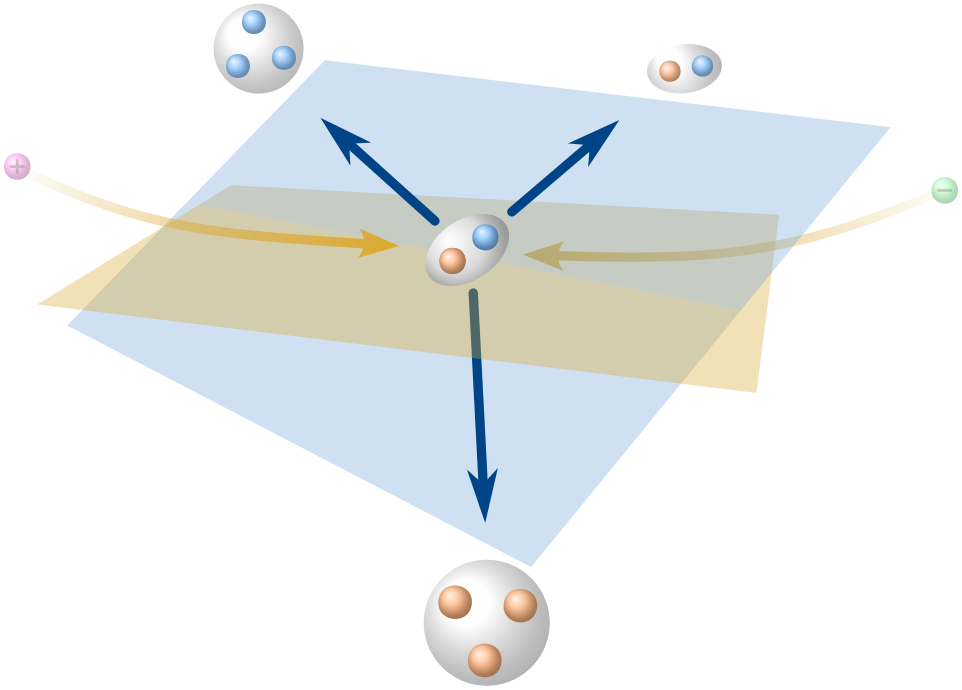
RUHR-UNIVERSITÄT BOCHUM

RUB

Aligning Baryons

High-performance computations with symbolic, spin-aligned amplitude models for hadron spectroscopy



Dissertation
zur
Erlangung des Grades eines
Doktors der Naturwissenschaften
an der Fakultät für Physik und Astronomie
der Ruhr-Universität Bochum

von

Remco Emiel de Boer
aus Wageningen (Niederlande)

Bochum, September 2025

1. Gutachter	Prof. Dr. Miriam Fritsch
2. Gutachter	Prof. Dr. Mikhail Mikhasenko
Datum der Disputation	1. Dezember 2025

Publikationsfassung / Published Version (Paperback),
24 September 2025

Abstract

Amplitude analysis is a key technique in hadron physics, linking experimental observables to the underlying dynamics of scattering and decay processes. This thesis pursues two complementary aims: to advance the computational workflow for formulating and evaluating amplitude models and to apply these methods to baryonic decays in experimental data.

On the computational side, the work introduces a novel approach in which amplitude models are first expressed symbolically in a Computer Algebra System and then automatically translated into array-oriented code for highly parallelised numerical evaluation. Implemented in the widely used dynamic language Python, this approach lowers the barrier for model development and integrates naturally with modern scientific software ecosystems. In addition to accelerating computations, the workflow produces self-documenting amplitude models that make analysis steps transparent and reproducible.

On the physics side, the work establishes a consistent formulation of amplitude models for three-body decays involving final states with spin, particularly baryons, using the Dalitz-plot decomposition method. The approach is validated in two case studies. First, a polarimetry analysis of the decay $\Lambda_c \to pK^-\pi^+$, based on LHCb results, demonstrates numerical correctness to floating-point precision when benchmarked against implementations in C++ and Julia, while enabling efficient uncertainty propagation through large-scale parallelised parameter sampling. Second, an amplitude analysis of $J/\psi \to \bar{p}K^0_S\Sigma^+$ with BESIII data shows that spin-aligned amplitude models provide an improved description of the data compared to earlier analyses that neglected spin-induced interference between topologically distinct decay chains.

The results demonstrate that symbolic, spin-aligned amplitude models achieve both reliable computational performance and correct descriptions of baryonic decay data, while the self-documenting workflow ensures that the models used can be transparently inspected and reproduced.

Zusammenfassung

Die Amplitudenanalyse ist eine Schlüsseltechnik der Hadronphysik, da sie experimentelle Observablen mit den zugrunde liegenden Dynamiken von Streu- und Zerfallsprozessen verknüpft. Diese Dissertation hat zwei komplementäre Ziele: die Weiterentwicklung des rechnerischen Arbeitsablaufes zur Formulierung und Auswertung von Amplitudenmodellen sowie die Anwendung dieser Methoden auf baryonische Zerfälle in experimentellen Daten.

