_F^{(i)}$ is integrated using the scalar routines (Algorithms 46 and 48), so that truncation flags, parity changes, and rigorous error bounds are preserved exactly as in the scalar case. The results are then reassembled into a vector series of the same dimension $d$, consistent with Definition 7.10, as it is summarised in Algorithm 57.

## 7.6 Operators

In the framework of computer-assisted proofs for functional equations, linear and nonlinear operators play a central role. From a theoretical point of view, two classes of operators are of particular

**Algorithm 57** $n$-th antiderivative of a vector-valued series $\mathbf{s_F}$

**function** ANTIDERIVATIVE($\mathbf{s_F}, n$)
  $R \leftarrow$ zero series in $\mathbf{s_F}.space$
  **for** $i = 1, \ldots, \mathbf{s_F}.space.d$ **do**
    $R[i] \leftarrow$ ANTIDERIVATIVE($s_F^{(i)}, n$) ▷ Algorithms 47 and 48
  **end for**
  **return** $R$
**end function**

importance:

- **Compact operators.** These map bounded sets into relatively compact ones. In practice, compactness often arises from smoothing effects (e.g. integral operators, Fourier multipliers with decaying symbols). Compact operators are crucial in computer-assisted proofs because they allow one to apply fixed-point theorems (Schauder, Leray–Schauder) and to control spectral properties.

- **Non-compact operators.** These may not reduce dimensionality but can still be bounded in the Banach space of interest. Examples include differential operators and multiplication by unbounded functions. In rigorous numerics, one typically separates the operator into a dominant finite-dimensional part (handled exactly on a truncated basis) and a tail estimate (controlled analytically).

In both cases, the key idea is to work with an operator as an abstract mapping on a series space, and to quantify its action on the basis elements of the space. This allows one to compute operator norms, approximate spectra, and verify the conditions required by fixed-point theorems.

### 7.6.1 Implementation

The implementation mirrors this theoretical structure. We introduce an abstract type `AbstractOperator{S}` parameterised by a series space $S$, and a concrete wrapper type `Operator{S,F}` that couples a space with an underlying function $f$ acting on series. The operator can then be applied directly as a callable object, and its norm is computed by evaluating its action on the canonical basis (Algorithms 29 and 32).

**Algorithm 58** Operator structure

**Type:** `Operator{S,F}` with fields:

- `space :: Space` (series space on which the operator acts),
- `f :: F` (function implementing the operator).

**function** OPERATOR($space, f$)
  **return** new operator wrapping $f$ with domain $space$
**end function**
**function** APPLY($O, u$)
  **return** $O.f(u)$
**end function**

The operator norm is obtained by iterating over the (normalised, infinite) basis of the space, applying the operator to each basis element, and returning the maximum of the resulting norms. This is summarised in Algorithm 59.

**Algorithm 59** Operator norm

```
function NORM(O)
    B ← BASIS(O.space, normalized=true, infinite=true)        ▷ Algorithm 29
    n ← 0
    for each basis element b ∈ B do                           ▷ Algorithms 32 and 53
        R ← O(b)
        n ← max(n, ‖R‖)                                       ▷ Algorithms 28 and 55
    end for
    return n
end function
```

#### 7.6.1.1 Compact operators

Compact operators arise naturally in computer-assisted proofs: they map bounded sets to relatively compact ones, which in practice often means that they "suppress" high-frequency modes. Examples include integral operators, Fourier multipliers with rapidly decaying symbols, and projections onto finite-dimensional subspaces. From a rigorous numerics perspective, compact operators are useful because they can be represented (up to controllable tails) by finite matrices, making it possible to compute their norms, spectra, and inverses explicitly.

**Implementation.** In code, compact operators are realised by restricting their action to a finite-dimensional subspace of the underlying series space. This is encoded in the type `CompactOperator{S,MT}` (see Algorithm 60), which stores:

- the ambient space $S$,
- a finite matrix $M$ describing the operator restricted to a chosen subspace of coefficients,
- the subspace dimension.

Two constructors are provided:

- direct construction from a matrix $M$, when the finite representation is known,
- automatic construction from a function $f$, which applies $f$ to each basis element and fills the columns of $M$ accordingly.

This design allows seamless use of linear algebra operations (addition, subtraction, scaling, inversion) by overloading the corresponding methods.
The action of a compact operator on a series is reduced to multiplying the stored matrix $M$ by the coefficient subvector corresponding to the chosen subspace. Because only this finite block of coefficients is involved, both storage and computational cost are significantly reduced. The operator norm is then computed by scanning over a normalised basis of the subspace (Algorithm 61).

*Remark.* Because compact operators are represented by finite matrices, they can be combined and manipulated symbolically using matrix arithmetic. For example, addition, scalar multiplication, and inversion are delegated to the underlying matrix $M$, while preserving the link with the ambient series space.

## 7.7 A fixed point method

We already encountered the fixed-point philosophy in Section 7.4.3.3, where the validation of the DFFT approximation relied on constructing the Newton-like operator

$$G(x) = x - A\mathcal{N}(x),$$

**Algorithm 60** Compact operator structure and construction

**Type:** `CompactOperator{S,MT}`
Fields:

- `space :: Space` (ambient series space)
- `M :: AbstractMatrix` (matrix representation on truncated basis)
- `subspace_dim :: Int` (dimension of the subspace)

**function** CompactOperator($space, M, subspace\_dim$)
  **return** new compact operator wrapping $M$ on $space$
**end function**
**function** CompactOperator($space, f, subspace\_dim$)
  $Mdim \leftarrow$ n_coeffs($space, subspace\_dim$) ▷ Algorithm 20
  initialise $M$ as zero matrix of size $Mdim \times Mdim$
  $B \leftarrow$ Basis($space$, `infinite=false`, `subspace_dim=subspace_dim`) ▷ Alg. from 29 to 31
  $sbs\_ind \leftarrow$ get_subspace_indices($space, subspace\_dim$) ▷ Algorithm 22
  **for** each basis vector $b$ with index $k = 1, \ldots, Mdim$ **do** ▷ Algorithms 32 and 53
    compute $M[:, k] \leftarrow f(b).C[sbs\_ind]$
  **end for**
  **return** `CompactOperator(space,M,subspace_dim)`
**end function**

**Algorithm 61** Action and norm of a compact operator

**function** mul!($A, F, R$)
  $C \leftarrow$ get_subspace_indices($A.space, A.subspace\_dim$) ▷ Algorithm 22
  $R.C[C] \leftarrow A.M * F.C[C]$
**end function**
**function** Norm($A$)
  $B \leftarrow$ Basis($A.space$, `true`, `A.subspace_dim`) ▷ Algorithms from 29 to 31
  $R \leftarrow$ zero series in $A.space$
  $n \leftarrow 0$
  **for** each basis vector $b \in B$ **do** ▷ Algorithms 32 and 53
    mul!($A, b, R$)
    $n \leftarrow \max(n, \|R\|)$ ▷ Algorithms 28 and 55
  **end for**
  **return** $n$
**end function**

and proving that it is contractive on a neighbourhood of the numerical approximation. The essential idea was to reduce the nonlinear problem $\mathcal{N}(S) = 0$ (where $S$ denotes the exact solution sought) to the verification of explicit defect and Lipschitz bounds, so that Banach's Theorem 7.9 could be applied. In this Section we return to the same principle, but with a *different operator construction*, adapted to general nonlinear fixed-point equations of the form

$$F(u) = u.$$

The unifying theme is the use of a Newton-like modification that transforms the original problem into a contraction mapping (see Definition 7.11). While in the DFFT setting this was achieved through the operator $G(x) = x - A\mathcal{N}(x)$, here we adopt a different construction tailored to general fixed-point problems. The guiding principle, however, remains the same: computer-assisted proofs often reduce a nonlinear equation to the search for a fixed point of a suitably chosen operator. That is, one seeks $u \in \mathcal{F}_\rho^d$ (cf. Definition 7.10) such that $F(u) = u$. Once such an operator $F : \mathcal{F}_\rho^d \to \mathcal{F}_\rho^d$

has been identified, Banach's fixed point Theorem (7.12) provides sufficient conditions to guarantee both the existence and the uniqueness of the solution within a neighbourhood of an approximate numerical guess.

The first step of our strategy is to compute a high-accuracy numerical approximation $u_0$ of the desired solution by means of a suitable numerical algorithm. Since the operator $F$ is typically nonlinear and not directly contractive, we introduce a correction mechanism inspired by Newton's method. To this end, we define a linear approximation of the inverse Jacobian at $u_0$ and construct a modified operator that is closer to being contractive. More precisely, we choose a "finite" approximation $M$ for the map

$$\mathrm{Id} + [DF(u_0) - \mathrm{Id}]^{-1},$$

and then define

$$C(u) = F(u) - M\,[F(u) - u].$$

This map $C$ may be regarded as a *Newton-like operator* for $F$: formally, if $M$ were exactly equal to $[DF(u_0)]^{-1}$, then $C$ would reduce to the Newton iteration associated with $F$.

**Contraction principle.** The goal is to prove that $C$ is a *contraction mapping*. Recall the definition:

**Definition 7.11** (Contraction mapping)**.** Let $(\mathcal{X}, \|\cdot\|)$ be a Banach space. A mapping $C : B \subset \mathcal{X} \to \mathcal{X}$ is called a *contraction* on the closed ball $B$ if there exists a constant $0 < k < 1$ such that

$$\|C(u) - C(v)\| \le k\,\|u - v\|, \quad \text{for all } u, v \in B.$$

The Banach contraction mapping theorem then asserts that every contraction mapping on a complete metric space admits a unique fixed point in $B$, and moreover, that iterates of $C$ converge exponentially fast to that fixed point.
In our setting, the contraction property is established by verifying two types of estimates:
1. *Closeness of the approximation:* we require that $C$ maps the approximate solution $u_0$ close to itself, i.e.

$$\|C(u_0) - u_0\| < \varepsilon,$$

for some small $\varepsilon > 0$.
2. *Boundedness of the derivative:* we need to show that the Fréchet derivative of $C$ is uniformly bounded on a closed ball $B$ of radius $r$ centered at $u_0$, namely

$$\|DC(u)\| < k, \quad \text{for all } u \in B,$$

with $k < 1$.
If both conditions are satisfied, together with the compatibility inequality

$$\varepsilon + kr < r,$$

then Banach's theorem applies and we can conclude the existence of a unique fixed point of $F$ in the ball $B$. This statement can be summarised in the following theorem:

**Theorem 7.12** (Banach fixed point method for CAPs on $\mathcal{F}^d_\rho$, [129])**.** *Let $u_0 \in \mathcal{F}^d_\rho$ be an approximate solution of the fixed point equation $F(u) = u$, where $F : \mathcal{F}^d_\rho \to \mathcal{F}^d_\rho$. Assume there exists a bounded linear operator $M : \mathcal{F}^d_\rho \to \mathcal{F}^d_\rho$ and constants $\varepsilon > 0$, $k \in (0, 1)$ such that:*

1. *$M - \mathrm{Id}$ has a bounded inverse on $\mathcal{F}^d_\rho$;*
2. *$\|C(u_0) - u_0\|_\rho < \varepsilon$;*

3. $\|DC(u)\|_{\mathcal{L}(\mathcal{F}_\rho^d)} < k$ *for all* $u \in B(u_0, r)$;

4. $\varepsilon + kr < r$,

*where* $C(u) := F(u) - M\,[F(u) - u]$ *and* $B(u_0, r) = \{u \in \mathcal{F}_\rho^d : \|u - u_0\|_\rho \le r\}$. *Then* $F$ *admits a unique fixed point* $u \in B(u_0, r)$.

*Remark.* This mirrors the DFFT validation in Section 7.4.3.3, where the Newton-like map $G(x) = x - A\mathcal{N}(x)$ (with $\mathcal{N}(S) = 0$) was made contractive via explicit defect and Lipschitz bounds. Here we instead use $C(u) = F(u) - M[F(u) - u]$ on $(\mathcal{F}_\rho^d, \|\cdot\|_\rho)$ to enforce contractivity for general fixed-point problems. Both approaches combine a numerical approximation with rigorous operator bounds in the same Banach framework.

This procedure forms the backbone of many computer-assisted proofs: one combines a high-precision numerical approximation $u_0$ with rigorous analytic estimates to certify the contractivity of the Newton-like operator $C$. The delicate part is the rigorous computation of the bounds $\varepsilon$ and $k$, which requires interval arithmetic, Fourier-based representations, and careful control of truncation errors.

### 7.7.1 Implemented code

The abstract framework developed in Section 7.4.3.3 and Theorem 7.12 is made concrete by the routine PROVE_CONTRACTION$(\cdot)$ (see Algorithm 63). This routine verifies the hypotheses of the fixed-point method on a finite, rigorously controlled truncation of the Banach space $(\mathcal{F}_\rho^d, \|\cdot\|_\rho)$. Given a problem specification $P$, which encodes the nonlinear map $F$, its Jacobian action $DF$, and the truncation level, together with an approximate solution $u_0$, it computes the defect $\varepsilon$, the Lipschitz constant $k$ of the Newton-like operator $C$, and checks the compatibility inequality $\varepsilon + kr < r$ on the ball $B(u_0, r)$ (see Section 7.5.2.2). In this way, the abstract contraction-mapping argument is turned into an explicit computer-assisted proof.
The structure $P$ mirrors the operator abstractions from Section 7.6. The field `space` carries the ambient series structure, while TRUNC_SPACE$(\cdot)$ fixes the finite-dimensional companion needed to realise compact operators (Algorithms 60-61). The entries `F` and `JF` provide the nonlinear map and its Jacobian action, respectively, which are used to assemble the Newton-like modifier $C(u) = F(u) - M[F(u) - u]$ (cf. Section 7.7).
The parameter `M_dim` specifies the truncation level at which $M$ is represented as a finite matrix,

$$M \;=\; \mathbb{P}_\ell\, \widehat{M}\, \mathbb{P}_\ell,$$

where $\mathbb{P}_\ell$ denotes the canonical projection in $\mathcal{F}_\rho^d$ (see Definition 7.4) onto Fourier polynomials of degree $k \le \ell$. On this truncated subspace we verify that $M - I$ is invertible; hence $C$ and $F$ share the same fixed point $u^*$:

$$C(u^*) = u^* \iff (I - M)\big(F(u^*) - u^*\big) = 0 \iff F(u^*) = u^*.$$

Thus the choice $M = \mathbb{P}_\ell \widehat{M} \mathbb{P}_\ell$ materialises the Newton-like correction directly on the truncated Fourier subspace, in perfect analogy with the finite-matrix representation of compact operators.
In practice, $\widehat{M}$ may be constructed from the matrix of $DF(u_0)$, for instance by taking a rational approximation of $(DF(u_0) - I)^{-1}$, or by using the default choice $\widehat{M} = I + (P\,DF(u_0)\,P - I)^{-1}$. Any such choice is valid provided $M - I$ is invertible and the associated bounds can be verified. Finally, the parameter `r` fixes the radius of the validation ball required by Theorem 7.12.
The construction is summarised in Algorithm 62.

**Lemma 7.13** (Preconditions for the contraction check)**.** *Assume that on the truncated subspace of size* `M_dim` *the compact-operator matrix* $DF(u_0) - I$ *is invertible (cf. Algorithm 60), and that the norm on* `trunc_space` *is consistent with the ambient norm on* $\mathcal{F}_\rho^d$. *Then the quantities* $\varepsilon$ *and* $k$

**Algorithm 62** Proof data structure on $\mathcal{F}_\rho^d$

**Type:** `Proof{S, ST, FT, DFT}`
Fields:

- `space :: Space` (ambient space $\mathcal{F}_\rho^d$)
- `trunc_space :: Space` (finite truncation $\mathbb{P}_\ell \mathcal{F}_\rho^d$ for certified numerics)
- `F :: Function` (nonlinear map $F : \mathcal{F}_\rho^d \to \mathcal{F}_\rho^d$)
- `DF :: Function` (Jacobian action $(u, h) \mapsto DF(u)[h]$ on $\mathcal{F}_\rho^d$)
- `M_dim :: Int` (dimension $\ell$ of truncated subspace; default 10)
- `R_an :: Float64` (analytic radius / tail parameter; default $1 + \frac{1}{8}$)
- `r :: Float64` (validation ball radius; default $10^{-9}$)

1: **function** PROOF($space, F, DF$)
2: **parameters:** `M_dim`, `R_an`, `r`
3: $trunc\_space \leftarrow$ TRUNCATE_SPACE($space$) ▷ construct finite companion $\mathbb{P}_\ell \mathcal{F}_\rho^d$
4: **return** `Proof`($space$, $trunc\_space$, $F$, $DF$, `M_dim`, `R_an`, `r`)
5: **end function**

*computed below are valid upper bounds for the defect and the Lipschitz constant of the Newton-like operator $C$ on $B(u_0, r)$, up to analytically controlled tails.*

With these preconditions in place, we can describe the routine that verifies the hypotheses of Theorem 7.12 on the truncated subspace. Algorithm 63 takes as input a problem specification $P$ and an approximate solution $u_0$, and returns validated bounds for the defect $\varepsilon$, the Lipschitz constant $k$, and the contraction inequality.

**Algorithm 63** Contraction proof routine `prove_contraction`

**Require:** $P =$ `Proof`$(space, trunc\_space, F, DF, M\_dim, R\_an, r)$, approximate solution $u_0$
**Ensure:** Defect $\varepsilon$, Lipschitz bound $k$, and success flag $\big(\varepsilon + k\,r < r\big)$
$F(u_0) \leftarrow P.F(u_0)$ ▷ evaluate the nonlinear map at $u_0$
$u \leftarrow$ BALL($u_0$, $P.r$) ▷ define the validation domain $B(u_0, P.r)$, Alg. 56
**Compact Jacobian at $u_0$**
$DF(u_0) \leftarrow$ COMPACTOPERATOR($u_0.space$, $h \mapsto P.DF(u_0, h)$, $P.M\_dim$) ▷ Alg. 60 and 61
**Newton-like modifier**
$M \leftarrow I + \big(DF(u_0) - I\big)^{-1}$ ▷ requires $(DF(u_0) - I)^{-1}$ on the truncated subspace
**Defect at $u_0$**
$C(u_0) \leftarrow F(u_0) - M\big(F(u_0) - u_0\big)$
$\varepsilon \leftarrow \|C(u_0) - u_0\|_\rho$ ▷ Algorithms 55
**Lipschitz bound for $C$ on $B(u_0, r)$**
$DC \leftarrow$ OPERATOR($u_0.space$, $h \mapsto P.DF(u, h) - M\big(P.DF(u, h) - h\big)$) ▷ Algorithm 58
$k \leftarrow$ NORM($DC$) ▷ Algorithm 59
**Contraction check**
`success` $\leftarrow \big(\varepsilon + k \cdot P.r < P.r\big)$
**return** ($\varepsilon$, $k$, `success`)

### 7.7.2 Test functions

To ensure correctness of the implementation, a dedicated test file is included with the package. It contains five test problems designed to validate each of the components introduced so far: the

construction of series spaces, the Newton-like modification, the contraction check, and the rigorous enclosure of solutions.
Each test follows the same structure:

1. a nonlinear map $F$ and its Jacobian $DF$ are defined on a suitable Fourier series space (odd, even, or vector-valued);
2. an approximate solution $u_0$ is computed either by a Runge-Kutta solver or by a Newton iteration on the truncated Fourier coefficients;
3. the problem specification $P$ is assembled (cf. Algorithm 62);
4. the routine PROVE_CONTRACTION$(P, u_0)$ (cf. Algorithm 63) is called to compute the defect $\varepsilon$ and Lipschitz constant $k$, and to check the inequality $\varepsilon + kr < r$ required by Theorem 7.12.

**Examples.** The following problems are included:

- *Scalar ODE.* A simple second-order equation $\ddot{x} = f(x)$ with $f(x) = x - x^3$, tested in an odd Fourier space.
- *Vector-valued ODE.* A two-dimensional system $\ddot{w} = f(w)$ with quartic potential $V(x, y) = x^4 + y^4 + xy$, solved in a vector Fourier space.
- *ODE with nonlinearity $\sqrt{x}$.* An even Fourier space test where $f(x) = -x + a/\sqrt{x}$, with $a > 0$ chosen to avoid singularities, illustrating the DFFT approach from Section 7.4.3.3.
- *ODE with nonlinearity $1/x$.* Another even Fourier space problem, this time with $f(x) = -x/a + 1/(ax)$, where $a > 0$ fixes the scaling. The test validates the rigorous handling of singular nonlinearities.
- *Two-body problem.* A planar gravitational system with two equal masses interacting through Newton's potential $U(x) = -1/|x|$, formulated as $\ddot{x} = -\nabla U$. The solution is represented in a mixed vector Fourier space (even/odd components).

*Remark.* These test problems are not meant to exhaust the range of possible applications, but they provide a minimal verification suite. Passing all five tests confirms that the code can (i) construct the appropriate Fourier series space, (ii) generate a high-accuracy numerical approximation, and (iii) certify existence and uniqueness of a true solution via the contraction-mapping argument.

*Remark* 7.14. In the even and complete Fourier settings, the constant mode requires special attention. Unlike in the odd case, where the antiderivative is uniquely determined, here the constant coefficient is not automatically fixed by integration. After applying the antiderivative operator, one must therefore add an explicit correction to the constant term, ensuring that the resulting series satisfies the appropriate average condition. Concretely, this means that the constant mode itself must be formulated as a *fixed-point equation*, to be solved together with the higher Fourier modes. For the simple test problems considered in this Section, this condition can be written explicitly and incorporated directly into the definitions of $F$ and $DF$. In more general situations, however, additional care is required: if the constant mode is not treated consistently with the analytic structure of the equation, the fixed-point operator may fail to map the Fourier space into itself, thereby invalidating the contraction argument.

# Chapter 8

# Results and conclusions

This Chapter reports the main numerical and rigorous outcomes of the Thesis. Building on the computer-assisted proof (CAPs) infrastructure developed in Chapter 7 and on the variational-optimisation search of Chapters 3-5, we turn high-accuracy numerical periodic orbits into mathematically certified objects. Concretely, for each candidate orbit we instantiate the Banach-space proof structure and apply Algorithm 63 to obtain explicit defect bounds $\varepsilon$, contraction estimates $k$, and validation radii $r$, thereby proving existence and local uniqueness of a true periodic solution within a certified neighbourhood of the numerical seed. All computations and validation codes are implemented in `Julia` [70] and released in the repositories [125,131], together with scripts and data enabling full reproducibility. Part of the validated families presented here, as well as additional equivariant periodic solutions, are discussed in forthcoming work [132].
The Chapter is organised as follows. Section 8.1 explains how the general CAPs routine is specialised to the gravitational $n$-body problem, including the definition of the nonlinear map and its derivative in Fourier coordinates. Section 8.2 summarises a representative set of validated periodic orbits, combining variational diagnostics (action, gradient norm, Morse indices) with validation output (defect, contraction bound, truncation size, and certified radius). Section 8.3 then analyses computational requirements through two benchmark suites designed to probe runtime scaling and validation success rates. Section 8.4 concludes the Chapter with a discussion focused on the computer-assisted proof framework, analysing the computational regime explored in this work and outlining directions for further extension. Section 8.5 finally concludes the Thesis, synthesising the end-to-end methodology developed in this work and discussing its implications for the certified study of periodic solutions.

## 8.1 Application to the $n$-body problem

To apply the validation routine of Algorithm 63 to the periodic solutions of the $n$-body problem, we must specify the elements of the proof structure

$$P = \textsc{Proof}(space,\ trunc\_space,\ F,\ DF,\ M\_dim,\ R\_an,\ r).$$

The functional spaces used in the proof are automatically determined from the symmetries of the approximate periodic solution. Each coordinate of every body is assigned to an appropriate Fourier subspace (odd, even, or complete) depending on the observed parity of the corresponding component of the numerical orbit. This classification ensures that each variable is represented in the smallest spectral space compatible with its symmetry, reducing the computational dimension while preserving the structure of the solution. The nonlinear map $F$ entering the proof structure

corresponds to the integrated form of Newton's equations (1.1):

$$F(x) = \begin{cases} \Pi_0 x + T^2\,\mathrm{Antiderivative}^2\big(f(x)\big), & \text{for the constant Fourier mode in even,} \\ & \text{or complete settings} \\ T^2\,\mathrm{Antiderivative}^2\big(f(x)\big), & \text{for all nonzero Fourier modes,} \end{cases}$$

where $T$ is the period, $\Pi_0$ denotes the projection onto the constant (zero) Fourier mode and

$$f_i(x) = -\sum_{j\neq i} m_j \frac{x_i - x_j}{\|x_i - x_j\|^3}.$$

This distinction at the constant mode $x = 0$ reflects the correction discussed in Remark 7.14: in the even or complete Fourier settings, the constant mode is not fixed by the antiderivative and must therefore be treated as a fixed-point variable, ensuring that $F$ maps the Fourier space into itself. Its Fréchet derivative acts as

$$DF(x)[v] = \begin{cases} \Pi_0 v + T^2\,\mathrm{Antiderivative}^2\big(DF(x)[v]\big), & \text{for the constant Fourier mode in even,} \\ & \text{or complete settings} \\ T^2\,\mathrm{Antiderivative}^2\big(DF(x)[v]\big), & \text{for all nonzero Fourier modes,} \end{cases}$$

where $\Pi_0$ denotes the projection onto the constant (zero) Fourier mode and

$$DF(x)[v]_i = -\sum_{j\neq i} m_j \left[ \frac{v_i - v_j}{\|x_i - x_j\|^3} - 3\,\frac{(x_i - x_j)\,\langle x_i - x_j,\, v_i - v_j\rangle}{\|x_i - x_j\|^5} \right].$$

As noted previously and in Remark 7.14, one needs to ensure proper treatment of the constant Fourier mode, which is not fixed by integration in the even or complete settings.
The operators `F` and `DF` are implemented according to these formulae, using memoised routines for the distance terms and spectral antiderivatives to guarantee efficiency and reproducibility. Once these components are defined, the remaining quantities $M\_dim$, $R\_an$, and the validation radius $r$ are set according to the chosen truncation order and analytic estimates, completing the data required by Algorithm 63.
In what follows, we present a short selection of periodic orbits obtained through the computer-assisted framework developed in this work. This sample is not exhaustive, but is intended to illustrate that the validation method proposed can successfully handle configurations with different masses, distinct group symmetries, and varying numbers of bodies. The first three examples, shown in Figures 8.1, 8.2, and 8.3, correspond to well-known classical orbits, while the last three, depicted in Figures 8.4, 8.5, and 8.6, represent new configurations that have been rigorously *validated* by means of our computer-assisted method.

## 8.2 Overview of the validated results

We now comment on the six periodic solutions reported in Figures 8.1-8.6. Each figure collects two layers of information: the *variational/optimisation output* (action value, gradient norm, Morse indices) and the *computer-assisted validation output* (the defect $\epsilon$, the contraction bound $k$, the truncation size $M_{\mathrm{dim}}$ and the validation radius $r$ for Algorithm 63). The purpose of this selection is twofold: first, to provide a representative sample of the orbit geometries produced by the symmetry-constrained variational search; second, to illustrate that the Banach-space validation pipeline can be applied systematically across different symmetry types, different Fourier parity layouts (even/odd/complete), and different problem sizes.
More precisely, the first three examples correspond to classical configurations that are widely used as benchmarks in the periodic-orbit literature. They serve here as sanity checks for the end-to-end pipeline: starting from a numerical approximation produced by the variational framework of

| Orbit data | | |
|---|---|---|
| **Group generator(s)** | $\ker\tau = \langle \mathrm{Id} \rangle, \quad \bar{G} = \langle r \rangle$<br>$\rho(r) = -\mathrm{Id}, \quad \sigma(r) = ()$ | |
| **Action value** | | 9.8022 |
| **Gradient norm** | | $7.72 \times 10^{-11}$ |
| **Morse on $\mathbb{I}$** | Index | 0 |
| | Min negative eigenvalue | — |
| **Morse on $T$** | Index | 0 |
| | Min negative eigenvalue | — |
| **Winding number** | | — |

| CAPs data | |
|---|---|
| **Result** | **True** |
| **k** | 0.00192522193236973 |
| $\epsilon$ | $4.882899389969273 \times 10^{-15}$ |
| $\mathbf{M}_{\mathrm{dim}}$ | 20 |
| **r** | $1.0 \times 10^{-7}$ |
| **Spaces** | [Even, Odd, Complete, Complete, Complete, Complete] |
| $\mathbf{R}_{\mathrm{an}}$ | $1 + 2^{-11}$ |
| **N.of coefficients** | 70 |

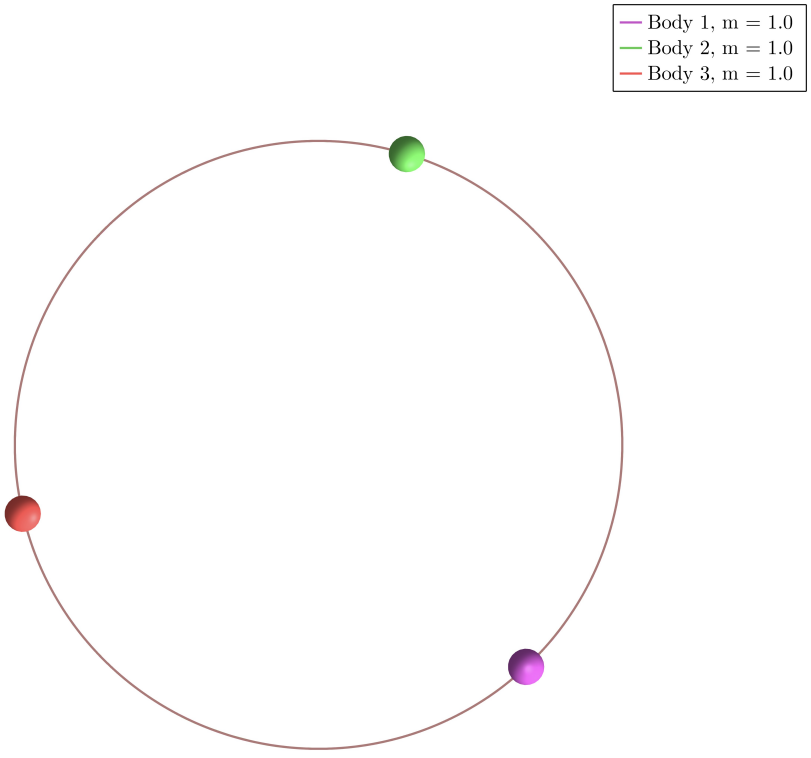



Figure 8.1: Circular periodic orbit, used as a baseline benchmark for the validation framework.

| Orbit data | | |
|---|---|---|
| **Group generator(s)** | $\ker\tau = \langle \mathrm{Id} \rangle, \quad \bar{G} = \langle r, s \rangle$<br>$\rho(r) = \mathrm{Id}, \quad \sigma(r) = (1,2,3)$<br>$\rho(s) = -\mathrm{Id}, \quad \sigma(s) = (2,3)$ | |
| **Action value** | | 5.8584 |
| **Gradient norm** | | $4.85 \times 10^{-12}$ |
| **Morse on $\mathbb{I}$** | Index | 0 |
| | Min negative eigenvalue | — |
| **Morse on $T$** | Index | 2 |
| | Min negative eigenvalue | -0.2257 |
| **Winding number** | | — |

| CAPs data | |
|---|---|
| **Result** | **True** |
| **k** | 0.29648219249810126 |
| $\epsilon$ | $2.6031971246524157 \times 10^{-8}$ |
| $\mathbf{M}_{\mathrm{dim}}$ | 20 |
| **r** | $1.0 \times 10^{-7}$ |
| **Spaces** | [Even, Odd, Complete, Complete, Complete, Complete] |
| $\mathbf{R}_{\mathrm{an}}$ | $1 + 2^{-11}$ |
| **N. of coefficients** | 70 |

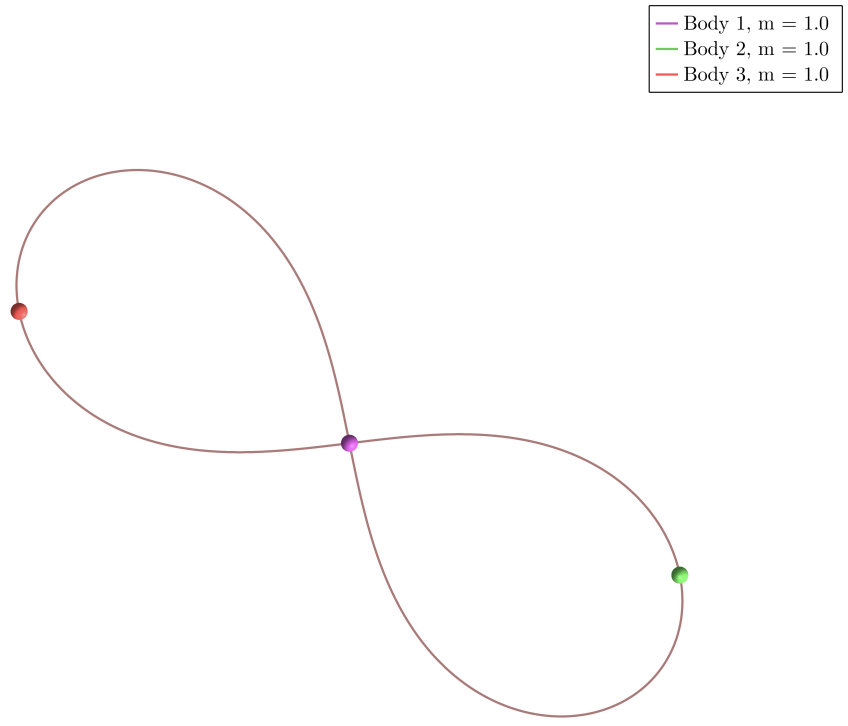



Figure 8.2: Figure-eight choreography, illustrating a classical nontrivial benchmark solution.

| Orbit data | | |
|---|---|---|
| **Group generator(s)** | $\ker \tau = \langle \mathrm{Id} \rangle, \quad \bar{G} = \langle r \rangle$<br>$\rho(r) = -\mathrm{Id}, \quad \sigma(r) = ()$ | |
| **Action value** | | 10.4421 |
| **Gradient norm** | | $3.22 \times 10^{-8}$ |
| **Morse on $\mathbb{I}$** | Index | 0 |
| | Min negative eigenvalue | — |
| **Morse on $T$** | Index | 2 |
| | Min negative eigenvalue | -0.0617 |
| **Winding number** | | — |

| CAPs data | |
|---|---|
| **Result** | **True** |
| **k** | 0.04181056784688636 |
| $\epsilon$ | $4.396679534045844 \times 10^{-10}$ |
| $\mathbf{M}_{\mathrm{dim}}$ | 20 |
| **r** | $1.0 \times 10^{-7}$ |
| **Spaces** | [Complete, Complete, Complete, Complete, Odd, Even] |
| $\mathbf{R}_{\mathrm{an}}$ | $1 + 2^{-11}$ |
| **N. of coefficients** | 70 |

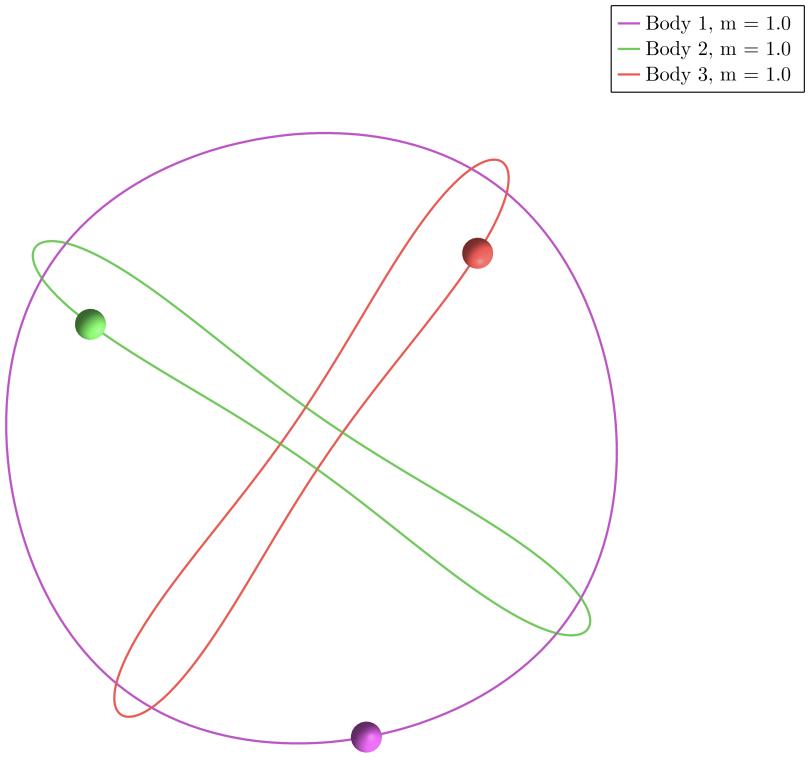



Figure 8.3: Ducati periodic orbit, highlighting a different symmetry and parity structure.

| Orbit data | | |
|---|---|---|
| **Group generator(s)** | $\ker\tau = \langle \mathrm{Id} \rangle, \quad \bar{G} = \langle r, s \rangle$<br>$\rho(r) = -\mathrm{Id}, \quad \sigma(r) = ()$<br>$\rho(s) = \begin{pmatrix} -\mathrm{Id}_2 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma(s) = ()$ | |
| **Action value** | | 11.8999 |
| **Gradient norm** | | $2.52 \times 10^{-8}$ |
| **Morse on $\mathbb{I}$** | Index | 0 |
| | Min negative eigenvalue | — |
| **Morse on $T$** | Index | 9 |
| | Min negative eigenvalue | -11.6320 |
| **Winding number** | | — |

| CAPs data | |
|---|---|
| **Result** | **True** |
| **k** | 0.023415210888609643 |
| $\epsilon$ | $1.38617673728876 \times 10^{-10}$ |
| $\mathbf{M}_{\mathrm{dim}}$ | 30 |
| **r** | $1.0 \times 10^{-7}$ |
| **Spaces** | [Odd, Odd, Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd, Even] |
| $\mathbf{R}_{\mathrm{an}}$ | $1 + 2^{-11}$ |
| **N. of coefficients** | 100 |

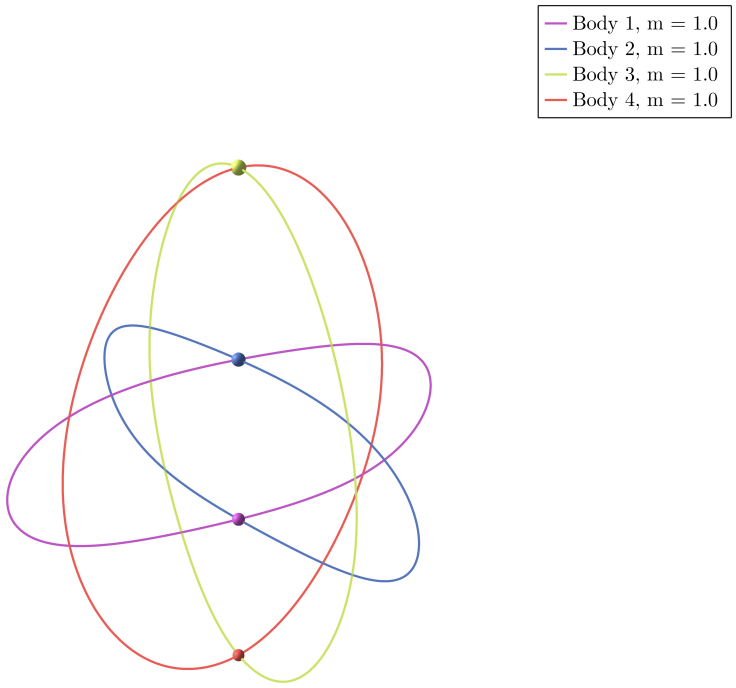



Figure 8.4: Planetary-type periodic orbit with four equal-mass bodies arranged in a symmetric configuration.

| Orbit data | | |
|---|---|---|
| **Group generator(s)** | $\ker\tau = \langle \mathrm{Id} \rangle, \quad \bar{G} = \langle r, s \rangle$<br>$\rho(r) = -\mathrm{Id}, \quad \sigma(r) = ()$<br>$\rho(s) = \begin{pmatrix} -\mathrm{Id}_2 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma(s) = ()$ | |
| **Action value** | | 10.9984 |
| **Gradient norm** | | $1.5 \times 10^{-9}$ |
| **Morse on $\mathbb{I}$** | Index | 4 |
| | Min negative eigenvalue | -0.3558 |
| **Morse on $T$** | Index | 19 |
| | Min negative eigenvalue | -0.4950 |
| **Winding number** | | — |

| CAPs data | |
|---|---|
| **Result** | **True** |
| **k** | 0.16018282271473855 |
| $\epsilon$ | $2.168712672834166 \times 10^{-11}$ |
| $\mathbf{M}_{\mathrm{dim}}$ | 40 |
| **r** | $1.0 \times 10^{-8}$ |
| **Spaces** | [Odd, Odd, Even, Odd, Odd, Even, Odd, Odd, Even] |
| $\mathbf{R}_{\mathrm{an}}$ | $1 + 2^{-11}$ |
| **N. of coefficients** | 100 |