Auf der methodischen Seite führt die Arbeit einen neuartigen Ansatz ein, bei dem Amplitudenmodelle zunächst symbolisch in einem Computeralgebrasystem formuliert und anschließend automatisch in array-orientierten Code für hochgradig parallelisierte numerische Auswertungen übersetzt werden. Implementiert in der weit verbreiteten dynamischen Sprache Python, senkt dieser Ansatz die Hürden für die Entwicklung von Amplitudenmodellen und integriert sich nahtlos in moderne Softwareumgebungen. Neben der Beschleunigung der Berechnungen können die Amplitudenmodelle direkt als Formeln eingesehen werden, die die einzelnen Analyseschritte transparent und reproduzierbar machen.

Auf der physikalischen Seite wird im Rahmen dieser Arbeit eine konsistente Formulierung von Amplitudenmodellen für Drei-Teilchen-Zerfälle mit spinbehafteten Endzuständen, insbesondere für Baryonen, unter Verwendung der Dalitz-Plot-Dekompositionsmethode bereitgestellt und anhand zweier Fallstudien validiert. Erstens zeigt eine Polarimetrie-Analyse des Zerfalls $\Lambda_c \to pK^-\pi^+$, basierend auf LHCb-Daten, numerische Übereinstimmung bis zur Gleitkommapräzision im Vergleich mit Implementierungen in C++ und Julia und ermöglicht eine effiziente Berechnung von Unsicherheiten. Zweitens zeigt eine Amplitudenanalyse von $J/\psi \to \bar{p}K^0_S\Sigma^+$ mit BESIII-Daten, dass Amplitudenmodelle mit Berücksichtigung der Spin-Ausrichtung die Daten besser beschreiben als frühere Analysen, die spininduzierte Interferenzen zwischen topologisch verschiedenen Zerfallsketten vernachlässigt hatten.

Die Ergebnisse zeigen, dass symbolische Amplitudenmodelle mit Berücksichtigung der Spin-Ausrichtung sowohl eine verlässliche rechnerische Leistungsfähigkeit als auch korrekte Beschreibungen baryonischer Zerfallsdaten liefern, während die Darstellung in Form von Formeln sicherstellt, dass die verwendeten Modelle leicht überprüft und reproduziert werden können.

The Road goes ever on and on,
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say.

— J. R. R. Tolkien, *The Fellowship of the Ring*

Acknowledgements

The road of this thesis research has often joined with the paths of others, and it has been a privilege to travel it with so many generous companions. I would not even have found the way without their guidance and support, and I feel fortunate for the lessons, inspiration, and friendship they provided throughout.

Above all, I would like to thank my advisor, Prof. Dr. Miriam Fritsch, without whom I would not have embarked on this journey in the first place. I am especially thankful for the regular discussions she organised, for the insightful feedback she provided on the different directions I pursued, and for the freedom she gave me to follow new ideas in developing the analysis framework. I am likewise grateful for the many opportunities she gave me to participate in meetings within the fields of hadron physics and computing.

Several of the most important collaborations of this thesis began at such meetings. Chief among them has been my collaboration with Prof. Dr. Mikhail Mikhasenko. Since the polarimetry studies, he has guided me in correctly implementing and verifying spin-aligned amplitude models. This work became an important foundation of the results presented in this thesis and would not have been possible without his expertise and support.

In the initial stages of this work, I benefited greatly from the advice and encouragement of Dr. Stefan Pflüger, who introduced me to the world of amplitude analysis and the existing software landscape. Our daily programming sessions, even when remote during the pandemic, shaped my understanding of good software design as we reworked the framework into its current symbolic and array-oriented form. I would also like to express my gratitude to Prof. Dr. Wolfgang Gradl, whose feedback and perspective during our group meetings were important in shaping the features of the analysis framework.

Alongside these collaborations, the day-to-day life at the institutes mattered just as much. I am grateful to my office mates and colleagues, first in Mainz and later in Bochum, for the many discussions during lunch and coffee breaks. Over the years, our office became a place where serious work was balanced with the humour we shared each day, and in the later stages of my dissertation, I especially valued how they motivated me to stay focused on the writing. I also owe much to my friends – those I met in Bochum, who made the city feel like home, and those in the Netherlands, whose support from afar has accompanied me throughout.

Finally, I would like to thank my mother and my brother; the regular trips back home were not only a welcome way to recharge my batteries, but also helped me stay grounded outside my research. My father inspired me to pursue a career in physics and nurtured my interest in computing, and although he sadly did not live to see this part of the journey, his backing and encouragement continue to accompany me.