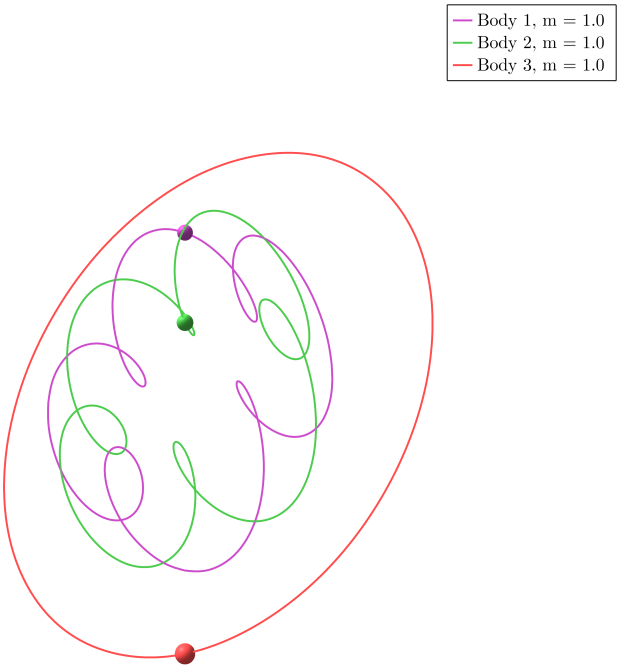



Figure 8.5: A non-classical periodic orbit with three equal-mass bodies, exhibiting a markedly different geometry from the standard choreographies.

| Orbit data | | |
|---|---|---|
| **Group generator(s)** | $\ker\tau = \langle \mathrm{Id} \rangle, \quad \bar{G} = \langle r, s \rangle$<br>$\rho(r) = -\mathrm{Id}, \quad \sigma(r) = ()$<br>$\rho(s) = \begin{pmatrix} -\mathrm{Id}_2 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma(s) = ()$ | |
| **Action value** | | 8.5415 |
| **Gradient norm** | | $1.3447 \times 10^{-10}$ |
| **Morse on $\mathbb{I}$** | Index | 2 |
| | Min negative eigenvalue | -0.1393 |
| **Morse on $T$** | Index | 12 |
| | Min negative eigenvalue | -0.1117 |
| **Winding number** | | — |

| CAPs data | |
|---|---|
| **Result** | **True** |
| **k** | 0.010699445927175977 |
| $\epsilon$ | $6.730972405928748 \times 10^{-12}$ |
| $\mathbf{M}_{\mathrm{dim}}$ | 40 |
| **r** | $1.0 \times 10^{-10}$ |
| **Spaces** | [Odd, Odd, Even, Odd, Odd, Even, Odd, Odd, Even] |
| $\mathbf{R}_{\mathrm{an}}$ | $1 + 2^{-11}$ |
| **N. of coefficients** | 150 |

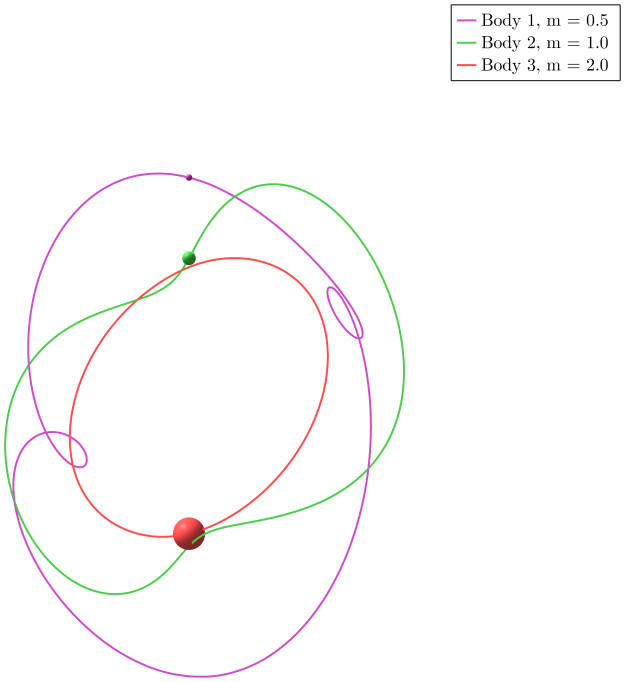



Figure 8.6: A periodic orbit with three bodies of unequal masses, illustrating the flexibility of the validation framework beyond equal-mass configurations.

Chapter 3, the present validation step confirms that the orbit is the shadow of a true solution of Newton's equations within explicit error bounds. The last three examples illustrate the same pipeline on less standard configurations, including higher Morse index solutions and configurations with different masses; these highlight that the method is not restricted to a single symmetry group or to minimising solutions.
Across all examples, the reported list of Fourier spaces (even/odd/complete blocks) is determined automatically from the observed parity of each coordinate of the numerical orbit, and the resulting reduced representation is used consistently in the evaluation of $F$, $DF$, antiderivatives, and operator bounds. This is precisely the mechanism that makes the validation routine scalable: the proof is carried out in the smallest Fourier space compatible with the orbit symmetries, while still producing a certified enclosure in the full phase space.

**Comparison with known Morse index results.** In addition to certifying the existence of the periodic solutions reported in Figures 8.1-8.6, the present validation framework provides discrete Morse index (see Section 4.1) information that can be directly confronted with known analytical and semi-analytical results for classical three-body orbits. To the best of our knowledge, explicit Morse index computations for periodic solutions of the three-body problem are available in the literature only for a limited number of highly symmetric configurations, most notably the Lagrangian circular orbit shown in Figure 8.1 and the figure-eight choreography shown in Figure 8.2, where the second variation of the action has been analysed by analytical or semi-analytical methods. In particular, for the Newtonian figure-eight choreography shown in Figure 8.2, Fukuda, Fujiwara, and Ozaki [102, 133] analysed the second variation of the action through the spectral properties of the continuous Hessian operator and introduced several Morse indices depending on the admissible class of variations. They showed that the figure-eight orbit has a positive Morse index ($N = 2$) when considered in the full space of periodic loops, while the choreographic and figure-eight constrained Morse indices vanish ($N_c = N_e = 0$), indicating that the orbit is a saddle point of the unrestricted action functional but a local minimiser within the corresponding symmetry-invariant subspaces. Our numerical results are in full agreement with this structure. When the discrete Hessian of the discretised action is evaluated over the full period, we obtain a Morse index equal to two, corresponding to unstable directions that break the choreographic symmetry, whereas the index vanishes when the computation is restricted to the fundamental domain associated with the figure-eight symmetry, as illustrated in Figure 8.2. Similar symmetry-dependent behaviour is observed for other highly symmetric periodic solutions among Figures 8.3 and 8.4, where nonzero Morse indices on the full loop space coexist with reduced or vanishing indices in symmetry-restricted settings.
This phenomenon is consistent with the general variational picture established, for instance, for the Lagrangian circular orbit shown in Figure 8.1 by Barutello, Jadanza, and Portaluri [99], who analysed the Morse index of such relative equilibria and highlighted the role played by symmetry constraints and invariant subspaces in determining their variational character. For further discussion of symmetry-restricted Morse index computations in this context, see [105].

## 8.3 Computational requirements and performance

Two complementary benchmark tests were performed to assess the practical performance of the computer-assisted proofs (CAPs) framework. The first test focuses primarily on *computational time* and is based on highly symmetric two-dimensional circular periodic orbits Section 8.3.1. The second test, discussed later in Section Section 8.3.2, considers three-dimensional "planetarium"-type configurations (see Figures 8.4 and 8.9) and is designed to analyse the *success rate of the validation procedure* as a function of the number of retained Fourier coefficients, the problem dimension, and the parity structure of the solution. All computer-assisted validation results reported in this Chapter were obtained using `Julia` [70] implementations provided in [125,131]. The periodic orbits themselves were *not* generated within the CAPs framework. Instead, high-accuracy numerical

approximations were first computed using the `Symorb` variational-optimisation system [43], together with the numerical procedures described in Chapters 3-5. These numerical solutions serve as input for the CAPs pipeline, whose sole purpose is the rigorous *validation* of their existence. For a fixed periodic orbit, the CAPs procedure consists of a single computational phase: the execution of Algorithm 63 in the setting described in Section 8.1. This algorithm verifies the existence and uniqueness of a true solution in a neighbourhood of a given numerical approximation by computing:

- the defect $\varepsilon$,
- a contraction bound $k$,
- and a validation radius $r$ such that the inequality

$$\varepsilon + kr < r$$

holds in the appropriate Fourier series Banach space.

**Hardware and software environment.** All benchmark tests were carried out on the virtual machine *MathHPC*, provided by the Data Center of the Department of Computer Science of the University of Turin within the HPC4AI infrastructure. The virtual machine is hosted on heterogeneous servers managed via OpenStack and runs on two Intel(R) Xeon(R) Gold 6230 CPUs at 2.10 GHz. Up to 30 CPU cores and 300 GB of RAM were available; however, each proof execution used only a single core because the code is not currently parallelisable. The operating system is Ubuntu 22.04.5 LTS, and all computations were performed using `Julia` v1.12.3 (15 December 2025, [70]). All results are fully reproducible using the provided repositories [125, 131]. Exact package versions, dependencies, and compiler settings are recorded in the corresponding project manifests.

**Empirical performance metrics.** For a fixed periodic orbit, the computational cost of the CAPs validation step is primarily governed by the following parameters:

1. the Fourier truncation level $N$, i.e. the number of retained Fourier coefficients per coordinate;
2. the parity structure of each coordinate (odd, even, or complete), which determines the effective number of stored coefficients and tail variables;
3. the dimension of the problem, determined by both the ambient spatial dimension and the number of bodies;
4. the compact-operator dimension $M_{\mathrm{dim}}$, which fixes the size of the finite Jacobian block used in the Newton-like correction and dominates the cost of matrix inversion and operator norm estimates.

### 8.3.1 Circular benchmark solutions (two-dimensional test)

To investigate the scaling behaviour of the CAPs framework with respect to these parameters, we consider a systematic benchmark based on circular periodic solutions. Among all periodic orbits studied in this Thesis, the circle is geometrically the simplest configuration and exhibits maximal symmetry. This makes it an ideal test case for isolating the computational impact of the Fourier truncation level, parity structure, and problem dimension from geometric complexity. The benchmark includes six distinct circular periodic orbits, obtained by increasing the number of bodies from three to five. For each number of bodies, two configurations are considered: one with equal masses and one with non-equal masses, where only a single mass differs from the others. This choice allows us to disentangle the effect of the number of bodies from that of symmetry breaking in the mass distribution. Figure 8.7 shows the six periodic orbits considered in this benchmark,

together with their symmetry-induced characteristics, namely whether each coordinate admits an odd, even, or complete Fourier expansion, as reported in the corresponding captions.
The validation results for these circular orbits are reported in two tables: Table 8.1 corresponding to low Fourier truncation levels , and Table 8.2 to moderate truncation levels.

Table 8.1: CAPs validation metrics for circular benchmark solutions with low Fourier truncation levels.

| **Case** | | **Coeff.** | **25** | | | **50** | | | **75** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bodies** | **Masses** | # | $\varepsilon$ | $k$ | time(s) | $\varepsilon$ | $k$ | time(s) | $\varepsilon$ | $k$ | time(s) |
| 3 | equal | $10N$+5 | 5.68E-16 | 1.92E-03 | 1.16E+03 | 6.64E-16 | 1.92E-03 | 6.15E+03 | 6.08E-16 | 1.92E-03 | 2.00E+04 |
| 3 | different | $10N$+5 | 6.11E-16 | 1.38E-03 | 1.16E+03 | 5.15E-16 | 1.38E-03 | 6.18E+03 | 8.41E-16 | 1.38E-03 | 2.00E+04 |
| 4 | equal | $8N$+4 | 4.23E-16 | 1.44E-03 | 9.83E+02 | 7.49E-16 | 1.44E-03 | 5.19E+03 | 7.10E-16 | 1.44E-03 | 1.69E+04 |
| 4 | different | $12N$+6 | 5.18E-16 | 1.46E-03 | 2.05E+03 | 1.19E-15 | 1.46E-03 | 1.13E+04 | 1.17E-15 | 1.46E-03 | 3.75E+04 |
| 5 | equal | $18N$+9 | 6.22E-16 | 2.07E-03 | 5.66E+03 | 1.08E-15 | 2.07E-03 | 3.49E+04 | 1.15E-15 | 2.07E-03 | 1.15E+05 |
| 5 | different | $18N$+9 | 9.52E-16 | 2.16E-03 | 5.71E+03 | 1.19E-15 | 2.16E-03 | 3.53E+04 | 1.08E-15 | 2.16E-03 | 1.14E+05 |

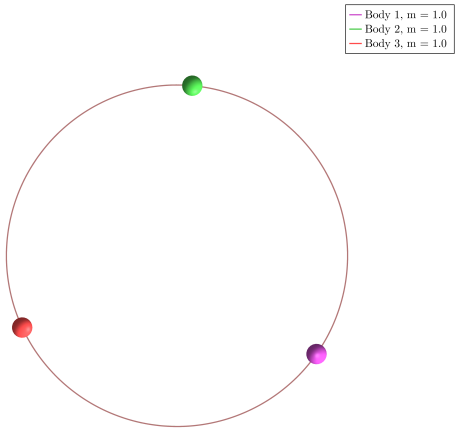
Table 8.2: CAPs validation metrics for circular benchmark solutions with moderate Fourier truncation levels.

| **Case** | | **Coeff.** | **100** | | | **125** | | | **150** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bodies** | **Masses** | # | $\varepsilon$ | $k$ | time(s) | $\varepsilon$ | $k$ | time(s) | $\varepsilon$ | $k$ | time(s) |
| 3 | equal | $10N$+5 | 9.28E-16 | 1.92E-03 | 4.64E+04 | 9.49E-16 | 1.92E-03 | 9.26E+04 | 9.81E-16 | 1.92E-03 | 1.66E+05 |
| 3 | different | $10N$+5 | 1.23E-15 | 1.38E-03 | 4.66E+04 | 1.26E-15 | 1.38E-03 | 9.40E+04 | 1.29E-15 | 1.38E-03 | 1.66E+05 |
| 4 | equal | $8N$+4 | 7.38E-16 | 1.44E-03 | 3.98E+04 | 7.72E-16 | 1.44E-03 | 7.56E+04 | 8.07E-16 | 1.44E-03 | 1.37E+05 |
| 4 | different | $12N$+6 | 1.12E-15 | 1.46E-03 | 9.10E+04 | 1.16E-15 | 1.46E-03 | 1.79E+05 | 1.20E-15 | 1.46E-03 | 3.18E+05 |
| 5 | equal | $18N$+9 | 1.06E-15 | 2.07E-03 | 2.74E+05 | 1.11E-15 | 2.07E-03 | 5.26E+05 | 1.15E-15 | 2.07E-03 | 9.95E+05 |
| 5 | different | $18N$+9 | 1.01E-15 | 2.16E-03 | 2.76E+05 | 1.03E-15 | 2.16E-03 | 5.31E+05 | 1.05E-15 | 2.16E-03 | 9.94E+05 |

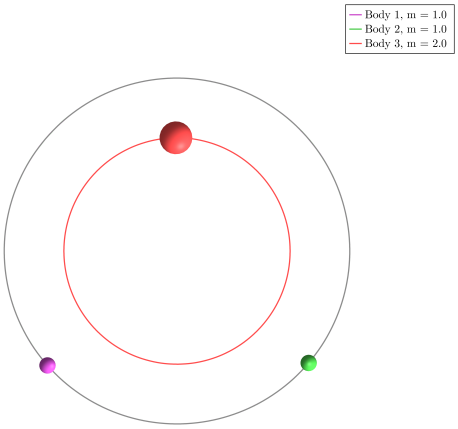
In all experiments, the compact-operator dimension was fixed to $M_{\text{dim}} = 20$, and the validation radius was chosen according to the rule $r = 8\,\varepsilon$. All bounds were computed using `BigFloat` arithmetic, and the analytic radius was set to $R_{\text{an}} = 1 + 2^{-11}$. These parameters were kept fixed throughout in order to ensure that the observed performance trends depend only on the structural properties of the Fourier spaces and the problem dimension. To complement the tabulated data, Figure 8.8 reports the wall-clock time required by the CAPs validation step as a function of the Fourier truncation level $N$, shown on a logarithmic scale.

**Observed scaling behaviour.** The results exhibit a clear and monotone increase of runtime as $N$ grows, across all tested configurations. This behaviour reflects the increasing cost of nonlinear operator evaluation, analytic tail bounds, and finite-dimensional Jacobian inversions as the Fourier representation is refined. Beyond this expected dependence on $N$, a consistent separation between configurations is observed. For every truncation level, the four-body equal-mass case is systematically the fastest, followed by the three-body configurations, while the five-body cases are the most expensive. This ordering is stable across all values of $N$. The explanation lies in the parity structure of the Fourier representation. In the four-body equal-mass configuration, all coordinates admit only odd or even Fourier expansions, and no complete series are required. As a result, the effective dimension of the underlying Banach space is significantly reduced. By contrast, configurations involving different masses or a larger number of bodies include complete Fourier components, which substantially increases the number of stored coefficients and the cost of operator bounds.

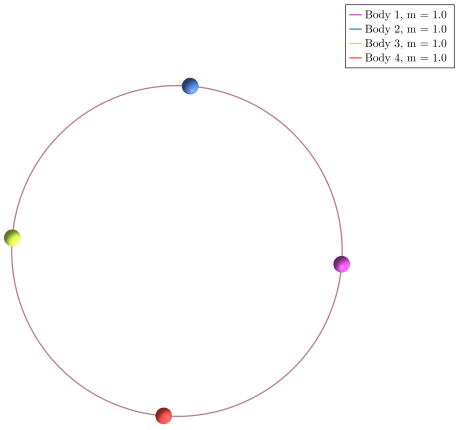
**Summary of the circle benchmark.** These benchmarks show that the computational cost of the CAPs validation increases monotonically with the Fourier truncation level $N$, reflecting the growing cost of operator evaluations and finite-dimensional Jacobian inversions. As discussed above, the dominant factors governing the runtime are the structure of the Fourier representation, most notably the number of retained modes, the parity-induced reduction of the underlying Banach space, and the overall problem size in terms of bodies and spatial dimensions, rather than the geometric complexity of the orbit itself. At the same time, it is important to observe that the



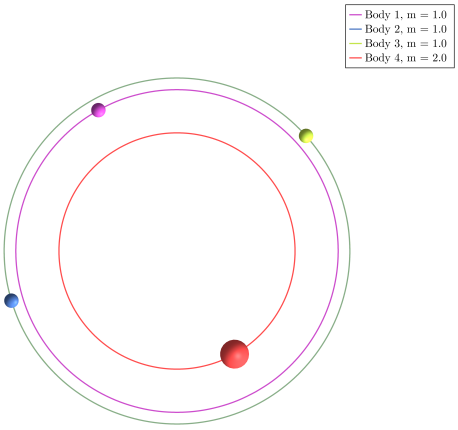

(a) 3B equal masses. Parity: [Odd, Even, Complete, Complete, Complete, Complete]



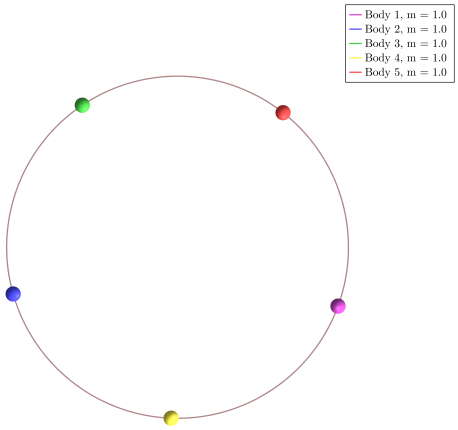

(b) 3B different masses. Parity: [Odd, Even, Complete, Complete, Complete, Complete]



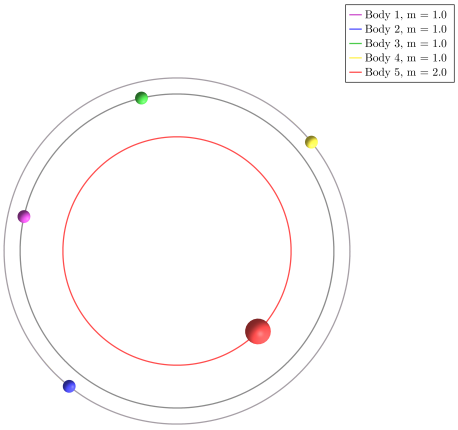

(c) 4B equal masses. Parity: [Odd, Even, Even, Odd, Odd, Even, Even, Odd]




(d) 4B different masses. Parity: [Odd, Even, Complete, Complete, Complete, Complete, Odd, Even]




(e) 5B equal masses. Parity: [Odd, Even, Complete, Complete, Complete, Complete, Complete, Complete, Complete, Complete]




(f) 5B different masses. Parity: [Odd, Even, Complete, Complete, Complete, Complete, Complete, Complete, Complete, Complete]

Figure 8.7: Circular periodic orbits used in the CAPs scaling benchmarks.

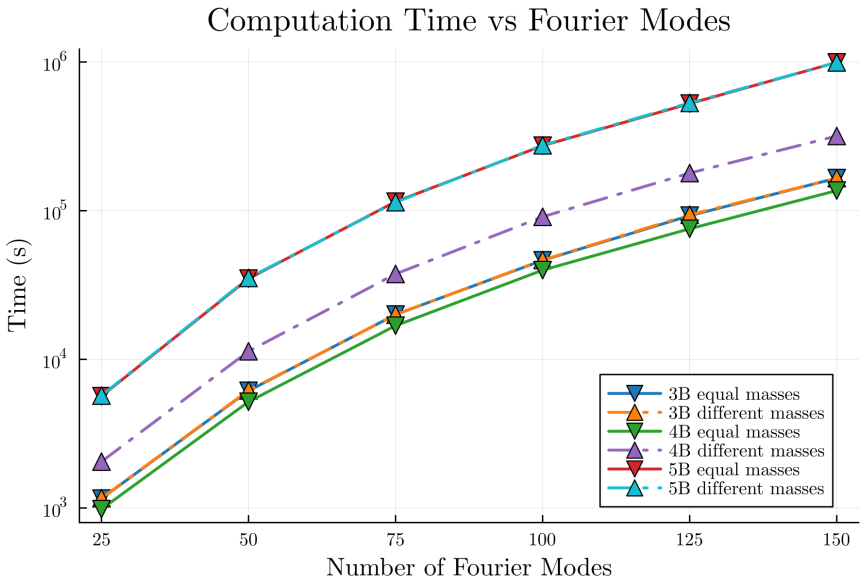



Figure 8.8: Runtime of the CAPs validation step versus the Fourier truncation level $N$ for circular benchmark solutions. Times (in seconds) are shown on a logarithmic scale. The consistent separation between curves reflects differences in parity structure and effective Banach-space dimension across the tested configurations.

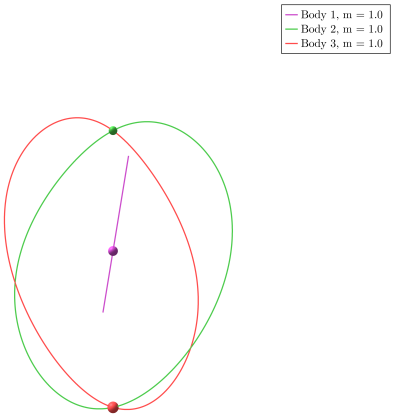
contraction constant $k$ remains essentially unchanged as $N$ increases for each fixed configuration. This behaviour is expected and desirable: $k$ is determined by the analytic properties of the problem and by the choice of the approximate inverse, and its stability across truncation levels indicates that the contraction mapping argument is not degraded by increasing spectral resolution. Consequently, once a sufficiently accurate numerical approximation is available, the CAPs validation step remains robust and predictable, with performance scaling consistently with the analytic and algebraic structure of the validated operators.

### 8.3.2 Planetarium benchmark solutions (three-dimensional test)

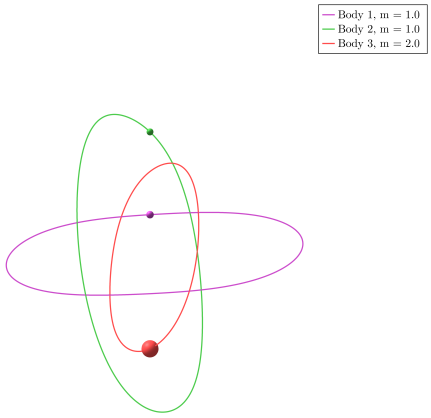
In order to complement the two-dimensional circular benchmarks, we performed a second set of experiments based on three-dimensional "planetarium"-type periodic solutions. Unlike the circular test, which was designed primarily to study runtime scaling, this benchmark focuses on the *success rate of the CAPs validation procedure* as a function of the number of retained Fourier coefficients, the number of bodies, and the symmetry properties of the orbit. As in the circular benchmark, we consider six distinct periodic orbits, obtained by increasing the number of bodies from three to five. For each number of bodies, two configurations are analysed: one with equal masses and one with different masses, where only a single mass differs from the others. All orbits are genuinely three-dimensional and therefore require a higher-dimensional Fourier representation than their planar counterparts. Figure 8.9 shows the six periodic orbits considered in this benchmark, together with their symmetry-induced characteristics, namely whether each coordinate admits an odd, even, or complete Fourier expansion, as reported in the corresponding captions.

The planetarium benchmark is conceptually analogous to the circular benchmark, in the sense that it is designed to test the CAPs framework on the geometrically simplest periodic orbits available in the three-dimensional setting. Just as circular motions play this role in two dimensions, planetarium-type solutions represent the most elementary and highly structured configurations among genuinely three-dimensional periodic orbits. This is the primary reason for their selection in the present benchmark.

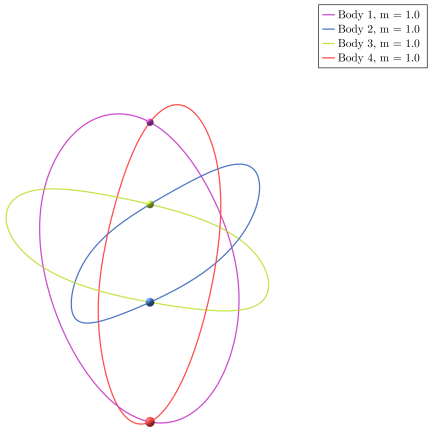
At the same time, the planetarium benchmark differs from the circular one in two important respects. First, the ambient spatial dimension is three, which substantially increases the dimension of the underlying vector-valued Fourier space. Second, all coordinates of the planetarium solutions



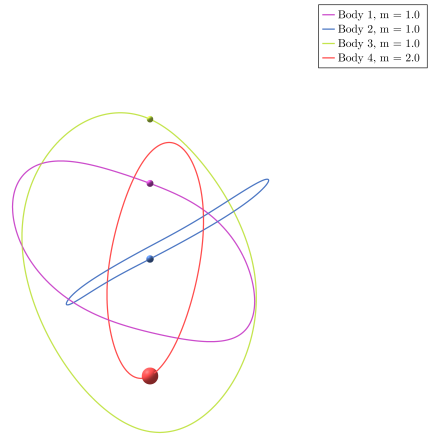

(a) 3B equal masses. Parity: [Odd, Odd, Odd, Odd, Odd, Even, Odd, Odd, Even]



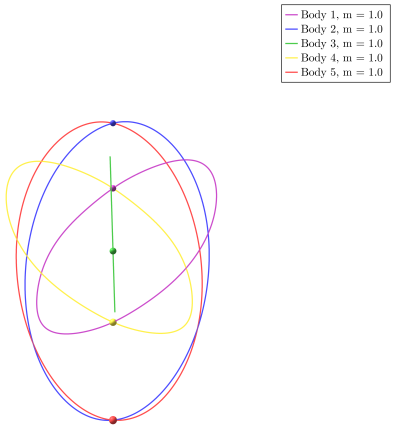

(b) 3B different masses. Parity: [Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd]



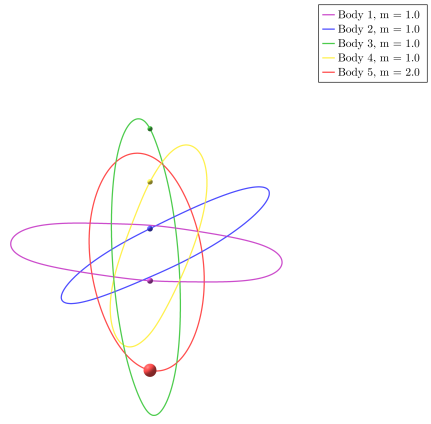

(c) 4B equal masses. Parity: [Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd]




(d) 4B different masses. Parity: [Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd]




(e) 5B equal masses. Parity: [Even, Odd, Odd, Even, Odd, Odd, Odd, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd]




(f) 5B different masses. Parity: [Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd, Even, Odd, Odd]

Figure 8.9: Planetarium periodic orbits used in the CAPs scaling benchmarks.

considered here admit purely odd or even Fourier expansions, and no complete Fourier series are required. This parity structure distinguishes the planetarium benchmark from several circular configurations and plays a crucial role in the computational efficiency, runtime scaling, and overall feasibility of the CAPs validation procedure.

Table 8.3: CAPs validation metrics for planetarium benchmark solutions with low Fourier truncation levels. Entries marked as `ERROR` correspond to cases in which the contraction condition required by the *DFFT a posteriori validation* (see Section 7.4.3.3) could not be verified. The four-body equal-mass configuration is split into Case (a) $M_{\text{dim}} = 20$ and Case (b) $M_{\text{dim}} = 30$.

| **Case** | | **Coeff.** | **25** | | | **50** | | | **75** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bodies** | **Masses** | # | $\varepsilon$ | $k$ | time(s) | $\varepsilon$ | $k$ | time(s) | $\varepsilon$ | $k$ | time(s) |
| 3 | equal | $9N+2$ | ——— ERROR ——— | | | 3.21E-9 | 4.81E-01 | 7.41E+03 | 1.80E-14 | 4.81E-01 | 2.42E+04 |
| 3 | different | $9N+3$ | ——— ERROR ——— | | | ——— ERROR ——— | | | 5.08E-10 | 1.24E-02 | 1.95E+04 |
| 4 | equal (a) | $12N+4$ | ——— ERROR ——— | | | ——— ERROR ——— | | | ——— ERROR ——— | | |
| 4 | equal (b) | $12N+4$ | ——— ERROR ——— | | | ——— ERROR ——— | | | ——— ERROR ——— | | |
| 4 | different | $12N+4$ | ——— ERROR ——— | | | ——— ERROR ——— | | | ——— ERROR ——— | | |
| 5 | equal | $15N+4$ | ——— ERROR ——— | | | ——— ERROR ——— | | | ——— ERROR ——— | | |
| 5 | different | $15N+5$ | ——— ERROR ——— | | | ——— ERROR ——— | | | ——— ERROR ——— | | |

Table 8.4: CAPs validation metrics for planetarium benchmark solutions with moderate Fourier truncation levels. Entries marked as `ERROR` correspond to cases in which the contraction condition required by the *DFFT a posteriori validation* (see Section 7.4.3.3) could not be verified. The four-body equal-mass configuration is split into Case (a) $M_{\text{dim}} = 20$ and Case (b) $M_{\text{dim}} = 30$.

| **Case** | | **Coeff.** | **100** | | | **125** | | | **150** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bodies** | **Masses** | # | $\varepsilon$ | $k$ | time(s) | $\varepsilon$ | $k$ | time(s) | $\varepsilon$ | $k$ | time(s) |
| 3 | equal | $9N+2$ | 1.15E-14 | 4.81E-01 | 5.09E+04 | 1.36E-14 | 4.84E-01 | 1.09E+05 | 1.70E-14 | 4.81E-01 | 2.03E+05 |
| 3 | different | $9N+3$ | 2.48E-14 | 1.24E-02 | 4.95E+04 | 4.29E-14 | 1.52E-02 | 8.25E+04 | 2.16E-14 | 1.24E-02 | 1.64E+05 |
| 4 | equal (a) | $12N+4$ | 1.39E-10 | 9.82E-01 | 1.21E+05 | 9.42E-14 | 9.82E-01 | 2.29E+05 | 4.23E-14 | 9.82E-01 | 4.10E+05 |
| 4 | equal (b) | $12N+4$ | 1.39E-10 | 3.79E-01 | 1.22E+05 | 9.42E-14 | 3.79E-01 | 2.34E+05 | 4.23E-14 | 3.79E-01 | 4.25E+05 |
| 4 | different | $12N+4$ | ——— ERROR ——— | | | 3.69E-09 | 1.09E-01 | 1.98E+05 | 3.31E-12 | 5.00E-02 | 3.91E+05 |
| 5 | equal | $15N+4$ | ——— ERROR ——— | | | ——— ERROR ——— | | | ——— ERROR ——— | | |
| 5 | different | $15N+5$ | ——— ERROR ——— | | | ——— ERROR ——— | | | ——— ERROR ——— | | |

In all experiments, the compact-operator dimension was fixed to $M_{\text{dim}} = 20$, and the validation radius was chosen according to the rule $r = 8\,\varepsilon$. All bounds were computed using `BigFloat` arithmetic, and the analytic radius was set to $R_{\text{an}} = 1 + 2^{-11}$. These parameters were kept fixed throughout in order to ensure that the observed performance trends depend only on the structural properties of the Fourier spaces and the problem dimension.

The only exception concerns the four-body equal-mass planetarium configuration, which is reported in Tables 8.3 and 8.4 as four-body equal-mass configuration Case (a) and Case (b). Case (a) corresponds to $M_{\text{dim}} = 20$, while in Case (b) the compact-operator dimension was increased to $M_{\text{dim}} = 30$. For $M_{\text{dim}} = 20$, validation repeatedly failed at low Fourier truncation levels, and no successful validation was obtained for $N = 25$, 50, or 75. At larger truncation levels, successful validations were eventually achieved, but with contraction bounds close to unity, namely $k \approx 0.92$. The corresponding defects decreased with increasing $N$, taking values $\varepsilon \approx 1.39 \times 10^{-10}$ for $N = 100$, $\varepsilon \approx 9.42 \times 10^{-14}$ for $N = 125$, and $\varepsilon \approx 4.23 \times 10^{-14}$ for $N = 150$.

In this regime, the observed contraction factor $k \approx 0.92$ implies that the condition

$$\varepsilon + kr < r$$

can only be satisfied if

$$r > \frac{\varepsilon}{1-k} \approx 12.5\,\varepsilon.$$

As a consequence, the standard choice $r = 8\,\varepsilon$ is insufficient to close the contraction argument. While this inequality suggests that a larger validation radius would be required, increasing the

radius does not provide a simple remedy: the entire a posteriori validation procedure would have to be repeated with the new value of $r$, and there is no guarantee that the contraction argument would then close.

As an alternative to increasing the validation radius, the dimension of the compact-operator approximation was increased to $M_{\mathrm{dim}} = 30$, yielding Case (b). This modification improves the quality of the approximate inverse primarily through a reduction of the contraction bound $k$ (while the defect $\varepsilon$ remains essentially unchanged at the relevant truncation levels). As a consequence, the contraction argument can be closed using the standard radius-selection rule $r = 8\,\varepsilon$. Successful validations are therefore obtained already at lower truncation levels, as reflected in the entries for Case (b) in Tables 8.3 and 8.4.

However, this general improvement is accompanied by a mildly increased computational cost. While the computation of the matrix $M$ becomes more expensive when larger finite blocks are used, the dominant part of the validation procedure is the estimation of the contraction bound $k$. Consequently, variations in $M_{\mathrm{dim}}$ have only a limited impact on the overall wall-clock time. This trade-off can be illustrated by the four-body equal-mass configuration at truncation levels $N = 100$, $N = 125$ and $N = 150$. For $N = 100$, increasing the compact-operator dimension from $M_{\mathrm{dim}} = 20$ (Case (a)) to $M_{\mathrm{dim}} = 30$ (Case (b)) leaves the defect essentially unchanged at $\varepsilon \approx 1.39 \times 10^{-10}$, while substantially reducing the contraction bound from $k \approx 0.982$ to $k \approx 0.379$. The corresponding wall-clock time increases slightly from $1.21 \times 10^5$ s to $1.22 \times 10^5$ s, corresponding to an overhead of approximately 0.8%. A similar behaviour is observed at $N = 125$. Both cases yield the same defect, $\varepsilon \approx 9.42 \times 10^{-14}$, while the contraction bound again drops from $k \approx 0.982$ to $k \approx 0.379$. The wall-clock time increases marginally, from $2.29 \times 10^5$ s to $2.34 \times 10^5$ s, corresponding to an overhead of approximately 2.2%. For $N = 150$, the defect remains unchanged at $\varepsilon \approx 4.23 \times 10^{-14}$, while increasing $M_{\mathrm{dim}}$ from 20 to 30 reduces the contraction bound from $k \approx 0.982$ to $k \approx 0.379$. The wall-clock time increases modestly from $4.10 \times 10^5$ s to $4.25 \times 10^5$ s, an overhead of about 3.7%. This behaviour is representative of the role played by the compact-operator dimension in the validation procedure. From a conceptual and algorithmic perspective, the compact-operator dimension $M_{\mathrm{dim}}$ is not intended as an accuracy parameter, but rather as the minimal finite-dimensional correction required to eliminate the non-contracting directions of the linearised operator. In the weighted Fourier spaces $\mathcal{F}^d_\rho$ considered here, the high-frequency modes are already strongly contractive due to analytic decay, and are therefore more efficiently controlled by abstract tail estimates than by explicit numerical inversion. For this reason, the most robust and computationally efficient choice of $M_{\mathrm{dim}}$ is the smallest value for which the contraction inequality can be rigorously verified. While increasing $M_{\mathrm{dim}}$ can be beneficial in situations where the contraction bound $k$ is the limiting factor, as illustrated by the benchmark tests above, such increases are used only to the extent needed to close the contraction argument. Enlarging the compact block beyond this point does not improve the defect $\varepsilon$ and may instead lead to unnecessary computational overhead and increased interval overestimation. Consequently, $M_{\mathrm{dim}}$ is increased only when required and never beyond the minimal level needed for a successful validation.

**Observed validation behaviour.** The results reveal a markedly different behaviour compared to the circular benchmark. For the three-body planetarium orbits, validation succeeds only beyond sufficiently large Fourier truncation levels. At low values of $N$, the DFFT-based a posteriori validation (see Section 7.4.3.3) fails because the contraction condition cannot be verified: the truncated Fourier representation does not yet provide sufficient spectral resolution for the defect $\varepsilon$ to fall below the threshold required by the Banach fixed-point argument. Once the Fourier representation is refined, the defect decreases and successful validations are obtained. This trend is observed for both equal-mass and different-mass configurations, although the latter typically requires a larger number of retained coefficients.

For the four-body planetarium configurations, the validation problem becomes significantly more demanding. In both the equal-mass and different-mass cases, successful validations are obtained

only at relatively large Fourier truncation levels, with the outcome depending sensitively on the parameters of the a posteriori validation. For $M_{\mathrm{dim}} = 20$, the contraction factor $k$ is below unity but remains close to one, so that the contraction inequality cannot be closed using the standard radius selection and would require repeating the full validation procedure with a larger validation radius, with no guarantee of success. An alternative strategy is to increase the compact-operator dimension, thereby extending the finite-dimensional correction applied to the weakly or non-contracting directions of the linearised operator. This improves the quality of the approximate inverse primarily through a reduction of the contraction bound, while leaving the defect essentially unchanged at the relevant truncation levels. As a result, the contraction argument can be closed using the standard radius-selection rule.

It is worth noting that, across all tested configurations, increasing the truncation level $N$ systematically leads to smaller defects $\varepsilon$, while the contraction constant $k$ remains comparatively stable. This behaviour mirrors what was observed in the circular benchmark and indicates that the apparent validation difficulties at low $N$ are primarily a consequence of the DFFT a posteriori validation framework (see Section 7.4.3.3), which requires sufficient spectral resolution to produce a defect small enough to satisfy the contraction inequalities, rather than a genuine deterioration of the underlying contraction properties.

For the five-body planetarium orbits, no successful validation has been obtained so far, neither in the equal-mass nor in the different-mass case, for the truncation levels explored at the time of writing. In light of the systematic decrease of the defect $\varepsilon$ with increasing $N$ and the relative stability of the contraction constant $k$ observed in lower-dimensional cases, we expect that the present limitation is primarily computational in nature. In particular, the DFFT a posteriori validation (see Section 7.4.3.3) requires sufficiently large truncation levels to reduce the defect below the threshold imposed by the contraction inequalities. Due to time and resource constraints, it was not possible to carry out validations at significantly higher truncation levels within the scope of the present work. Further computations at larger $N$ are therefore expected to clarify the feasibility of these cases and are currently ongoing.

**Summary of the planetarium benchmark.** The planetarium benchmark provides a three-dimensional analogue of the circular tests and is designed to assess the feasibility of the CAPs validation procedure in a higher-dimensional setting. The results show that successful validation requires increasingly refined Fourier representations as the number of bodies grows. For three-body configurations, validation is achieved once the truncation level $N$ is sufficiently large, with defects $\varepsilon$ decreasing monotonically and contraction constants $k$ remaining stable. Four-body configurations already entail a substantial increase in computational cost and require a higher-quality approximate inverse to close the contraction argument. For five-body planetarium orbits, no successful validation has been obtained so far within the truncation levels explored, a limitation that appears to be primarily computational. Overall, these results confirm that, also in three dimensions, the performance of the CAPs framework depends primarily on the effective Fourier-space dimension and the quality of the approximate inverse, rather than on the geometric complexity of the orbit.

## 8.4 Discussion

This Chapter completes the final step of the Thesis pipeline by transforming periodic solutions obtained through the variational-optimisation framework into mathematically certified objects. Starting from high-accuracy Fourier representations, we instantiated the Banach-space proof structure and applied Algorithm 63 to obtain explicit defect bounds $\varepsilon$, contraction constants $k$, and validation radii $r$ guaranteeing the existence and local uniqueness of true periodic solutions of Newton's equations near the computed numerical seeds. The six representative examples Figures 8.1-8.6 show that the validation method applies uniformly across different symmetry groups, parity layouts, numbers of bodies, and mass distributions. The benchmark studies Figures 8.7 and 8.9 further

delineate the practical operating regime of the approach, indicating that computational costs scale mainly with the effective Fourier-space dimension and truncation level, and that genuinely three-dimensional validations may require higher spectral resolution or improved approximate inverses via larger compact-operator blocks $M_{\mathrm{dim}}$.