Table of contents

Introduction

At the smallest scales of nature, particle and nuclear physics seek to unravel the fundamental constituents of matter and the forces that bind them. From the structure of nuclei to the search for fundamental particles, the field seeks a unified understanding of nature's building blocks and their interactions. Particle physics in particular has long captured the public imagination: colliding particles at the Large Hadron Collider, the hunt for the Higgs boson, and the effort to bring the fundamental interactions under one theoretical roof in the Standard Model. This is the world of **high-energy physics**, where theorists strip matter to its most elementary constituents and experiments probe phenomena at ever smaller scales.

Alongside this pursuit lies a domain less widely known yet no less fundamental. This is the world of **hadron physics**, which focuses not on the most elementary particles, but on the hadronic states that the strong force binds together. The subdiscipline of **hadron spectroscopy** in particular classifies and analyses the discrete, stable and unstable states produced by strong-force dynamics, much as atomic spectra reflect underlying electron configurations. It is a study not of ultimate building blocks, but of the intricate structures that emerge from them.

Where high-energy physics often aims to reduce, hadron spectroscopy seeks to classify, understand, and predict emergent structures in the strongly interacting regime [1; 2]. The challenge here is different: hadron physicists must grapple with the non-perturbative regime of Quantum Chromodynamics [3]. At these lower energies, the basic question of "What states can exist?" requires experimental guidance. Each newly observed state provides additional information about the underlying binding mechanism and the principles that govern the formation of hadrons. The discipline has therefore historically always drawn on experimental data to spot patterns, guide classification, and build effective models of strong interactions. Among the methods developed for this task, **amplitude analysis** serves to process experimental data, untangling interfering contributions, and extract information about short-lived hadronic states.

The distinction between high-energy physics and hadron physics is a relatively modern development. Both disciplines trace their roots to **nuclear physics**, which emerged in the early 20th century with the discovery of the atomic nucleus, the exploration of radioactive decays, and the first investigations into nuclear forces. Early accelerators and detectors were originally built to probe the structure of nuclei, but soon began uncovering phenomena that pointed beyond them and resulted in the discovery of new particles and short-lived resonances in scattering experiments [4]. As the energy reach of experiments grew from the 1960s, research began to diverge [5]. One line of inquiry led toward ever higher energies and the search for more elementary constituents. The other gave rise to hadron physics, which remained focused on the emergent complexity of the strong interaction and the spectrum of hadrons it produces [6]. Both, however, ultimately study aspects of the same underlying theory that manifests differently across energy scales.

Against this historical backdrop, the present thesis is situated within hadron physics, and more specifically in the study of **baryon spectroscopy**. Here, the spin degrees of freedom of fermions add a level of complexity that makes theoretical modelling especially demanding. Nucleon excitations in particular form a challenging but rich arena: their spectrum is dense, resonances

often overlap, and many predicted states remain difficult to isolate experimentally. Addressing these excitations requires both advanced parametrisations and computational techniques capable of evaluating models across vast experimental data samples.

The central motivation of this work is therefore the need for flexible, transparent, and computationally efficient tools that can bridge the gap between the intricate theoretical description of hadron physics and the rapidly growing body of experimental data. Much of this description rests on scattering theory and partial-wave expansions, which provide the language in which amplitudes are modelled. At its core, this thesis develops a framework that unites symbolic, computer-algebra–assisted model building with array-based high-performance computing to produce a transparent, extendable, and self-documenting workflow to address these needs. The techniques are validated through two case studies – nucleon excitations in charmonium decays and polarimetry in Λ_c decays.

Beyond computational efficiency, this work places strong emphasis on **reproducibility**, both in methodology and in exposition. The computational methodology evolved with an eye for transparency and long-term maintainability, with lock files, documentation, and open code to make analyses verifiable and extendable. Hadron spectroscopy sits at the crossroads of many disciplines – experiment, theory, statistics, and computing – and newcomers must master a wide range of skills that are rarely spelled out in one place. For this reason, the first three chapters provide an overview of the theoretical background that builds up concepts gradually. For a more interlinked reading experience, the thesis is also available in an online form at redeboer.github.io/phd-thesis, the source code of which can be easily reused and extended. The underlying philosophy is simple: in hadron spectroscopy, with its many intersecting disciplines, both theory and code should strive for reproducibility, clarity, and accessibility.

The remainder of the thesis is organised to first establish the physics background and then develop and apply this methodological framework. The first three chapters set the stage by introducing hadron spectroscopy with emphasis on symmetries, quantum numbers, and nucleon excitations, and by developing the theoretical framework of amplitude analysis. These chapters are not meant as original contributions but as a structured overview of the relevant physics background, presented with an emphasis on clarity and gradual build-up of concepts. Chapters 4 and 5 present the methodological contribution of this work, namely new computational techniques based on symbolic representations and high-performance computational implementations. Finally, Chapters 6 and 7 turn to the experimental situation and demonstrate the methods with real data from the BESIII and LHCb experiments. The closing chapter offers a summary and points to future developments, notably the implementation of more advanced amplitude-model parametrisations using symbolic expressions and the exploration of new optimisation techniques that array computing makes feasible.