**Limitations and future work on CAPs.** While the results demonstrate that the proposed framework provides a robust and flexible tool for the rigorous validation of periodic solutions, they also clarify the present range of applicability of the method and point naturally toward further developments. From a computational standpoint, the CAPs validation step is substantially more demanding than the numerical generation of candidate orbits. Runtime grows with the Fourier truncation level $N$ and with the effective Banach-space dimension induced by parity structure, spatial dimension, and number of bodies. The dominant costs arise from nonlinear operator evaluations, rigorous tail bounds, and the finite-dimensional linear algebra associated with the compact-operator block of size $M_{\mathrm{dim}}$.These observations indicate that parallelisation offers the greatest potential for extending the range of feasible validations, while improvements in linear-algebra routines and adaptive parameter-selection strategies provide complementary benefits.

Furthermore, the CAPs procedure inherently depends on the quality of the numerical approximation provided as input, as is the case for any fixed-point validation method. Its role is not to generate periodic solutions, but to certify the existence of an exact solution in a neighbourhood of a given numerical seed. Consequently, if the variational or optimisation stage does not yield a sufficiently accurate approximation, the contraction argument may fail even when a true nearby solution exists. This behaviour reflects a limitation of the input data rather than of the CAPs methodology itself, while also suggesting that closer integration between numerical discovery and rigorous validation could further improve the robustness and efficiency of the overall pipeline.

Moreover, the distinction between periodicity and symmetry certification identifies a clear theoretical direction for extending the framework. While symmetry is exploited at the variational level to construct candidate solutions by restricting the search to equivariant loop spaces and fundamental domains, the current validation stage certifies periodicity rather than the persistence of symmetry constraints themselves. Developing computer-assisted techniques that explicitly certify symmetry properties, such as membership in a prescribed equivariant class or by exploiting algebraic relations between Fourier coefficients induced by symmetry, would further strengthen the connection between variational theory and rigorous validation and represents a natural continuation of the ideas developed in this Thesis.

In this context, an additional open direction concerns the analysis of solutions associated with symmetry groups of type S (see Definition 5.9). As discussed earlier in the Thesis, such symmetries lead to non-isolated critical points arising from continuous rotational invariance. A systematic exploration of these solutions would require a substantially more refined treatment, involving deeper algebraic analysis of symmetry groups, their normalisers, and the resulting stratification of equivariant loop spaces. Addressing these issues would necessitate more sophisticated mathematical tools than those employed in the present work and is therefore left for future investigation.

In conclusion, this Chapter shows that the combination of variational discovery and computer-assisted validation developed in this Thesis provides a reliable framework for the rigorous study of periodic solutions of the gravitational $n$-body problem. Numerical approximations obtained through symmetry-constrained optimisation are shown to admit fully certified counterparts, yielding explicit and reproducible existence bounds for true solutions of Newton's equations. Beyond the specific families of orbits validated here, the results demonstrate the generality and scalability of the approach and establish that rigorous computer-assisted proofs can be systematically embedded within variational and computational investigations in celestial mechanics. In this way, the Chapter completes the end-to-end methodology of the Thesis, linking high-precision computation to mathematically guaranteed existence results.

## 8.5 Conclusions

This Thesis has developed a unified variational–computational framework for the detection, classification, and rigorous validation of periodic solutions of the gravitational $n$-body problem. By combining tools from variational analysis, numerical optimisation, and computer-assisted proofs, the work bridges the gap between abstract existence theory and effective, certifiable computation.

**What has been done.** The first objective of this work was to design a systematic methodology for exploring the action functional associated with the $n$-body problem beyond classical minimisation approaches. This was achieved by discretising the variational problem through Fourier representations of periodic trajectories, enforcing symmetry and boundary constraints, and reducing the infinite-dimensional setting to a finite-dimensional optimisation problem. Within this framework, a hybrid optimisation strategy was introduced, coupling global stochastic methods with local deterministic refinement to detect both minimising and non-minimising critical points. In parallel, constructive variational techniques inspired by the Mountain Pass Theorem were implemented to locate saddle-type periodic orbits that cannot be obtained by direct minimisation. Beyond detection, an analysis phase was developed to characterise the qualitative and dynamical properties of the resulting periodic solutions. This includes the computation of discrete Morse indices to assess variational stability, together with geometric and landscape-based diagnostics that reveal the organisation and connectivity of critical points within the action landscape. Finally, a comprehensive computer-assisted proof (CAP) framework was developed, allowing numerical approximations of periodic solutions to be transformed into mathematically rigorous existence results with explicit error bounds, using validated numerics in Fourier series spaces.

**Main results.** The methods developed in this Thesis enabled the construction of an end-to-end framework for the study of periodic solutions of the gravitational $n$-body problem, covering their detection, classification, and rigorous validation. Within this framework, multiple classes of periodic solutions were systematically identified, including local minima and higher-index saddle points of the action functional. These solutions were characterised both locally and globally through the computation of discrete Morse indices, which describe the local variational structure of the action functional around each critical point, and through statistical analyses of the action landscape based on intra- and trans-level distances. For planar periodic orbits, additional topological information was extracted via winding numbers, providing a geometric classification of the detected families and complementing the variational analysis. Crucially, the framework culminates in a computer-assisted proof stage, where selected numerical solutions are rigorously validated, establishing the existence and uniqueness of exact periodic orbits in neighbourhoods of the computed approximations. This final step completes the methodological pipeline, transforming numerical candidates into certified mathematical results and ensuring full reproducibility and reliability of the entire discovery process.

**Perspectives and future work.** Several promising directions emerge from this work. A natural extension is the generalisation of the CAPs framework to higher-index critical points and more complex saddle configurations, enabling a rigorous exploration of the global topology of the variational landscape. In the same spirit, addressing the analytical difficulties associated with symmetry groups of type S (see Definition 5.9) would allow the full variational–validation pipeline developed in this Thesis to be reused. Once the non-isolated nature of the corresponding critical sets is properly handled, numerical detection and computer-assisted validation can proceed within the same framework. On the applied side, the proposed methodology provides a foundation for investigating periodic and resonant structures in astrophysical and astrodynamical contexts, including planetary systems, stellar clusters, and trajectory design problems. More broadly, the combination of variational principles, advanced optimisation algorithms, and rigorous validation techniques offers a flexible paradigm for the systematic and certified study of complex periodic phenomena in celestial mechanics and related nonlinear dynamical systems.

# Acknowledgements

To complete this Thesis, I wish to acknowledge the many people who have contributed to its realisation and to my personal and professional growth.

I am deeply grateful to my supervisor, Professor Susanna Terracini, for her continuous guidance and support throughout my doctoral studies. Her expertise and passion for celestial mechanics have been a constant source of inspiration and have profoundly shaped my approach to research. I also extend my sincere thanks to my co-supervisors, Professor Vivina Barutello and Professor Massimiliano Vasile, for their valuable advice, constructive feedback, and encouragement during the course of this work. Their insights have been instrumental in the completion of this Thesis and in my development as a researcher.

My gratitude also goes to the members of my research group and others, Mattia, Diego, Gian Marco, Irene, Pietro, Professor Gianni Arioli, Professor Davide Ferrario, and Professor Isabella Cravero, for their helpful discussions and constructive feedback. In particular, I thank my fellow PhD colleagues, Davide and Roberto, whose collaboration, thoughtful insights, and friendship have been invaluable; I could not have achieved as much without their support and the many moments of laughter we shared. I also sincerely thank the referees, Professor Bonnie Steves and Professor Joseph J. Masdemont, for their careful review and insightful suggestions.

I wish to thank my family, who has been a constant source of support and encouragement throughout my life. They have always pushed me to pursue my goals and have been there to console me in moments of difficulty. From them, I have learned not only perseverance and a strong sense of duty, but also the importance of humility, kindness, and respect towards others. In particular, I would like to express my profound gratitude to my sister, whose remarkable achievements as a researcher have been a continuous source of inspiration. Her dedication and example have motivated me to strive for excellence in my own path.

I would also like to thank my friends, whose presence and support have accompanied me throughout this journey. In particular, I am deeply grateful to Gabriele, Francesca, and Michaela for their encouragement, their timely hugs, and the countless evenings spent talking and enjoying each other's company. Their friendship has been an essential source of balance and joy during the more challenging stages of this work.

Last but not least, I wish to thank my better half, Robert. In your own unique way, Munro by Munro, you have taught me the importance of perseverance and optimism. Life, as you often remind me, is much like climbing a Munro: you never quite know what the weather will bring, how muddy the path might be, or whether there will be a path at all. Yet, if you *never give up never surrender*, reaching the cairn always makes the hike worthwhile.

This Thesis was produced while attending the PhD program in Space Science and Technology at the University of Trento, Cycle XXXVIII, with the support of a scholarship financed by Francesco Severi National Institute of High Mathematics (INdAM).

# Nomenclature

| | |
|---|---|
| $\dot{(\cdot)}$ | First time derivative (velocity) |
| $\ddot{(\cdot)}$ | Second time derivative (acceleration) |
| $(\cdot)^{\perp}$ | Orthogonal complement |
| $(\mathbf{s_F})^{(i)}$ | $i$-th component of a vector-valued series $\mathbf{s_F}$ |
| $(\mathbb{R}^d)^n$ | Product configuration space |
| $(f^+, h^+)$ | Pair associated with the trial point $x^+$ |
| $(f_k, h_k)$ | Pair of objective value and infeasibility measure associated with $x_k$ |
| $(n_1, \ldots, n_k)$ | Cycle (cyclic permutation) sending $n_1 \mapsto n_2 \mapsto \cdots \mapsto n_k \mapsto n_1$ |
| $1$ | Identity element when the group is clear from context |
| $1/s_F$ | Reciprocal of a Fourier series $s_F$ |
| $1_G$ | Identity (unit) element of the group $G$ |
| $[S^-, S^+]$ | Interval enclosing a real number with machine-representable endpoints |
| $[x]$ | Time average of a loop $x$, $\frac{1}{T}\int_0^T x(t)\,\mathrm{d}t$ |
| $[y]^-$ | Negative part of $y$: $[y]^- = \max\{-y, 0\}$ |
| $\alpha_{\min}$ | Minimum step length triggering feasibility restoration |
| $\bar{G}$ | Quotient group $\bar{G} \coloneqq G/\ker\tau$ |
| $\bar{n}$ | Truncation index defining the finite-dimensional space $H_{\bar{n}}$ |
| $\coloneqq$ | Definition symbol ("is defined as") |
| $\Delta$ | Set of all collision configurations |
| $\Delta_k$ | Trust-region radius at iteration $k$ |
| $\Delta_{ij}$ | Configurations where bodies $i$ and $j$ collide |
| dim | Spatial dimension (winding-number computation assumes dim = 2) |
| $\ell$ | Truncation dimension used to represent operators/matrices on a finite subspace (stored as `M_dim`) |
| $\varnothing$ | Empty set (used e.g. when $\hat{\Lambda}^G = \varnothing$) |
| $\eta$ | Acceptance threshold parameter in trust-region algorithms |

| | |
|---|---|
| $\eta_f(t, x)$ | Steepest descent flow associated with $f$ starting at $x$ and evolved for time $t$ |
| $\Gamma$ | Feasible set (points satisfying all constraints) |
| $\gamma$ | Trajectory (path) in configuration space |
| $\gamma^k$ | $k$-th iterate of a periodic orbit $\gamma$ |
| $\hat{\Delta}$ | Maximum allowable trust-region radius |
| $\hat{\Lambda}$ | Space of collision-free $H^1$ $T$-periodic loops |
| $\hat{\Lambda}^G$ | Space of collision-free $G$-equivariant loops |
| $\hat{\chi}$ | Collision-free configuration space |
| $\mathbb{I}$ | Fundamental domain (subinterval) for the action of $G$ on $\mathbb{T}$ |
| Id | Identity transformation |
| $\mathfrak{I}(t)$ | Imaginary part of a complex number $t$ |
| $\ker \tau$ | Kernel of the time-shift homomorphism $\tau$ (elements acting trivially on time) |
| $\Lambda$ | Space of $H^1$ $T$-periodic loops in $\chi$ |
| $\lambda$ | Vector of Lagrange multipliers |
| $\lambda^\star$ | Lagrange multiplier vector at a solution |
| $\Lambda^G$ | Space of $G$-equivariant $H^1$-periodic loops |
| $\lambda^k$ | Lagrange multiplier vector at iteration $k$ |
| $\lambda_1$ | Most negative eigenvalue of the matrix $B_k$ |
| $\lambda_i$ | Lagrange multiplier vector associated with constraint $i$ |
| $\lambda_i^\star$ | Lagrange multiplier associated vector with constraint $i$ at a solution |
| $\langle g \rangle$ | Cyclic subgroup generated by the element $g$ |
| $\langle S \rangle$ | Subgroup of $G$ generated by the subset $S \subseteq G$ |
| $\langle \cdot, \cdot \rangle$ | Euclidean inner product |
| $\mathbb{E}^-(A)$ | Negative eigenspace of the Hessian operator $A$ |
| $\mathbb{R}_+$ | Nonnegative real numbers (time domain for the flow) |
| $\mathbb{T}$ | One-dimensional torus $\mathbb{R}/\mathbb{Z}$ |
| $\mathbf{0}$ | Zero matrix of dimension $(n \cdot dim) \times (n \cdot dim)$ |
| $\mathbf{s_F}$ | Vector-valued Fourier series $(s_F^{(1)}, \ldots, s_F^{(d)})$ |
| $\mathbf{s_{1,F}}, \mathbf{s_{2,F}}$ | Vector-valued Fourier series in $\mathcal{F}_\rho^d$ |
| $\mathcal{A}$ | Lagrangian action functional |
| $\mathcal{A}(\gamma)$ | Lagrangian action functional evaluated along a path $\gamma$ |
| $\mathcal{A}_h^{(1)}$ | Discrete approximation of $\mathcal{A}_\mathbb{I}$ obtained via quadrature + finite differences |

| | |
|---|---|
| $\mathcal{A}_h^{(2)}$ | Discrete approximation of $\mathcal{A}_{\mathbb{I}}$ in the Fourier-coefficient model |
| $\mathcal{A}_\rho$ | Odd subspace of $\mathcal{F}_\rho$ |
| $\mathcal{A}_{\mathbb{I}}$ | Action functional restricted to the fundamental domain $\mathbb{I}$ |
| $\mathcal{A}_{\mathbb{I}}^{\text{reg.}}$ | Regularised action functional on $\mathbb{I}$ (smooth $\mathcal{C}^2$ functional) |
| $\mathcal{B}_\rho$ | Even subspace of $\mathcal{F}_\rho$ |
| $\mathcal{D}(A)$ | Domain of the Hessian operator $A$ |
| $\mathcal{E}$ | Index set of inequality constraints |
| $\mathcal{F}_\rho^P$ | Direct-sum / vector-valued extension of $\mathcal{F}_\rho$ with $P$ components |
| $\mathcal{F}_\rho$ | Banach space of $2\pi$-periodic analytic functions on $\mathcal{D}_\rho$ with exponential Fourier decay |
| $\mathcal{F}_\rho^d$ | Vector-valued analytic Fourier space of dimension $d$ endowed with the max-norm with analytic radius $\rho$ |
| $\mathcal{F}_i^c$ | Basin of attraction of $F_i^c$, i.e. $\{x \in X : \omega_x \subset F_i^c\}$ |
| $\mathcal{I}$ | Index set of equality constraints |
| $\mathcal{L}(x, \lambda)$ | Lagrangian function |
| $\mathcal{L}_A(x, \lambda; \mu)$ | Augmented Lagrangian function (equality-constraint form) |
| $\mathcal{M}$ | Set of local minimum points detected during the optimisation |
| $\mathcal{N}(x)$ | Residual map used in Newton-like validation (e.g. $\mathcal{N}(S) = S^2 - s_F$ in DFFT) |
| $\mathcal{P}_\ell$ | Canonical projection onto Fourier polynomials of degree $\leq \ell$ (truncation projector) |
| $\mathfrak{m}_{I_k}$ | Subset of minima with objective values in $I_k$ (level set) |
| $(\text{PS})_c$ | Palais–Smale condition at level $c$ (every PS sequence at level $c$ has a convergent subsequence) |
| $\text{Antiderivative}^2$ | Second antiderivative operator acting componentwise on Fourier series |
| $\text{crit}(f)$ | Set of critical points of the functional $f$ |
| $\mathrm{d}\mathcal{A}$ | Differential of the action functional $\mathcal{A}$ |
| Id | Identity operator (or matrix, when finite-dimensional), acting on the appropriate space (e.g. $\mathbb{R}^{n \cdot \dim}$ or $\mathcal{F}_\rho^d$) |
| $\text{span}[\cdot]$ | Linear span of a set of vectors |
| $\mu$ | Penalty parameter |
| $\mu_k$ | Penalty parameter at iteration $k$ |
| $\nabla f$ | Gradient of the functional $f$ |
| $\nabla f(x_k)$ | Gradient of $f$ at $x_k$ |
| $\nabla^2 U(t_i)$ | Hessian of the potential term $U$ evaluated at time node $t_i$ |

| | |
|---|---|
| $\nabla_x \mathcal{L}$ | Gradient of the Lagrangian w.r.t. the primal variable $x$ |
| $\mathbb{S}$ | Circle in $\mathbb{R}^d$ centered at the origin |
| $\omega(t)$ | Angular-velocity integrand used to compute the winding number |
| $\Omega(x)$ | Active set at a feasible point $x$ |
| $\Omega^\star$ | True active set at a local solution (constraints active at the optimum) |
| $\Omega^{\bar{G}}$ | $\bar{G}$-equivariant loop space $H^1(\mathbb{T}; \chi^{\ker \tau})^{\bar{G}}$ |
| $\omega_x$ | $\omega$-limit set of $x$ under $\eta_f$ (limit points of $\eta_f(t, x)$ as $t \to +\infty$) |
| $\mathrm{std}(X)$ | Set of standard sets associated with a space $X$ |
| $\mathrm{std}(X_1 \times X_2)$ | Cartesian product of standard sets in product spaces |
| $\partial \mathcal{F}_1^c$ | Boundary of the basin of attraction $\mathcal{F}_1^c$ |
| $\partial / \partial x_i$ | Partial derivative with respect to $x_i$ |
| $\partial^{-2}$ | Second antiderivative operator on $2\pi$-periodic functions with zero average |
| $\Phi$ | Nonlinear analytic map applied to a Fourier series (e.g. $\Phi(z) = \sqrt{z}$ or $\Phi(z) = 1/z$) |
| $\phi$ | Penalty function applied to inequality constraint violations |
| $\phi(g, x)$ | Action of the group element $g \in G$ on $x \in X$ |
| $\phi_k(t)$ | Fourier basis function (sine or cosine, depending on parity) |
| $\Pi$ | Permutation matrix in the QR factorisation with pivoting |
| $\Pi_0$ | Projection onto the constant (zero) Fourier mode |
| $\pi_c$ | Projector enforcing the centre-of-mass constraint (projection onto $\chi$) |
| $\pi_g$ | Projector enforcing cyclic boundary constraint in the cyclic-action case |
| $\pi_H$ | Projector onto $X^H$ defined by averaging over the subgroup $H$ |
| $\pi_i$ | Projector onto $\chi^{H_i}$ (endpoint constraint projector) |
| $\pi_{\ker \tau}$ | Projector onto $\chi^{\ker \tau}$ |
| $\psi$ | Depending on the context, either the action of the Weyl group $W_G H$ on $X^H$, or a penalty function applied to equality constraint violations |
| $\psi(t)$ | Fourier-truncated correction term with $\psi(0) = \psi(\pi) = 0$ |
| $\psi_k(t)$ | Fourier correction term vanishing at the endpoints in the Bolza ansatz |
| $\mathbb{R}^d$ | $d$-dimensional Euclidean space |
| $\chi$ | Configuration space with vanishing centre of mass |
| $\chi^G$ | Set of configurations fixed by the action of $G$ |

| | |
|---|---|
| $\rho$ | Depending on the context, either the strip-width (analyticity scale) parameter in the definition of $\mathcal{F}_\rho$ and its norm, or the orthogonal representation $\rho : G \to O(d)$ |
| $\rho_g$ | Spatial (orthogonal) action associated with the element $g$ |
| $\rho_k$ | Ratio of actual to predicted reduction in trust-region methods |
| $\rho_{\mathrm{IL}}(x_i)$ | Intra-level distance of $x_i$: mean distance to other minima in the same level |
| $\rho_{\mathrm{TL}_k}(x_i)$ | Trans-level distance of $x_i$: mean distance to minima in the next lower level |
| $\sigma$ | Permutation representation of $G$ on bodies indices, $\sigma : G \to \Sigma_n$ |
| $\Sigma_n$ | Symmetric group of degree $n$, group of permutations of $\{1, \dots, n\}$ |
| $\Sigma_X$ | Group of all permutations of the set $X$ |
| $\sum_{i=0}^{h} X^i$ | Truncated geometric series approximating the reciprocal |
| sup | Supremum (maximum along a path) |
| $\mathbb{T}$ | Time circle $\mathbb{R}/T\mathbb{Z}$ |
| $\tau$ | Depending on the context, either the orthogonal representation $\tau : G \to O(2)$ on the time circle, or a geometric tolerance used to identify and discard near-duplicate minima |
| $\tau(G)$ | Image of the representation $\tau$, a finite subgroup of $O(2)$ |
| $\tau_g$ | Time-action associated with the element $g$ |
| $\tau_k$ | Optimal scaling parameter defining the Cauchy point |
| $\mathfrak{A}$ | Archive of detected local minima and associated restart states (MP-AIDEA) |
| `Proof` | Constructor assembling the proof structure $P = (space, trunc\_space, F, DF, M\_dim, R\_an, r)$ |
| $\theta$ | Rotation parameter for $R_\theta$ |
| $\tilde{\Xi}$ | Union of subregions after a split, $\tilde{\Xi} = \Xi_1 \cup \Xi_2$ |
| $\tilde{n}^-(x)$ | Discrete Morse index of a critical point $x$ |
| $\tilde{s}_F$ | Normalised series $\tilde{s}_F = s_F / c_0$ |
| $\mathrm{D}F(x)$ | Fréchet differential of the functional $F$ at $x$ |
| $\mathrm{D}^2F(x)$ | Second Fréchet differential of the functional $F$ at $x$ |
| $\varepsilon$ | Depending on the context, either an action tolerance used to build level intervals around $f_k$, or a regularisation parameter in the softened potential $U_\varepsilon$ |
| $\varepsilon + kr < r$ | Compatibility inequality ensuring $C(B(u_0, r)) \subseteq B(u_0, r)$ and contractivity |
| $\widehat{M}$ | Finite matrix used to assemble $M = \mathcal{P}_\ell \widehat{M} \mathcal{P}_\ell$ on the truncated subspace |
| $\Xi$ | Optimisation domain (parameter domain) |
| $\Xi = I \times J$ | Two-dimensional parameter domain in the illustrative example |

| | |
|---|---|
| $\zeta$ | Complex coordinate along the curve $\gamma$ |
| $\{B_n\}_{n\geq 1}$ | Basis elements of a series space (canonical generators supported on a single mode or tail slot) |
| $A$ | Depending on the context, either a bounded preconditioner (approximate inverse) used in the DFFT Newton operator $G$, or a constraint matrix in linear equality constraints $Ax = b$ |
| $a$ | Group action of $G$ on $X$: the induced homomorphism $a : G \to \Sigma_X$, where for each $g \in G$, $a(g)$ is the action map defined by $a(g)x = gx$ |
| $A(x)$ | Jacobian of the equality constraint map $c(x)$ |
| $A_k$ | Fourier coefficients in the truncated Fourier expansion |
| $b$ | Right-hand side vector in linear constraints $Ax = b$ |
| $B_k$ | Context-dependent symbol: symmetric Hessian approximation, or Fourier coefficient in a truncated Fourier expansion |
| $C$ | Flat storage vector containing Fourier coefficients of a series |
| $c$ | Energy level $c > 0$ (Palais–Smale), corresponding to a disconnected sublevel set $f^c$ |
| $C(u)$ | Newton-like corrected map $C(u) = F(u) - M\,[F(u) - u]$ used to enforce contractivity |
| $c_0$ | Context-dependent notation: constant Fourier coefficient (zero mode) or mountain-pass level $c_0 = \inf_{\gamma\in\Gamma_{x_1,x_2}} \sup_{s\in[0,1]} f(\gamma(s))$ |
| $c_i(x)$ | Equality constraint function indexed by $i$ |
| $d$ | Ambient physical space dimension |
| $D(Q_1(x;\mu);p)$ | Directional derivative of $Q_1$, the $\ell_1$-based merit function, at $x$ along direction $p$ |
| $d_i(x)$ | Inequality constraint function indexed by $i$ |
| $D_{2n}$ | Dihedral group, symmetry group of an $n$-gon, of order $2n$ |
| $DF(\cdot)$ | Fréchet derivative (Jacobian) of a map at the indicated point; e.g. $D\mathcal{N}(S)$ for the nonlinear operator $\mathcal{N}$ at $S$, $DF(u)$ for $F$ at $u$, and $DC(u)$ for the corrected map $C$ at $u$ |
| $DF(u_0) - I$ | Linearised fixed-point operator on the truncated subspace; must be invertible to define $M$ |
| $DF(x)[v]$ | Action of the Fréchet derivative of a map at $x$ on a direction $v$ |
| $DF(x)[v]_i$ | Componentwise evaluation of the Jacobian action $DF(x)[v]$ at index $i$ |
| $dim$ | Spatial dimension (number of coordinates per body) |
| $dt$ | Discrete time increment used for numerical time integration |
| $E$ | Flat storage vector containing error/tail enclosures associated with the series |

| | |
|---|---|
| $e_k^{(i)}$ | Canonical vector basis element with mode $k$ in component $i$ |
| $F$ | Number of retained Fourier modes, i.e. the truncation order/length in a truncated Fourier series or polynomial |
| $f$ | Objective (cost) functional; a generic function or functional—e.g. an action functional, possibly discretised, whose critical points or minima are sought and used to rank solutions |
| $F(t)$ | Fourier series representing a function |
| $F(u) = u$ | Fixed-point equation posed on the Banach space $\mathcal{F}_\rho^d$ |
| $F(x)$ | Integrated fixed-point map associated with Newton's equations used in the CAP validation |
| $f^c$ | Sublevel set at level $c$, $f^c = \{x \in X : f(x) \le c\}$ |
| $f^H$ | Restriction of a $G$-equivariant map to fixed-point sets $X^H \to Y^H$ |
| $f_1$ | Objective function corresponding to $\mathcal{A}_h^{(1)}$ |
| $f_2$ | Objective function corresponding to $\mathcal{A}_h^{(2)}$ |
| $F_i^c$ | Connected component of the sublevel set $f^c$ |
| $f_k$ | Objective function evaluated at $x_k$, $f_k = f(x_k)$ |
| $G$ | Finite group, typically acting as a symmetry group on the space of admissible configurations |
| $G(t)$ | Discrete-time integrand combining kinetic and potential terms (used in quadrature) |
| $G(x)$ | Newton-like operator $G(x) = x - A\mathcal{N}(x)$ used in DFFT a posteriori validation |
| $G/H$ | Quotient group, acting naturally on $X^H$ if $H \lhd G$ |
| $g = (\rho, \tau, \sigma)$ | Generic element of the symmetry group $G$ (representation on space, time, and permutations) |
| $g^{-1}$ | Inverse of the element $g \in G$ |
| $G_\bullet$ | Isotropy (stabiliser) subgroup under the relevant group action; e.g. $G_x = \{g \in G : gx = x\}$ or $G_t = \{g \in G : \tau(g)t = t\}$ |
| $gH$ | Left coset of $H$ in $G$ |
| $H$ | Context-dependent notation: either (i) a subgroup $H \le G$, (ii) the Sobolev space $W^{1,2}(\mathbb{T}, \hat{\chi})$ of periodic trajectories, or (iii) a Hilbert space equipped with an increasing family of finite-dimensional subspaces $H_n \subset H$ used in discrete Morse index constructions |
| $h$ | Context-dependent discretisation parameter: either the truncation order of a Taylor/binomial expansion, or a step size in time or space discretisation (e.g. $h = \frac{T}{w-1}$ or $h = \frac{\pi}{M+1}$), depending on context |
| $h(x)$ | Measure of infeasibility (sum of constraint violations) |

| | |
|---|---|
| $H_0, H_1$ | Maximal $\mathbb{T}$-isotropy subgroups associated with $t_0$ and $t_1$ |
| $h_0, h_1$ | Generators of $H_0$ and $H_1$ (cyclic of order 2 in brake/dihedral case) |
| $H_f(\cdot)$ | Hessian matrix of the function $f$ evaluated at the indicated point (e.g. $H_k(x_k)$ or $H_f(p)$) |
| $H_n$ | Finite-dimensional subspace of $H$ used for Morse index discretisation |
| $Hg$ | Right coset of $H$ in $G$ |
| $I$ | Symbol whose meaning depends on context: index form, identity matrix, or interval $I = [a, b]$ in $\mathbb{R}$ |
| $i$ | Imaginary unit or index, depending on context |
| $I_k$ | Action interval defining a level around $f_k$, $I_k = [f_k - \varepsilon, f_k + \varepsilon]$ |
| $I_M$ | Interpolation operator mapping Fourier coefficients to grid values |
| $I_{\bar{n}}$ | Restriction of the index form to the space $H_{\bar{n}}$ |
| $i_{\mathrm{Morse}}(\gamma)$ | (Continuous) Morse index of an orbit $\gamma$ |
| $j$ | Index variable (layer/recursion index in algorithmic contexts) |
| $J = [c, d]$ | Interval domain for a parameter (example in $\mathbb{R}$) |
| $K$ | Context-dependent notation: either the total number of detected levels (clusters) after partitioning by action or a compact operator appearing in the decomposition $\nabla \mathcal{A}_{\mathbb{I}}^{\mathrm{reg.}} = \mathrm{Id} + K$ |
| $K(\cdot)$ | Kinetic energy term in the Lagrangian |
| $k^{-n}$ | Scaling factor applied to the $k$-th Fourier mode under $n$ integrations |
| $K_i$ | Isotropy subgroup of the index $i$ (within a given subgroup action) |
| $L$ | Context-dependent notation: either the Lagrangian function (typically kinetic minus potential energy, or kinetic plus $U$ under the chosen convention), or a list (archive) of known critical points used by the Mountain Pass Algorithm |
| $l$ | Number of distinct $\mathbb{T}$-isotropy subgroups of $G$ |
| $L(x, \dot{x})$ | Lagrangian of the system |
| $L_\ell$ | Representative action value of level $\ell$ used in level construction |
| $L_\varepsilon(y, \dot{y})$ | Regularised Lagrangian $L_\varepsilon = K(\dot{y}) + U_\varepsilon(y)$ |
| $M$ | Context-dependent notation: either (i) a discretisation parameter such as the number of grid subintervals or grid size (e.g. in time discretisation or FFT/DCT evaluation), (ii) a bounded linear operator or its finite matrix representation used as a Newton-like inverse-Jacobian correction on a truncated subspace (e.g. in the construction of a corrected map $C$), or (iii) a cardinality, such as the number of detected local minima |
| $m$ | Number of linear equality constraints (rows of $A$ in the linear constraint elimination: $Ax = b$) |

| | |
|---|---|
| $m_i$ | Context-dependent notation: either a detected local minimiser (element of $\mathcal{M}$), or the mass of the $i$th body ($m_i > 0$) |
| $m_k$ | Local quadratic model of the objective function at $x_k$ |
| $M_{\mathrm{dim}}$ | Truncation level for the finite-dimensional matrix representation used in the Newton-like correction |
| $N \triangleleft G$ | Normal subgroup $N$ of $G$ |
| $N$ | Context-dependent notation: either (i) the non-basis submatrix of a constraint matrix $A$ containing the remaining columns in linear constraint elimination ($Ax = b$), or (ii) the number of stored Fourier coefficients in a truncated representation |
| $n$ | Context-dependent notation: either (i) number of bodies, or (ii) order of the antiderivative |
| $n(\gamma, a)$ | Winding number of $\gamma$ around $a$ |
| $n^-(x)$ | Morse index of a critical point $x$ |
| $N_G H$ | Normaliser of $H$ in $G$ |
| $N_{\mathrm{A}}$ | Number of agents per population in MP-AIDEA |
| $N_{\mathrm{P}}$ | Number of populations in MP-AIDEA |
| $N_{\mathrm{S}}$ | Number of stages in the DE phase (per population) |
| $O$ | Operator acting on a series space (scalar or vector) |
| $O(2)$ | Orthogonal group of $2 \times 2$ real orthogonal matrices |
| $P$ | Context-dependent notation: either (i) a dimension parameter (number of components in a vector-valued series space), (ii) a structured object bundling data for proofs or algorithms (e.g. a contraction-proof specification such as $P = (space, trunc\_space, F, DF, M\_dim, R\_an, r)$ or a problem structure collecting $f$, $\nabla f$, $\eta_f$, $L$, and $x_{\min}$), or (iii) a permutation matrix used to reorder variables in linear constraint elimination |
| $p$ | Context-dependent notation: either a $p$-value used in statistical tests (e.g. the chi-square uniformity test in the Lattice split criterion), or a non-degenerate critical point of the objective function $f$ |
| $p = (p_1, p_2)$ | Chosen reference point for winding computation (typically the centre of mass) |
| $P_h(X)$ | Truncated binomial polynomial $\sum_{i=0}^{h} \binom{m}{i}(-X)^i$ |
| $p_k$ | Search direction or trial step at iteration $k$ |
| $p_k^B$ | Full (Newton-type) step, $p_k^B = -B_k^{-1} g_k$ |
| $p_k^c$ | Cauchy point at iteration $k$ |
| $p_k^s$ | Steepest-descent direction in the trust-region subproblem |
| $p_k^U$ | Unconstrained minimiser of the quadratic model along $-g_k$ |
| $P_N$ | Projection operator truncating to the first $N$ Fourier modes |

| | |
|---|---|
| $P_{\text{in}}$ | Parity of the input Fourier space |
| $P_{\text{out}}$ | Parity of the output Fourier space after $n$ antiderivatives |
| $Q$ | Orthogonal matrix in the QR factorisation |
| $q(p)$ | Quadratic model of a merit function used for sufficient decrease tests |
| $q(t)$ | Unknown periodic solution of the differential equation |
| $Q(x;\mu)$ | Penalty function (penalised objective) |
| $Q_1(x;\mu)$ | $\ell_1$ exact penalty function |
| $Q_1, Q_2$ | Orthonormal blocks in the QR factorisation of $A^T$ |
| $Q_2(x;\mu)$ | $\ell_2$-based (nonsmooth) exact merit function for equality constraints |
| $Q_F(x;\mu)$ | Fletcher augmented Lagrangian merit function |
| $q_i$ | Context-dependent notation: either the position of body $i$ (e.g. in the three-body problem), or the $i$th component of a vector-valued Fourier series $q \in \mathcal{F}_\rho^P$ |
| $R$ | Context-dependent notation: either (i) the upper-triangular factor in a QR factorisation, (ii) an analyticity radius parameter for Fourier coefficients (sometimes written also as $R_{\text{an}}$), or (iii) a DFFT-based approximation obtained after nonlinear evaluation and projection |
| $r$ | Context-dependent notation: either the radius of a validation ball (e.g. $B(u_0, r)$ in a Banach fixed-point argument), or a rotation generator of the dihedral group $D_{2n}$ |
| $R^{-T}$ | Inverse transpose of $R$ (used in triangular substitution) |
| $R_\theta$ | Rotation by angle $\theta$: either the rotation matrix in $O(2)$ about the origin, or a one-parameter family of rotations preserving $Y^G$ (e.g. in the type-S case) |
| $R_{\text{an}}$ | An analyticity radius parameter for Fourier coefficients |
| $R_{h+1}(X)$ | Remainder of the binomial series after truncation at order $h$ |
| $s$ | Context-dependent notation: either (i) a reflection generator of the dihedral group $D_{2n}$, (ii) a path parameter $s \in [0,1]$ used to traverse a path $\gamma(s)$, or (iii) a global sign factor arising from parity alternation under repeated integration |
| $s(t)$ | Truncated Fourier perturbation of the linear path |
| $S^-, S^+$ | Lower and upper bounds of an interval |
| $S^{(i)}$ | $i$-th Newton iterate for a nonlinear series equation |
| $S_\theta$ | Reflection matrix in $O(2)$ across a line forming angle $\theta$ with the $x$-axis |
| $s_F$ | Series object representing a Fourier series in a given Fourier space |
| $s_F^n$ | $n$-th integer power of a Fourier series $s_F$ under validated multiplication |
| $s_F^{(i)}$ | $i$-th scalar component of a vector-valued Fourier series |
| $s_F^m$ | Rational power of a Fourier series $s_F$, with exponent $m = \frac{1}{n} > 0$ |

| | |
|---|---|
| $S_{k,h}$ | Quadratic bound $E_k^F E_h^G$ for error–error contributions in multiplication |
| $SO(d)$ | Special orthogonal group in dimension $d$ |
| $T$ | Context-dependent notation: either (i) a temperature parameter controlling acceptance of non-improving moves (e.g. in simulated annealing), or (ii) a time-scale parameter such as the period of an orbit or loop, the length of the time circle, or the time horizon of an interval, depending on context |
| $t$ | Time variable |
| $t_0, t_1$ | Endpoints of a parameter interval, e.g. the fundamental domain $\mathbb{I} = [t_0, t_1]$ or the initial and final parameter values for one traversal of a path $\gamma$ |
| $t_i$ | Discrete time grid point on a parameter interval (e.g. $[0, \pi]$ or $\mathcal{I}$), indexed by $i$ (notation also used as $t_h$ or $t_k$ depending on context) |
| $T_x W^{1,2}(\mathbb{T}, \hat{\chi})$ | Tangent space to the Sobolev manifold at $x$ |
| $T_x \chi$ | Tangent space to configuration space at $x$ |
| $U(\cdot)$ | Newtonian potential energy |
| $U = (U_1, \dots, U_n)$ | Interval enclosures for the first $n$ Fourier coefficients (low modes) |
| $U_k^{(i)}$ | Interval enclosure for the $k$-th stored coefficient of component $i$ |
| $u_0$ | High-accuracy numerical approximation of the fixed point |
| $U_k$ | Interval enclosure for the $k$-th stored Fourier coefficient |
| $u_k$ | Selected (exact) coefficient within interval enclosure $U_k$ |
| $U_\varepsilon(y)$ | Regularised (softened) potential term |
| $V$ | Linear subspace of $\mathbb{R}^d$ |
| $v$ | Tangent/perturbation direction used in evaluating the Jacobian action |
| $V = (V_n, V_{n+1}, \dots)$ | Interval description of the tail coefficients (high modes) |
| $V^G$ | Fixed subspace in configuration space, $V^G = \{x \in \hat{\chi} : \rho x = x\}$ |
| $V^g$ | Fixed-point subspace of $\rho(g)$ in $\mathbb{R}^d$, $V^g = \{v \in \mathbb{R}^d : \rho(g)v = v\}$ |
| $V_m^{(i)}$ | Tail/error enclosure for component $i$ beyond the truncation index |
| $V_g$ | Constraint set $V_g = \{(v, w) : gv = w\}$ for cyclic boundary conditions |
| $V_m$ | Tail/error set (enclosure) accounting for contributions beyond the truncation index |
| $v_m$ | Selected (exact) tail coefficient within interval enclosure $V_m$ |
| $W$ | Working set (current estimate of the active constraints in active-set methods) |
| $w$ | Context-dependent notation: either the number of discretisation time nodes over a time interval, or an integer winding number |
| $W^{1,2}(\mathbb{T}, \hat{\chi})$ | Sobolev manifold of $H^1$-periodic loops in $\hat{\chi}$ |

| | |
|---|---|
| $W_G H$ | Weyl group of $H$ in $G$, defined as $N_G H/H$ |
| $w_m$ | Analytic weight sequence used for tail bookkeeping in multiplication |
| $w_{\mathrm{num}}$ | Numerical approximation of the winding number before rounding |
| $x = (x_1, \ldots, x_n)$ | Configuration of the system |
| $X \star Y$ | Interval extension of a binary operation $\star$ |
| $X$ | Ambient Hilbert (or metric) space where the functional $f$ is defined |
| $X = 1 - \alpha s_F$ | Deviation used in the geometric expansion for the reciprocal |
| $X = 1 - \tilde{s}_F$ | Deviation used in binomial expansion |
| $x^\star$ | Best-known solution (reference point for the lowest level) |
| $X^H$ | Set of points of $X$ fixed by all elements of the subgroup $H \leq G$ |
| $x^+$ | Trial point (candidate iterate) in line-search/trust-region methods |
| $x^{-1}\Delta$ | Collision times of a loop $x$, i.e. preimage of $\Delta$ in $\mathbb{T}$ |
| $X_0$ | Positively invariant subset of $X$ under the flow $\eta_f$ |
| $x_0, x_1$ | Context-dependent notation: a pair of distinguished configurations, used either as endpoint configurations of a parameter interval (e.g. at $t = 0$ and $t = \pi$ in a Fourier-coefficient model), or as initial points for the Mountain Pass algorithm |
| $x_0^s$ | Initial starting point for the first inner minimisation |
| $x_1, x_2$ | Two distinct points in $X$ used to define the mountain-pass path class |
| $x_B$ | Basic variables associated with the basis matrix $B$ |
| $x_i$ | Position of body $i$ (vector in the ambient physical space) |
| $x_i^k$ | Position of body $i$ at discrete time node $k$ |
| $x_k$ | Current iterate in an optimisation algorithm |
| $x_k^s$ | Starting point for the inner minimisation at iteration $k$ |
| $x_N$ | Non-basic variables (free variables) after elimination |
| $x_Y$ | Reduced coordinates associated with $Y$ in the representation $x = Y x_Y + Z x_Z$ |
| $x_Z$ | Reduced (free) coordinates associated with $Z$ |
| $x_{\min}$ | Current reference minimum point stored in the problem structure $P$ |
| $x_{k+1}$ | Next iterate in an optimisation algorithm |
| $Y$ | Context-dependent notation: either (i) a functional space of admissible curves (e.g. the space on which $\mathcal{A}_{\mathcal{I}}^{\mathrm{reg.}}$ is defined, including fixed-end path spaces with endpoint constraints), or (ii) a matrix defining a particular solution mapping in linear constraint elimination (as in $x = Y b + Z x_N$) |
| $y(t)$ | Initial or optimised path on the fundamental domain |

| | |
|---|---|
| $Y^G$ | Fixed-point subspace of $Y$ under the action of $G$ (i.e. $G$-symmetric configurations) |
| $Y_F$ | Finite-dimensional approximation space for optimisation (Fourier-truncated model) |
| $y_i^k$ | Discrete approximation of $x_i(t_k)$ |
| $y_i^{M+2}$ | Ghost point used for finite-difference boundary handling |
| $Z$ | Null-space basis matrix satisfying $AZ = 0$ |
| $z_c$ | Objective function value at the current solution (simulated annealing) |
| $z_n$ | Objective function value at a neighbouring candidate solution (simulated annealing) |
| `AbstractOperator{S}` | Abstract operator type parameterised by a series space $S$ |
| `Antiderivative_Complete` | Routine computing the antiderivative in a complete Fourier space |
| `Antiderivative_Parity` | Routine computing the antiderivative in an even or odd Fourier space |
| `Apply` | Operator application routine returning $O.f(u)$ |
| `Base.:*` | `Julia` overload implementing validated multiplication of `Series` objects |
| `Base.:/` | `Julia` overload implementing validated scalar–series division |
| `Base.:^` | `Julia` overload implementing integer powers of a `Series` |
| `Base.:integrate` | `Julia` overload implementing validated antiderivatives of `Series` |
| `Base.:sqrt` | `Julia` overload computing the square root of a `Series` |
| `Base.copy` | Overloaded method returning a deep copy of a series or vector series object |
| `Base.length` | `Julia` overload returning the dimension $d$ of a vector series space |
| `Base.one` | Overloaded constructor returning the multiplicative identity series in a given space |
| `Base.zero` | Overloaded constructor returning the additive identity series in a given space |
| `Basis` | Iterator over canonical basis elements of a series space (including tail slots when infinite) |
| `Bisection` | Auxiliary routine that bisects the segment between $x_0$ and $x_1$ and applies `Gradient_descent` |
| `Check_compatible_spaces_sum` | Routine testing componentwise compatibility of two vector series spaces for addition/subtraction |
| `cidx_shift` | Parity-dependent shift mapping coefficient storage indices to frequencies |
| `cidx` | Helper returning the storage index window of the coefficient block |
| `Cmap` | Mapping from global coefficient indices to component-local indices |
| `coefftype` | Helper returning coefficient type of a series space |