1 Hadron physics

The origins of hadron physics lie in the same early 20th-century efforts that shaped nuclear and high-energy physics. Following the discovery of the electron in 1897, Rutherford's scattering experiments between 1906 and 1913 revealed a dense atomic nucleus by scattering helium nuclei (α particles) on a metal foil [7], which he later showed in 1919 as containing **protons** [8]. The experiments mark the beginning of particle scattering, where distributions are used as a quantitative measure to extract information about the microscopic scale (Figure 1.1), and opened the door to formulating the problem in terms of probabilities rather than classical trajectories (Chapter 2).

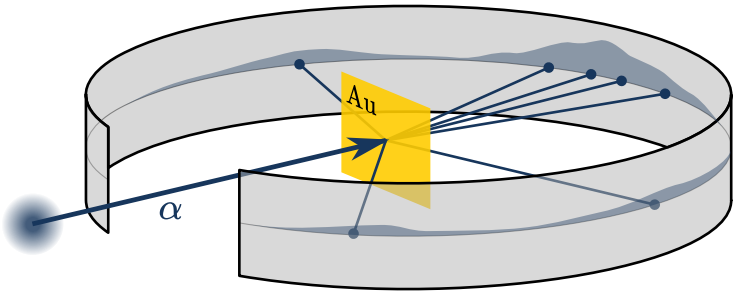


Figure 1.1. Sketch of Rutherford's scattering experiments, in which α particles from a radioactive source on the left were deflected by the nuclei in the gold foil (Au). Deflected particles hit a fluorescent screen around it and some are deflected at large angles.

By itself, however, the proton could not account for the differing masses of isotopes, which called for a neutral counterpart to the proton [9]. The puzzle was resolved in 1932, when James Chadwick proved the existence of a **neutron** [10], a particle with similar properties to that of the protons, but without charge and with a slightly larger mass. The nature of the force responsible for holding protons and neutrons – collectively known as **nucleons** – together within the atomic nucleus remained unresolved. This nuclear force had to be strong enough to overcome the electrostatic repulsion between positively charged protons, yet short-ranged enough to prevent nuclei from growing indefinitely large.

1.1. Symmetries and quantum numbers

At the same time, quantum theory had revealed that particles such as the proton, neutron, and electron possess an intrinsic form of angular momentum known as **spin**. The idea was introduced to explain a puzzling feature in atomic spectra: certain spectral lines appeared to be subtly split. In 1925, Uhlenbeck and Goudsmit proposed that electrons carry an internal twist or spin ("Eigenrotation") that interacts with the magnetic field generated by their motion

around the nucleus, producing this fine-structure splitting [11]. This new degree of freedom was later formalised within quantum mechanics as an intrinsic angular momentum, the operator of which is denoted by $\mathbf{S}$, which couples to the orbital angular momentum $\mathbf{L}$ of the particle's motion to form the total angular momentum $\mathbf{J} = \mathbf{L} + \mathbf{S}$ (see Section 3.2). This relation is known as **angular-momentum addition**, and it reflects the underlying symmetry group SU(2), the special unitary group of 2×2 matrices,

$$\mathrm{SU}(2) = \left\{ \boldsymbol{U} \in \mathrm{Mat}_{2\times 2}(\mathbb{C}) \;\middle|\; \boldsymbol{U}^\dagger \boldsymbol{U} = \mathbf{1},\ \det \boldsymbol{U} = 1 \right\}, \tag{1.1}$$

whose algebra enforces that angular momenta can take only discrete total values $s = 0, \frac{1}{2}, 1, ...$ and discrete projections $-s, -s+1, \dots, +s$ along a chosen quantisation axis. For a spin-$\frac{1}{2}$ particle such as the electron, this means only two possible outcomes, conventionally called "up" and "down".

The quantisation of spin not only fixes the projections of individual particles, but also dictates how systems of identical particles behave collectively. Particles with half-integer spin, such as electrons and nucleons, obey the Pauli exclusion principle [12], which forbids two identical particles from occupying the same quantum state. These particles are known as **fermions**, since their wave functions are antisymmetric ("flip sign") under exchange and follow Fermi–Dirac statistics [13; 14]. In contrast, particles with integer spin are called **bosons** and follow Bose–Einstein statistics [15], which allow any number of identical particles to share the same state. This distinction also determines which types of particles can mediate forces: bosons can be exchanged between fermions, as they are unrestricted by exclusion.

Since the photon that mediates the electromagnetic force was already known to obey Bose–Einstein statistics, it was natural to expect that the nuclear force would be carried by a boson as well. In 1935, Yukawa proposed a new particle – the **meson** – to transmit the interaction between nucleons, "just as the electromagnetic field is accompanied by the photon" [16]. To account for the short range of the nuclear force, he argued that the mediator must be far heavier than the electron, since light particles give rise to long-range forces, but lighter than the proton so that it could still be produced in nuclear processes. He therefore predicted a mass between that of the electron and proton – hence, "meson" – which would yield a range comparable to the size of the nucleus.

As Yukawa anticipated, cosmic rays soon offered a glimpse of this new particle. In 1937, Anderson and Neddermeyer observed a charged particle that behaved like a heavy electron [17; 18] ("mesotron" [19]). It was mistaken for Yukawa's meson, as it lay in the predicted mass range, and it took another ten years before it was shown that this mesotron did not interact strongly with the nucleus [20]. Rather, the mesotron was a secondary decay product [21] – a "μ-meson" [22], or **muon** – of a primary particle – a "π"-meson, or **pion** – which did interact through nuclear scattering and absorption. Like the electron, the muon followed Fermi–Dirac statistics and interacted electromagnetically, and was therefore put under a new category of **leptons** [23, p. VIII].

What made these pions interact with nucleons? Already in 1932 [24], Heisenberg had introduced the idea of a new spin-like quantum number ρ – later dubbed **isotopic spin**, or **isospin** [25] – to explain why protons and neutrons interacted so similarly via the strong force [26]. The two were seen as a "doublet" representation of a single nucleon state that are distinguished by a degree of freedom (isospin), much like up and down spin states of the electron. Yukawa's meson field naturally fit into this symmetry scheme as a force mediator with *integral* isospin, so that a nucleon flips its isospin by emitting a pion [27]. Extending the concept of isospin to force

mediators even led to the prediction of a neutral pion state to complete the "triplet" of integral isospin states [28], which was discovered soon after the charged pions [29].

At this stage, symmetry principles and quantum numbers had become an important tool for classifying the proliferating array of new particles. As detection methods improved, especially with accelerators in the 1950s, a growing number of short-lived particles were discovered (see Table 1.1). To bring order to this chaos, physicists grouped the heavier, strongly interacting particles – including the proton and neutron – under the common label **baryons** [30], distinguishing them from the lighter mesons. Some, however, like "K mesons" (**kaons**) and "V particles" (**hyperons** [31]), displayed unexpectedly long lifetimes or unusual decay patterns. These new states didn't fit into existing categories and formed a bewildering "particle zoo".

	Particle	Discovered	Predicted
e^-	electron	1897 [32]	
p	proton	1919 [8]	
n	neutron	1932 [10]	1920 [9]
e^+	positron	1932 [33]	1931 [34]
μ^-	muon	1936–1937 [35;18]	
$\pi^\pm$	charged pion	1947 [21]	1935 [16]
π^0	neutral pion	1950 [29]	1938 [28]
$K^\pm$	charged kaon	1947 [36]	
K^0	neutral kaon	1949 [37]	
Λ	Lambda	1950 [38]	
Ξ	Xi or "cascade"	1952 [39;40]	
$\Sigma^\pm$	Sigma	1953 [41]	
η	eta	1961 [42]	1959 [43]
Ω	Omega	1964 [44]	1961 [45]

Table 1.1. Overview of the main findings of hadrons and other related particles that eventually led to the categorisation scheme for this "particle zoo". The discovery years are not to be taken as absolute: often, first findings were mere indications that were confirmed a few years later [46].

The proliferation of strongly interacting particles made it clear that existing categories were no longer sufficient. Rather than postulate entirely new constituents, physicists sought regularities in how these particles could be grouped and compared [47]. Quantum numbers – discrete labels associated with conserved quantities – emerged as the guiding principle for this approach. The earlier concept of isospin, which had already unified protons, neutrons, and pions into symmetrical multiplets under an SU(2) structure, provided a blueprint: perhaps other groupings could be understood in the same way.

Yet some of the newly discovered states, such as kaons and hyperons, exhibited unusual behaviour. For example, while the Λ hyperon was produced abundantly in cosmic rays and collisions, its dominant decay $\Lambda \rightarrow p\,\pi^-$ would violate isospin conservation if it proceeded through the strong force. Instead, the decay occurred only via the weak interaction, giving the Λ hyperon its unusually long lifetime for a hadron. To account for such anomalies, a new additive quantum number – **strangeness** – was proposed [48]. Conserved in strong interactions but violated in weak decays, it explained why these particles were readily produced but decayed slowly, and allowed them to be systematically included in extended symmetry schemes [49; 50].