`CompactOperator{S,MT}` Type storing a compact operator on space $S$ with matrix type $MT$

`CompactOperator` Finite-matrix representation of a (compact) linear operator on a chosen truncated basis

`Compute_FFT_Error` A posteriori validation routine for DFFT output

`Compute_weights` Procedure generating weight array $w$ used in tail allocation

`Cstart` Cumulative offsets for coefficient blocks of vector components

`CT` Coefficient type used in a series space (e.g. `Float64`, interval type)

`C` Coefficient vector storing the finite Fourier modes of a `Series`

`DF` Stored callable implementing the Jacobian action $(u, h) \mapsto DF(u)[h]$

`eidx` Helper returning the storage index window of the error block

`eltype` `Julia` function returning the element type of an array

`errtype` Helper returning enclosure/error type of a series space

`Estart` Cumulative offsets for error blocks of vector components

`ET` Error/enclosure type used in a series space (often an interval type)

`Even, Odd, Complete` Parity flags selecting cosine basis, sine basis, or full Fourier basis

`E` Error/tail enclosure vector storing validated bounds for the truncated remainder (non-truncated spaces only)

`FFTAlgorithm` DFFT-based strategy for nonlinear operations

`first_cidx` Helper returning the first storage index of the coefficient block

`first_eidx` Helper returning the first storage index of the error block

`fourier_mul` Validated multiplication kernel handling C–C, C–E and E–E interactions

`FourierCompleteSpace` Type alias for Fourier spaces with complete parity

`FourierEvenSpace` Type alias for Fourier spaces with even parity

`FourierOddSpace` Type alias for Fourier spaces with odd parity

`FourierSpace{CT,ET,T,P}` Concrete series space storing Fourier parameters, truncation, and parity

`freq_coeff` Helper mapping coefficient storage indices to Fourier frequencies

`freq_error` Helper mapping error storage indices to Fourier frequencies

`F` Stored callable implementing the nonlinear map $F : \mathcal{F}_\rho^d \to \mathcal{F}_\rho^d$

`g_tol` Tolerance for the gradient norm used as stopping criterion

`Get_component_length` Routine computing the number of scalar blocks in a vector space

`Get_Space_Antiderivative` Routine constructing the target Fourier space for an $n$-th antiderivative

`get_space_product` Helper constructing the product Fourier space and checking compatibility of inputs

`Get_space_sum` Constructor producing the vector sum space from two compatible vector series spaces

`get_subspace_indices` Routine returning coefficient index windows for the truncated subspace

`Gradient_descent` Auxiliary routine integrating $\eta_f$ until a stopping criterion is met

`idx_shift` Generic parity-dependent shift mapping storage indices to frequencies (coefficients or errors)

`infinite` Flag: if `true`, basis iteration includes tail slots (not allowed for truncated spaces)

`inverse_power` Algorithm/parameter object for computing series reciprocals

`istruncated` Helper returning whether a series space is truncated

`Iterate` `Julia` iteration method producing successive basis elements and the next iterator state

`Julia` High-performance programming language [70] used for the CAPS implementation

`last_cidx` Helper returning the last storage index of the coefficient block

`last_eidx` Helper returning the last storage index of the error block

`LinearAlgebra.norm` `Julia` overload computing the validated norm of a `Series`

`M_dim` Integer parameter specifying the truncated subspace size used to build $M$

`mul` In-place application of a compact operator using matrix–vector multiplication on coefficient blocks

`n_bodies_caps` Code repository implementing the CAP validation pipeline specialised to the $n$-body problem

`n_coeffs` Total number of stored Fourier coefficients in a vector series

`n_errors_basis` Number of tail slots in the basis (depends on parity: even/odd: 1, complete: 2)

`n_errors` Total number of tail/error slots in a vector series

`NewtonAlgorithm` Newton–Raphson strategy for nonlinear operations

`normalised` Basis flag rescaling each basis element to unit norm

`Norm` Routine computing the series norm by summing weighted coefficient bounds and tail contributions

`Operator{S,F}` Wrapper pairing a space $S$ with a function $F$ implementing an operator

`Power` Routine computing integer powers via exponentiation by squaring, calling validated multiplication

`Proof` Problem specification bundling space, truncation, nonlinear map $F$, derivative action $DF$, and proof parameters

`prove_contraction` Routine computing $\varepsilon$, $k$ and checking $\varepsilon + kr < r$ for the corrected map $C$

`quadgk` Gauss–Kronrod adaptive quadrature routine used for numerical integration

`R_an` Auxiliary analytic/tail parameter used for controlling truncation effects in the proof routine

`rational_power` Algorithm/parameter object for computing rational powers of series

`Reciprocal_Taylor` Routine computing the reciprocal via geometric expansion

`repository_caps` Code repository implementing the computer-assisted proof (CAP) framework used in this thesis

`r` Validation radius defining the ball $B(u_0, r)$

`Series{T,V}` Data container coupling a series space with coefficient vector $C$ and error vector $E$

`SeriesSpace{CT,ET,T}` Abstract type encoding coefficient type, error type, and truncation flag for a series space

`space` Ambient series space

`Sqrt_FFT` DFFT-based square-root routine with rigorous validation

`Sqrt_Newton` Routine computing the square root via Newton iteration

`SqrtAlgorithm` Enum selecting the method used to compute square roots of series

`Sqrt` Dispatcher selecting the square-root algorithm based on `rational_power`

`Stop_basis_iterator` Stopping rule determining when basis iteration ends, depending on parity and available tail slots

`subspace_dim` Truncation dimension defining the finite coefficient subspace for a compact operator

`success` Boolean flag indicating whether the contraction inequality $\varepsilon + kr < r$ is satisfied

`supabs` Supremum of the absolute value of a scalar or interval, used to bound interval coefficients in norms

`TaylorAlgorithm` Binomial/Taylor expansion strategy for nonlinear operations

`TaylorPower` Routine computing rational powers via truncated binomial expansion

`to_min` Boolean flag controlling `Gradient_descent` stopping: minimum-seeking vs. increase-of-gradient rule

`To_Points_FFT` Routine mapping Fourier coefficients to grid values

`To_Series_FFT` Routine mapping grid values back to Fourier coefficients

`trunc_space` Finite-dimensional companion space $\mathcal{P}_\ell \mathcal{F}_\rho^d$ used for certified matrix computations

`truncate_space` Utility producing the truncated variant of a Fourier space

`T` Boolean truncation flag indicating whether the space is truncated

`VectorSeriesSpace` Data structure encoding a vector-valued series space

A posteriori validation Certification step ensuring existence and uniqueness of a true solution near a numerical approximation

Action value Value of the variational action functional at the computed orbit

Analytic continuation Extension of a real function to a complex domain where it remains analytic

Analytic periodic function Function admitting a convergent Fourier series representation

Banach algebra Banach space closed under multiplication with submultiplicative norm

Banach fixed-point theorem Contraction principle guaranteeing existence and uniqueness of solutions

Base reservoir Low-mode error accumulator used when a contribution cannot be assigned to a positive-frequency slot

Basin hopping Restart mechanism that perturbs the search to escape a basin of attraction

Basin of attraction Set of initial points leading a local method to a given local minimiser

Basis element Canonical series supported on exactly one coefficient mode (or one tail slot), all others zero

Brake orbit Loop $x \in \Lambda$ invariant under a time-reflection: $x(\tau(g)t) = x(t)$ for some $g \in G$

Brake subgroup Subgroup $H = \{1, S\} \leq O(2)$ generated by a reflection

C–C Coefficient–coefficient interaction in series multiplication

C–E Coefficient–error interaction in series multiplication

CAPs Computer-assisted proofs: rigorous mathematical proofs relying on validated numerical computation

Coercive Property ensuring boundedness of sublevel sets and boundedness of Palais–Smale sequences

Complete series Fourier series carrying both even and odd components

Component length Number of independent coefficient blocks carried by a scalar space

Critical point Point where the first variation of a functional vanishes

DFFT Discrete Fast Fourier Transform based method for nonlinear operations

Discrete Morse index Finite-dimensional approximation of the Morse index obtained from a discretised functional

Diversification Strategy promoting exploration of less-visited regions of the feasible set

E–E Error–error interaction in series multiplication

Enclosure property Guarantee that the true result lies within the computed interval

Error slot Tail enclosure accounting for unresolved Fourier modes

Even function $2\pi$-periodic function expanded only in cosines

| | |
|---|---|
| Even space | Fourier space storing cosine modes (includes the constant mode $k = 0$) |
| Feasibility restoration phase | Procedure aiming to reduce infeasibility, typically by (approximately) minimising $h(x)$ |
| Fitness | Quality measure of a candidate solution (often based on the objective value) |
| Fixed-point equation | Equation of the form $q = \partial^{-2} f(q)$ whose solutions correspond to periodic orbits |
| Formal verification | Mechanical verification of proofs using proof assistants such as Coq, Isabelle, or Lean |
| Fourier basis | Trigonometric basis used to represent periodic functions |
| Fourier subspace | Parity-restricted spectral space chosen per component (cosine-only, sine-only, or both) |
| Function representation | Encoding of functions via Fourier coefficients or series |
| Functional space | Banach or Hilbert space used to represent periodic functions |
| Funnel structure | Landscape structure where progressively better minima are organised in a funnel-like hierarchy |
| Generation | Iteration step of a genetic algorithm |
| Gradient norm | Norm of the action gradient at the numerical solution (criticality indicator) |
| Group generator(s) | Generators specifying the symmetry group used to constrain/describe the orbit |
| Heuristic method | Problem-solving method aimed at finding a feasible (not necessarily optimal) solution |
| Horner-like accumulation | Incremental construction of powers of $X$ to reduce complexity |
| Hybrid algorithm | Algorithm combining complementary optimisation strategies (e.g., tabu search and simulated annealing) |
| IEEE 1788–2015 | International standard for interval arithmetic |
| Index form | Second variation of a functional evaluated at a critical point |
| Intensification | Strategy focusing the search on promising regions of the feasible set |
| Interval arithmetic | Algebraic framework extending arithmetic operations to intervals |
| Interval enclosure | Set guaranteed to contain the true mathematical value |
| Interval evaluation | Computation of function ranges over intervals |
| Interval extension | Set of all possible results of an operation over interval operands |
| Isolated minimum | Local minimiser not belonging to a continuous family (important for Mountain Pass construction) |
| KKT | Karush–Kuhn–Tucker optimality conditions |
| LICQ | Linear Independence Constraint Qualification |

Local improvement procedure Iterative neighbourhood-based method for improving a candidate solution

Machine-representable number Number exactly representable in floating-point arithmetic

Memoised routines Cached computations (e.g. distances) used to accelerate repeated evaluations of $f$ and $DF$ in the proof pipeline

Metaheuristic method High-level strategy for designing problem-specific heuristics

Monotone function Function preserving order over an interval

Morse index Number of negative directions of the second variation at a critical point

Multi-funnel Landscape with multiple funnels (distinct basins/clusters separated by higher barriers)

Multi-interval Union of disjoint intervals arising e.g. from division by intervals containing zero

Mutation Random modification introducing diversity in offspring solutions

N. of coefficients Total number of stored coefficients across all components in the chosen product/vector space

Newton iteration (truncated) Finite-dimensional Newton method on truncated Fourier coefficients used to compute $u_0$

Newton–Raphson method Iterative root-finding scheme with quadratic convergence

NoG Number of generations in the MP-AIDEA&Lattice recursion

NoL Number of Lattice layers (depth of domain decomposition)

Non-isolated minimum Minimum belonging to a continuum of minimisers (e.g. by rotational invariance in type-S groups)

Non-truncated space Series space where tail bounds (error/enclosure vector) are carried explicitly

Norm Quantity measuring the size of a function in the chosen functional space

Normalisation Rescaling of a basis element to unit norm in the weighted $\ell^1$ structure

Odd function $2\pi$-periodic function expanded only in sines

Odd space Fourier space storing sine modes (starts at $k = 1$)

Offspring New candidate solutions generated from parents in a genetic algorithm

Oversampling Using $M > 2N$ grid points to suppress aliasing errors

Palais–Smale sequence Sequence with bounded functional values and vanishing gradient

Parents Selected individuals used to generate offspring in a genetic algorithm

Parity Even/odd/complete classification of a Fourier series

Piecewise monotone function Function monotone only on subintervals

Population Set of candidate solutions evolved in a genetic algorithm

Positively invariant Property: $\eta_f(t, x_0) \in X_0$ for all $t \geq 0$ and $x_0 \in X_0$

Quadratic convergence Property that the error squares at each Newton step

Rigorous numerics Numerical computation with mathematically certified error bounds

Rotating circle property Property ensuring local minimisers are collision-free under suitable hypotheses

Rounding error Numerical error arising from finite-precision arithmetic

Runge–Kutta solver Numerical ODE integrator used to compute an approximate solution $u_0$

Standard set Interval with machine-representable endpoints enclosing a real quantity

Tabu list Short-term memory structure preventing immediate repetition of recent moves

Tail slot Distinguished basis direction associated with an error reservoir (validated remainder beyond stored modes)

Temperature schedule Rule defining how the temperature $T$ is decreased during simulated annealing

Test problem Validation example specifying $F$, $DF$, a numerical guess $u_0$, and a contraction proof via `prove_contraction`

tol Tolerance

Truncated space Series space where only a finite number of modes is represented explicitly

Truncation error Error introduced by finite Fourier series approximations

Type S Property: existence of a one-parameter rotation group $R_\theta$ such that $R_\theta x \in Y^G$ for all $x \in Y^G$

Uncertainty propagation Rigorous tracking of all numerical uncertainties through computations

Validated numerics Numerical methods that rigorously control all sources of error, including rounding and truncation

Vector Banach algebra Algebra structure inherited from scalar spaces under componentwise multiplication

Vector basis iterator Iterator traversing coefficient and tail modes across all components

Vector canonical basis Family $\{e_k^{(i)}\}$ spanning $\mathcal{F}_\rho^d$

Vector series space Ordered tuple of $d$ compatible scalar series spaces

Vector space compatibility Condition that two vector series spaces have the same dimension and componentwise compatible scalar spaces

Winding number Topological winding indicator of the orbit (when defined/available)

Zero-average condition Constraint ensuring uniqueness of the antiderivative operator $\partial^{-2}$

# Bibliography


[1] H. Poincaré, *Les méthodes nouvelles de la mécanique céleste.* Paris: Gauthier-Villars, 1892.

[2] M. C. Gutzwiller, *Chaos in Classical and Quantum Mechanics*, ser. Interdisciplinary Applied Mathematics (IAM), Volume 1. Springer New York, NY, 1990.

[3] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*, 2nd ed. CRC Press, 2003.

[4] J. Moser, *Stable and Random Motions in Dynamical Systems: With Special Emphasis on Celestial Mechanics (AM-77)*, rev - revised ed. Princeton University Press, 1973.

[5] S. Bolotin, "Symbolic dynamics of almost collision orbits and skew products of symplectic maps," *Nonlinearity*, vol. 19, no. 9, pp. 2041–2063, 2006.

[6] J. Llibre and C. Simó, "Oscillatory solutions in the planar restricted three-body problem," *Math. Ann.*, vol. 248, no. 2, pp. 153–184, 1980.

[7] M. Guardia, P. Martín, and T. M. Seara, "Oscillatory motions for the restricted planar circular three body problem," *Invent. Math.*, vol. 203, no. 2, pp. 417–492, 2016.

[8] T. C. N. Boekholt, S. F. Portegies Zwart, and M. Valtonen, "Gargantuan chaotic gravitational three-body systems and their irreversibility to the planck length," *Monthly Notices of the Royal Astronomical Society*, vol. 493, no. 3, pp. 3932–3937, 2020.

[9] A. Celletti, *Stability and Chaos in Celestial Mechanics.* Berlin; Heidelberg; New York: Springer, 2009.

[10] N. W. C. Leigh, N. C. Stone, A. M. Geller, M. M. Shara, H. Muddu, D. Solano-Oropeza, and Y. Thomas, "The chaotic four-body problem in newtonian gravity– i. identical point-particles," *Monthly Notices of the Royal Astronomical Society*, vol. 463, no. 3, pp. 3311–3325, 2016.

[11] V. Manwadkar, A. A. Trani, and N. W. C. Leigh, "Chaos and lévy flights in the three-body problem," *Monthly Notices of the Royal Astronomical Society*, vol. 497, no. 3, pp. 3694–3712, 2020.

[12] T. Kapela and C. Simó, "Computer assisted proofs for nonsymmetric planar choreographies and for stability of the eight," *Nonlinearity*, vol. 20, no. 5, p. 1241, 2007.

[13] T. Kapela and P. Zgliczyński, "The existence of simple choreographies for the N-body problem—a computer-assisted proof," *Nonlinearity*, vol. 16, no. 6, pp. 1899–1918, 2003.

[14] D. Wilczak and P. Zgliczyński, "Heteroclinic connections between periodic orbits in planar restricted circular three body problem. part ii," *Communications in Mathematical Physics*, vol. 259, no. 3, pp. 561–576, 2005.

[15] M. Fenucci and G. F. Gronchi, "On the stability of periodic n-body motions with the symmetry of platonic polyhedra," *Nonlinearity*, vol. 31, no. 11, p. 4935, oct 2018.

[16] T. Kapela, M. Mrozek, D. Wilczak, and P. Zgliczyński, "CAPD::DynSys: A flexible c++ toolbox for rigorous numerical analysis of dynamical systems," *Communications in Nonlinear Science and Numerical Simulation*, vol. 101, p. 105578, October 2021.

[17] R. Montgomery, *Four Open Questions for the N-Body Problem.* Cambridge: Cambridge University Press, 2024.

[18] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, 2nd ed., ser. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer, 2006, vol. 31.

[19] J. M. Sanz-Serna and M. P. Calvo, *Numerical Hamiltonian Problems.* London: Chapman & Hall, 1994.

[20] H. Rein, D. Tamayo, and G. Brown, "High-order symplectic integrators for planetary dynamics and their implementation in rebound," *Monthly Notices of the Royal Astronomical Society*, vol. 489, no. 4, pp. 4632–4640, 2019.

[21] J. Wisdom and M. Holman, "Symplectic maps for the $n$-body problem," *The Astronomical Journal*, vol. 102, no. 4, pp. 1528–1538, 1991.