As the study of neutral mesons and vector mesons advanced in the 1950s, physicists realised that the concept of charge conjugation – exchanging particles with their antiparticles – could be used to classify states that are their own antiparticles (self-conjugate). This led to the definition of C**-parity** [51], a quantum number that tells whether the wave function of a self-conjugate particle changes sign under this operation. Alongside charge conjugation, **parity** (P) described how a particle's wave function transforms under spatial inversion $\vec{x} \to -\vec{x}$. To extend similar symmetry constraints to charged members of isospin multiplets, for which *C-parity* cannot be defined, physicists introduced G**-parity** [52], defined as $G = Ce^{i\pi I_2}$. The prediction and discovery of the η meson in 1961 [45; 42] provided another example of a neutral meson whose decays could be constrained by these discrete symmetries. Together with total spin J, isospin I, and strangeness, these quantum numbers formed a powerful labelling scheme. It was in this context that the collective term **hadron** was coined [53], denoting all particles that participate in strong interactions, in contrast to leptons and gauge bosons.

Although originally introduced case by case to solve specific experimental puzzles, we today understand these quantum numbers as manifestations of underlying symmetry principles, with conservation laws arising from those symmetries. Some, like electric charge or baryon number, are *additive* because they derive from continuous internal symmetries (such as a phase rotation, forming the group $\mathrm{U}(1) = \{e^{i\phi} \mid \phi \in \mathbb{R}\}$), so the total value is obtained by simple summation over constituents. Additive quantum numbers also include strangeness and the separate lepton numbers that apply to each lepton type (e, μ, τ). Others, like the parity operations, are *multiplicative*, because they reflect discrete symmetries such as spatial inversion or particle–antiparticle exchange; in these cases the wave function is either preserved $(+1)$ or changes sign (-1). For orbital motion, parity includes an additional factor $\times(-1)^L$ with orbital angular momentum L. Finally, spin and isospin follow the rules of SU(2) *addition*, reflecting the non-Abelian continuous algebra of rotations, where their conservation takes the form of angular-momentum coupling. Hadrons are not labelled by the intrinsic spin s of their constituents, but by the total angular momentum J, obtained through angular-momentum addition. Together, the labels J for total angular momentum and I, I_3 for isospin and its projection form the basis of the hadron classification scheme, where states are organised by combinations such as $J^{PC}(I^G)$. These categories are summarised in Table 1.2.

Quantum number	Symbol	Strong	EM	Weak	Conservation rule
Electric charge	Q	✓	✓	✓	Additive
Spin	J	✓	✓	✓	SU(2) addition
Baryon number		✓	✓	✓	Additive [54]
Lepton number		✓	✓	✓	Additive
Strangeness		✓	✓		Additive
Parity	P	✓	✓		Multiplicative $\times(-1)^L$
C-parity	C	✓	✓		Multiplicative
G-parity	G	✓			Multiplicative
Isospin	(I, I_3)	✓			SU(2) addition

Table 1.2. Overview of the main quantum numbers and their conservation laws for the strong force, electromagnetic (EM) force, and weak force.

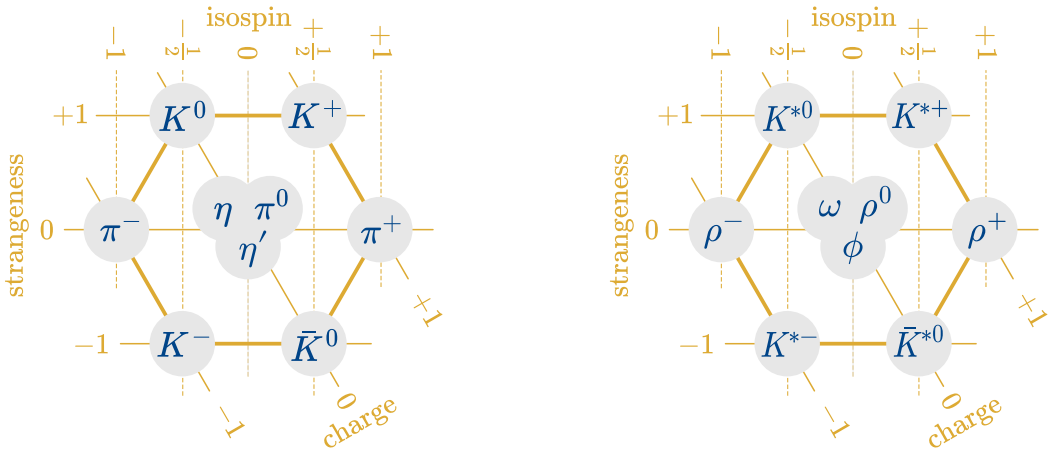
1.2. The Eightfold Way and the quark model

As more hadrons were discovered, their regularities pointed toward an even deeper symmetry, what we now call **flavour** symmetry. In the early 1960s, Gell-Mann and Ne'eman independently proposed the Eightfold Way [45; 55; 56], a classification scheme based on SU(3) group symmetry. Isospin formed an SU(2) subgroup, as expressed in Equation (1.1), and the inclusion of strangeness called for an extension of that symmetry. Hadrons were organised into **multiplets**, sets of states related by the symmetry, much as angular-momentum coupling produces families of states in atomic physics. The structure of these multiplets is conveniently displayed in weight diagrams that plot the states according to their strangeness S and isospin projection I_3 (Figure 1.2 for mesons and Figure 1.3 for baryons). When S and I_3 are on the vertical and horizontal axes, the electric charge Q runs diagonally across the diagram, as it is related to S, I_3, and baryon number B via the Gell-Mann–Nishijima relation,

$$Q = I_3 + \tfrac{1}{2}(B + S)\,.$$

Within each multiplet, states are distinguished by their total spin and parity, which result from coupling intrinsic spin to orbital angular momentum L. The clearest patterns arise for the **ground states** with $L = 0$. These give the $J^P = 0^-$ **pseudoscalar mesons** (e.g. pions and kaons) and the $J^P = 1^-$ **vector mesons** (e.g. ρ and K^*). Higher orbital excitations generate further families such as the **scalar** (0^+), **axial-vector** (1^+), and **tensor** (2^+) mesons, but these have larger masses and tend to mix more strongly.

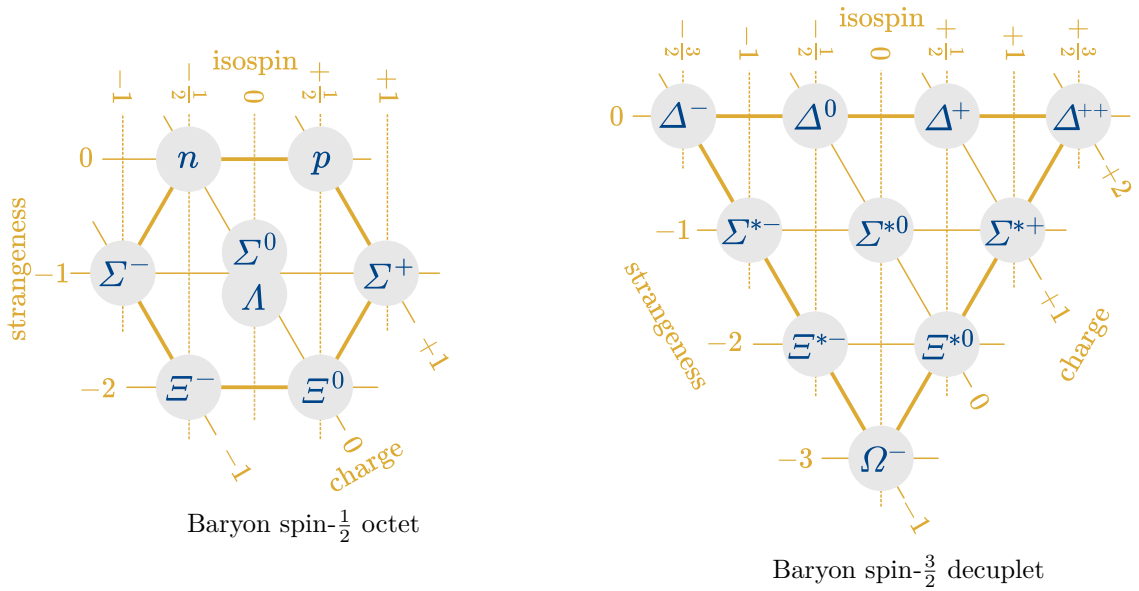
For mesons, the most characteristic pattern is the **octet**: eight states arranged in a hexagonal weight diagram with neutral members in the centre. An additional singlet accompanies the octet, so the observed spectrum forms a **nonet** of nine states. This is why the diagrams in Figure 1.2 show three neutral entries in the centre: one belongs to the octet, the other to the singlet, and together they mix to form the observed η–η' and ω–ϕ pairs. In the pseudoscalar case, the η' meson is sometimes omitted from the nonet diagram, since it is predominantly a singlet state and its mass lies far above that of its octet companions. For vector mesons, the octet and singlet mix better, producing the physical ω and ϕ alongside the ρ and K^*.



Pseudoscalar-meson nonet ($J^P = 0^-$) Vector-meson nonet ($J^P = 1^-$)

Figure 1.2. Multiplets of the ground-state mesons visualised as weight diagrams.

In the baryon sector, shown in Figure 1.3, the pattern differs. The spin-$\frac{1}{2}$ baryons (nucleon, Λ, Σ, and Ξ) form an octet, while the spin-$\frac{3}{2}$ states (Δ, Σ^*, Ξ^*, and Ω^-) arrange into a **decuplet**. A spectacular confirmation of this scheme came in 1964 with the discovery of the Ω^- [44], the predicted missing member of the decuplet, whose mass and decay properties matched the expectations of the Eightfold Way with remarkable precision.