[22] J. Laskar, "Frequency analysis for multi-dimensional systems. global dynamics and diffusion," *Physica D: Nonlinear Phenomena*, vol. 67, no. 1, pp. 257–281, 1993.

[23] T. Levi-Civita, "Sur la résolution qualitative du problème restreint des trois corps," *Acta Mathematica*, vol. 30, pp. 305–327, 1906.

[24] P. Kustaanheimo and E. Stiefel, "Perturbation theory of the kepler motion based on spinor regularization," *Journal für die reine und angewandte Mathematik*, vol. 218, pp. 204–219, 1965.

[25] P. Liu and S. Bhatt, "Experiences with parallel n-body simulation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 12, pp. 1306–1323, 2000.

[26] H. Menon, L. Wesolowski, G. Zheng, P. Jetley, L. Kale, T. Quinn, and F. Governato, "Adaptive techniques for clustered n-body cosmological simulations," *Computational Astrophysics and Cosmology*, vol. 2, no. 1, p. 1, 2015.

[27] V. I. Arnol'd, "Small denominators and problems of stability of motion in classical and celestial mechanics," *Russian Mathematical Surveys*, vol. 18, no. 6, p. 85, 1963.

[28] G. Pinzari, "Perturbation theory and canonical coordinates in celestial mechanics," *Reviews in Mathematical Physics*, vol. 36, no. 05, p. 2430006, 2024.

[29] J. Féjoz, *13. Introduction to KAM theory with a view to celestial mechanics.* Berlin, Boston: De Gruyter, 2017, pp. 387–433.

[30] N. N. Nekhoroshev, "An exponential estimate of the time of stability of nearly-integrable hamiltonian systems," *Russian Mathematical Surveys*, vol. 32, no. 6, p. 1, 1977.

[31] J. Pöschel, "Lecture notes on nekhoroshev estimates," in *Hamiltonian Systems with Three or More Degrees of Freedom*, ser. NATO Science Series C, C. Simó, Ed. Dordrecht: Springer, 2001, vol. 533, pp. 271–292.

[32] A. Ambrosetti and P. Rabinowitz, "Dual variational methods in critical point theory and applications." *Journal of Functional Analysis*, vol. 14, no. 4, pp. 349–381, 1973.

[33] U. Bessi and V. C. Zelati, "Symmetries and noncollision closed orbits for planar n-body-type problems," *Nonlinear Analysis*, vol. 16, no. 6, pp. 587–598, 1991.

[34] A. Chenciner and R. Montgomery, "A remarkable periodic solution of the three-body problem in the case of equal masses." *Ann. Math. (2)*, vol. 152, pp. 881–901, 2000.

[35] D. Ferrario and S. Terracini, "On the existence of collisionless equivariant minimizers for the classical n-body problem." *Inventiones mathematicae*, vol. 155, pp. 305–362, 2004.

[36] K.-C. Chen, "Action-minimizing orbits in the parallelogram four-body problem with equal masses," *Archive for Rational Mechanics and Analysis*, vol. 158, no. 4, pp. 293–318, 2001.

[37] M. Ramos and S. Terracini, "Noncollision periodic solutions to some singular dynamical systems with very weak forces," *Journal of Differential Equations*, vol. 118, no. 1, pp. 121–152, 1995.

[38] A. Chenciner and A. Venturelli, "Minima de l'intégrale d'action du probléme newtonien de 4 corps de masses égales dans $\mathbb{R}^3$: orbites "hip-hop"." *Celest. Mech. Dyn. Astron.*, vol. 77, pp. 139–152, 2000.

[39] V. Barutello and S. Terracini, "Action minimizing orbits in the n-body problem with simple choreography constraint," *Nonlinearity*, vol. 17, no. 6, pp. 2015–2039, 2004.

[40] G. Fusco, G. Gronchi, and P. Negrini, "Platonic polyhedra, topological constraints and periodic solutions of the classical n-bodyproblem." *Inventiones mathematicae*, vol. 185, pp. 283–332, 2011.

[41] J. Montaldi and K. Steckles, "Classification of symmetry groups for planar n-body choreographies," *Forum of Mathematics, Sigma*, vol. 1, pp. e5, 55, 2013.

[42] C. Simó, "New families of solutions in n-body problems," in *European Congress of Mathematics*, C. Casacuberta, R. M. Miró-Roig, J. Verdera, and S. Xambó-Descamps, Eds. Basel: Birkhäuser Basel, 2001, pp. 101–115.

[43] V. Barutello, G. M. Canneori, R. Ciccarelli, S. Terracini, M. G. Bergomi, P. Vertechi, and D. L. Ferrario, "Equivariant optimisation for the gravitational n-body problem: A computational factory of symmetric orbits," *Communications in Nonlinear Science and Numerical Simulation*, vol. 152, p. 109180, 2026.

[44] D. L. Ferrario, "Symmetries and periodic orbits for the n-body problem: about the computational approach," 2024.

[45] ——, "Symmetry groups and non-planar collisionless action-minimizing solutions of the three-body problem in three-dimensional space," *Archive for Rational Mechanics and Analysis*, vol. 179, pp. 389–412, 2006.

[46] ——, "Transitive decomposition of symmetry groups for the n-body problem," *Advances in Mathematics*, vol. 213, no. 2, pp. 763–784, 2007.

[47] V. Barutello, D. L. Ferrario, and S. Terracini, "Symmetry groups of the planar 3-body problem and action–minimizing trajectories," *Archive for Rational Mechanics and Analysis*, vol. 190, p. 189–226, 2018.

[48] Z. Xia and T. Zhou, "Applying the symmetry groups to study the n-body problem," *Journal of Differential Equations*, vol. 310, pp. 302–326, 2022.

[49] J. Burgos-García, J.-P. Lessard, and J. D. M. James, "Spatial periodic orbits in the equilateral circular restricted four-body problem: computer-assisted proofs of existence," *Celestial Mechanics and Dynamical Astronomy*, vol. 131, no. 2, p. 2, 2019.

[50] D. Wilczak and P. Zgliczyński, "Heteroclinic connections between periodic orbits in planar restricted circular three-body problem – a computer assisted proof," *Communications in Mathematical Physics*, vol. 234, no. 1, pp. 37–75, 2003.

[51] T. Kapela and C. Simó, "Computer assisted proofs for nonsymmetric planar choreographies and for stability of the eight," *Nonlinearity*, vol. 20, no. 5, pp. 1241–1255, 2007.

[52] M. Guardia, P. Martín, and T. M. Seara, "Oscillatory motions for the restricted planar circular three body problem," *Inventiones mathematicae*, vol. 203, no. 2, pp. 417–492, 2016.

[53] J. Lamas, M. Guardia, and T. M. Seara, "Oscillatory motions, parabolic orbits and collision orbits in the planar circular restricted three-body problem," *Communications in Mathematical Physics*, vol. 406, p. 106, 2025.

[54] G. Arioli, V. Barutello, and S. Terracini, "A new branch of mountain pass solutions for the choreographical 3-body problem," *Communications in Mathematical Physics*, vol. 268, no. 2, pp. 439–463, December 2006.

[55] G. Arioli, "Periodic orbits, symbolic dynamics and topological entropy for the restricted 3-body problem," *Communications in Mathematical Physics*, vol. 231, no. 1, pp. 1–24, August 2002.

[56] G. Arioli and J. D. Mireles James, "Branches and bifurcations of ejection–collision orbits in the planar circular restricted three body problem," *Nonlinearity*, vol. 38, no. 4, p. 045010, March 2025.

[57] G. Arioli, H. Koch, and S. Terracini, "Two novel methods and multi-mode periodic solutions for the fermi-pasta-ulam model," *Communications in Mathematical Physics*, vol. 255, no. 1, pp. 1–19, April 2005.

[58] M. Vasile, E. Minisci, and M. Locatelli, "An inflationary differential evolution algorithm for space trajectory optimization." *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, p. 267–281, 2011.

[59] V. Barutello and S. Terracini, "A bisection algorithm for the numerical mountain pass," *Nonlinear Differential Equations and Applications NoDEA*, vol. 14, pp. 527–539, 2007.

[60] V. Barutello and G. Canneori, "Lecture notes in selected topics in celestial mechanics." *Politecnico of Turin*, March 2023.

[61] R. Palais, "The principle of symmetric criticality." *Comm. Math. Phys.*, vol. 69(1), pp. 19–30, 1979.

[62] I. Cravero and M. Introna, "A study on optimisation methods in celestial mechanics." *Rend. Sem. Mat. Univ. Pol. Torino*, 2024.

[63] B. C. Rust and J. R. Lewis, "Finite difference discretizations by differential quadrature techniques," *Communications in Numerical Methods in Engineering*, vol. 15, no. 11, pp. 823–833, 1999.

[64] Y. Huang and A. Oberman, "Numerical methods for the fractional laplacian: A finite difference–quadrature approach," *SIAM Journal on Numerical Analysis*, vol. 52, no. 6, pp. 3056–3084, 2014.

[65] J. E. Hicken and D. W. Zingg, "Summation-by-parts operators and high-order quadrature," *SIAM Journal on Scientific Computing*, vol. 33, no. 2, pp. 893–922, 2011.

[66] K. K. G. Myrdal, "Some theoretical and numerical aspects of the n-body problem," Bachelor's thesis, Lund University Faculty of Science, Centre for Mathematical Sciences, Mathematics, 2013.

[67] C. Simó, "Periodic orbits of the planar n -body problem with equal masses and all bodies on the samepath." *IoP Publishing, Bristol*, p. 265–284, 2001.

[68] ——, "Dynamical properties of the figure eight solution of the three-body problem," in *Celestial Mechanics: Dedicated to Donald Saari for his 60th Birthday*, ser. Contemporary Mathematics, A. Chenciner, R. Cushman, C. Robinson, and Z. J. Xia, Eds. American Mathematical Society, 2002, vol. 292, pp. 209–228.

[69] M. D. Carlo, M. Vasile, and E. Minisci, "Adaptive multi-population inflationary differential evolution," *Soft Computing*, vol. 24, p. 3861–3891, 2020.

[70] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[71] E. Geminiani, G. Marra, and I. Moustaki, "Single- and multiple-group penalized factor analysis: A trust-region algorithm approach with integrated automatic multiple tuning parameter selection," *Psychometrika*, vol. 86, no. 1, pp. 65–95, 2021.

[72] J. C. Gross, P. Seshadri, and G. Parks, "Optimisation with intrinsic dimension reduction: A ridge informed trust-region method," in *AIAA SciTech Forum, Session: Emerging Methods, Algorithms, and Software Development in MDAO*. American Institute of Aeronautics and Astronautics, 2020, paper AIAA 2020-0157.

[73] J. G. Kuba, R. Chen, M. Wen, Y. Wen, F. Sun, J. Wang, and Y. Yang, "Trust region policy optimisation in multi-agent reinforcement learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022, conference paper.

[74] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, Long Beach, CA, USA, 2017.

[75] B. Liu and W. Zhao, "New exact penalty functions for nonlinear constrained optimization problems," *Abstract and Applied Analysis*, 2014, article ID 738926.

[76] Y. Qian, S. Pan, and L. Xiao, "Error bound and exact penalty method for optimisation problems with nonnegative orthogonal constraint," *IMA Journal of Numerical Analysis*, vol. 00, pp. 1–37, 2023.

[77] G. Z. Oztas and S. Erdem, "A penalty-based algorithm proposal for engineering optimization problems." *Neural Computing and Applications*, vol. 35, p. 763–765, 2023.

[78] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 7th ed. New York: McGraw-Hill, 2004.

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

[80] M. B. Oguntola and R. J. Lorentzen, "Ensemble-based constrained optimisation using an exterior penalty method," *Journal of Petroleum Science and Engineering*, vol. 207, p. 109165, Dec. 2021.

[81] Z. Meng, Q. Hu, C. Dang, and X. Yang, *An Objective Penalty Function Method for Nonlinear Programming*. McGraw-Hill Higher Education, 2005.

[82] S. Gupta, H. Abderazek, B. S. Yıldız, A. R. Yildiz, S. Mirjalili, and S. M. Sait, "Comparison of metaheuristic optimisation algorithms for solving constrained mechanical design optimisation problems," *Expert Systems with Applications*, vol. 183, 2021.

[83] J. A. Parejo, A. Ruiz-Corte´s, and S. Lozano, "Metaheuristic optimization frameworks: a survey and benchmarking." *Springer-Verlag*, 2011.

[84] K. Sörensen, "Metaheuristics—the metaphor exposed." *Trans. in Op. Res. Intl.*, vol. 22, p. 3–18, 2015.

[85] T. Akan, A. M. Anter, A. Ş. Etaner-Uyar, and D. Oliva, *Engineering Applications of Modern Metaheuristics*. Springer Nature Switzerland AG, 2023.

[86] M. Vasile and M. Locatelli, "A hybrid multiagent approach for global trajectory optimization," *Journal of Global Optimization*, vol. 44, no. 4, pp. 461–479, 2009.

[87] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, ser. Natural Computing Series. Berlin/Heidelberg, Germany: Springer-Verlag, 2005.

[88] R. H. Leary, "Global optimization on funneling landscapes," *Journal of Global Optimization*, vol. 18, no. 4, pp. 367–383, December 2000.

[89] D. J. Wales and J. P. K. Doye, "Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms," *Journal of Physical Chemistry A*, vol. 101, pp. 5111–5116, July–August 1997.

[90] R. H. Leary, "Global optimization on funneling landscapes." *Journal of Global Optimization.*, vol. 18, pp. 367–383, 2000.

[91] M. D. Carlo, M. Vasile, E. Minisci, and M. Introna, "Smart o2c - mpaidea," https://github.com/strath-ace-labs/smart-o2c/tree/development/Julia/Optimisation/MPAIDEA, 2024.

[92] R. Ciccarelli, M. Introna, S. Terracini, and M. Vasile, "New solutions for the symmetrical $n$-body problem through variational approach and optimisation techniques," *Aerotecnica Missili & Spazio*, 2025.

[93] M. Morse, *The Calculus of Variations in the Large*, ser. American Mathematical Society Colloquium Publications. Providence, RI: American Mathematical Society, 1934, vol. 18.

[94] S. Smale, "On the morse index theorem," *Journal of Mathematics and Mechanics*, vol. 14, no. 5, pp. 1049–1055, 1965.

[95] A. Ambrosetti and V. Coti Zelati, *Periodic Solutions of Singular Lagrangian Systems*, ser. Progress in Nonlinear Differential Equations and Their Applications. Basel: Birkhäuser, 1993, vol. 10.

[96] P. H. Rabinowitz, *Periodic Solutions of Hamiltonian Systems*, ser. Communications on Pure and Applied Mathematics. American Mathematical Society, 1978, vol. 31.

[97] K.-C. Chang, *Infinite Dimensional Morse Theory and Multiple Solution Problems*, ser. Progress in Nonlinear Differential Equations and Their Applications. Boston, MA: Birkhäuser, 1993, vol. 6.

[98] V. Barutello and S. Secchi, "Morse index properties of colliding solutions to the n-body problem," *Annales de l'Institut Henri Poincaré C, Analyse non linéaire*, vol. 25, no. 3, pp. 539–565, 2008.

[99] V. Barutello, R. D. Jadanza, and A. Portaluri, "Morse index and linear stability of the lagrangian circular orbit in a three-body-type problem via index theory," *Archive for Rational Mechanics and Analysis*, vol. 219, no. 1, pp. 387–444, 2016.

[100] X. Hu and S. Sun, "Morse index and stability of elliptic lagrangian solutions in the planar three-body problem," *Advances in Mathematics*, vol. 223, no. 1, pp. 98–119, 2010.

[101] V. L. Barutello, X. Hu, A. Portaluri, and S. Terracini, "An index theory for asymptotic motions under singular potentials," *Advances in Mathematics*, vol. 370, p. 107230, 2020.

[102] H. Fukuda, T. Fujiwara, and H. Ozaki, "Morse index and bifurcation for figure-eight choreographies of the equal mass three-body problem," *Journal of Physics A: Mathematical and Theoretical*, vol. 52, no. 18, p. 185201, 2019.

[103] R. Forman, "Morse theory for cell complexes," *Advances in Mathematics*, vol. 134, no. 1, pp. 90–145, 1998.

[104] M. Introna, S. Terracini, M. Vasile, and R. Ciccarelli, "Exploring new orbits for the n-body problem." *75th International Astronautical Conference, Milan, Italy*, 2024.

[105] D. Berti, G. M. Canneori, R. Ciccarelli, I. De Blasi, M. Introna, and D. Polimeni, "Symmetric solutions of the $n$-body problem: a numerical study of floquet multipliers and morse indices," 2025, under review.

[106] M. Vasile, E. Minisci, and M. Locatelli, "Analysis of some global optimization algorithms for space trajectory design," *Journal of Spacecraft and Rockets*, vol. 49, no. 3, pp. 450–462, 2012.

[107] C. R. Reeves and T. Yamada, "Genetic algorithms, path re-linking and the flowshop sequencing problem," *Evolutionary Computation*, vol. 6, no. 1, pp. 45–60, 1998.

[108] A. Moreno, "Periodic orbits and invariants in the restricted three-body problem," *EMS Magazine*, 2024.

[109] R. Montgomery, "A new solution to the three-body problem," *Notices of the American Mathematical Society*, vol. 48, no. 4, pp. 471–481, 2001.

[110] Y. Jiang, H. Baoyin, and J. Li, "Topological classifications and bifurcations of periodic orbits in the potential field of highly irregular-shaped celestial bodies," *Astrophysics and Space Science*, vol. 351, no. 1, pp. 203–219, 2014.

[111] P. Pucci and J. Serrin, "The structure of the critical set in the mountain pass theorem." *Transactions of the American Mathematical Society*, vol. 299, no. 1, pp. 115–132, 1987.

[112] M. Shibayama, "Minimax approach to the $n$-body problem." *Nonlinear Dynamics in Partial Differential Equations.*, vol. 64, pp. 221–228, 2015.

[113] K. Appel and W. Haken, "The solution of the four-color-map problem," *Scientific American*, vol. 237, no. 4, pp. 108–121, 1977.

[114] T. C. Hales, "A proof of the kepler conjecture," *Annals of Mathematics*, vol. 162, no. 3, pp. 1065–1185, 2005.

[115] ——, "Introduction to the flyspeck project," in *Mathematics, Algorithms, Proofs*, ser. Dagstuhl Seminar Proceedings, no. 05021. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[116] W. Tucker, "A rigorous ode solver and smale's 14th problem," *Foundations of Computational Mathematics*, vol. 2, no. 1, pp. 53–117, 2002.

[117] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2009.

[118] W. Research, "Mathematica 8.0," Champaign, Illinios, 2010.

[119] Maplesoft, a division of Waterloo Maple Inc.., "Maple," Waterloo, Ontario.

[120] S. Rump, "INTLAB - INTerval LABoratory," in *Developments in Reliable Computing*, T. Csendes, Ed. Dordrecht: Kluwer Academic Publishers, 1999, pp. 77–104.

[121] F. Johansson, "Arb: efficient arbitrary-precision midpoint-radius interval arithmetic," *IEEE Transactions on Computers*, vol. 66, pp. 1281–1292, 2017.

[122] The Coq Development Team, *The Coq Proof Assistant Reference Manual – Version V8.0*, INRIA, April 2004.

[123] W. Naraschewski and T. Nipkow, "Isabelle/hol," 2020. [Online]. Available: http://www.cl.cam.ac.uk/research/hvg/Isabelle/

[124] L. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer, "The lean theorem prover (system description)," in *International Conference on Automated Deduction (CADE)*, ser. Lecture Notes in Computer Science, vol. 9195. Springer, August 2015, pp. 378–388.

[125] R. Ciccarelli and M. Introna, "Computer assisted proofs," https://github.com/CelestialCAPs/ComputerAssisted, 2025, open source 2026.

[126] D. P. Sanders and L. Benet, "Intervalarithmetic.jl," 2014. [Online]. Available: https://github.com/JuliaIntervals/IntervalArithmetic.jl

[127] L. Feranti, M. Forets, and D. P. Sanders, "Intervallinearalgebra.jl: linear algebra done rigorously," 9 2021. [Online]. Available: https://github.com/juliaintervals/IntervalLinearAlgebra.jl

[128] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. New York: McGraw-Hill, 1976.

[129] S. Banach, "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales," *Fundamenta Mathematicae*, vol. 3, pp. 1–97, 1922.

[130] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, special issue on "Program Generation, Optimization, and Platform Adaptation".

[131] R. Ciccarelli and M. Introna, "SymOrbProofs," https://github.com/CelestialCAPs/SymOrbProofs, 2025, open source 2026.

[132] R. Ciccarelli, M. Introna, S. Terracini *et al.*, "New periodic and equivariant solutions to the $n$-body problem," 2025, to appear.

[133] H. Fukuda, T. Fujiwara, and H. Ozaki, "Morse index for figure-eight choreographies of the planar equal mass three-body problem," *Journal of Physics A: Mathematical and Theoretical*, vol. 51, no. 14, p. 145201, 2018.