Baryon spin-$\frac{1}{2}$ octet

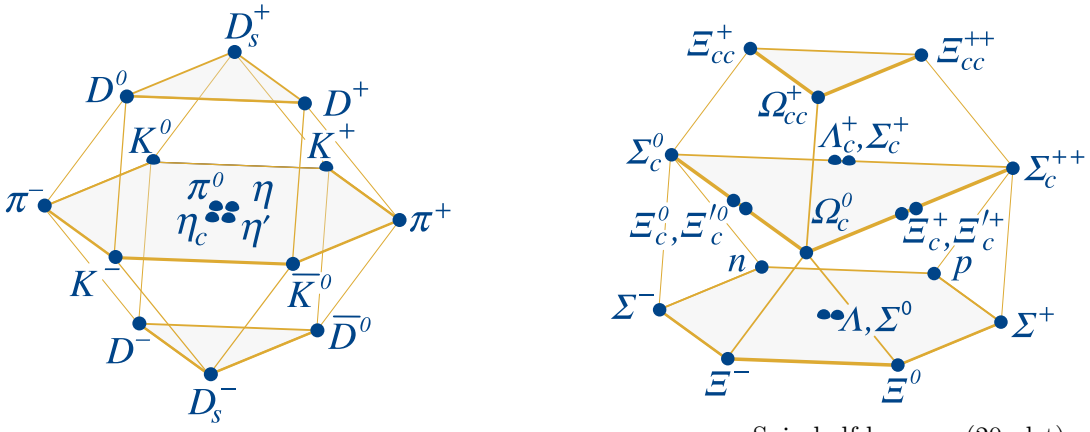
Baryon spin-$\frac{3}{2}$ decuplet

Figure 1.3. Multiplets of the ground-state baryons visualised as weight diagrams.

Out of this symmetry-based classification emerged a new idea: that hadrons were not fundamental, but composite particles made of simpler constituents. These hypothetical building blocks – eventually named **quarks** – were proposed independently by Gell-Mann [57] and Zweig [58] in the mid-1960s. The Eightfold Way's multiplet structure hinted at an underlying triplet, leading to the postulation of three quark types (flavours): **up**, **down**, and **strange**. These carry fractional electric charges ($+\frac{2}{3}$ or $-\frac{1}{3}$ of the electron charge) and transform according to the fundamental representation of SU(3). Because no free particles with such fractional charges were known, Gell-Mann described quarks as "mathematical" entities – permanently confined and not observable in isolation – whereas Zweig's "aces" were presented more straightforwardly as physical constituents [59; 60]. The SU(2) isospin subgroup now reflects the relative numbers of up and down quarks: its projection I_3 equals half the difference between the number of up and down quarks, while the total isospin I determines the multiplet the hadron belongs to.

As experiments ran into other anomalies and puzzling behaviour of the weak interaction, theorists soon realised that a fourth quark, **charm**, was needed to complete the picture, extending the flavour symmetry to an approximate SU(4) symmetry (see Figure 1.4) and explaining the absence of certain flavour-changing neutral currents [61; 62]. Initially conceived as a mathematical tool to encode symmetry patterns [63, p. 73], the quark model soon began to be taken more seriously as a physical picture of hadrons, especially after deep inelastic scattering experiments at SLAC revealed evidence of point-like constituents inside the proton [64; 65].

A theoretical foundation for the strong interaction arrived only later, with the development of **Quantum Chromodynamics (QCD)** in the early 1970s [67]. While the Eightfold Way was based on a *global* SU(3) flavour symmetry, QCD introduced a fundamentally different structure: a



Pseudoscalar mesons (16-plet) Spin-half baryons (20-plet)

Figure 1.4. Extension of the baryon multiplet weight diagrams of Figure 1.3 to an approximate SU(4) symmetry to include the quantum number "charm" in the vertical direction. Adapted from [66, §15].

local SU(3) gauge symmetry. This symmetry, acting in an internal space later dubbed **colour** [68; 69], governs the interactions between quarks and massless vector bosons known as **gluons** [70]. Unlike flavour symmetry, which described patterns among hadrons, this local gauge symmetry dictated the dynamics of the strong force at a fundamental level. At the time, however, it was unclear whether such a theory could be consistent: local gauge theories were thought to become uncontrollably strong at high energies, making calculations divergent and predictions unreliable. A turning point came in 1973 with the discovery of **asymptotic freedom** by Gross, Wilczek, and Politzer [71; 72]. They realised that, in non-Abelian gauge theories like QCD, the coupling strength *decreases* at high energies, allowing for meaningful, perturbative calculations. This behaviour is illustrated in Figure 1.5, which shows experimental measurements of the coupling strength α_s at different energy scales Q. The measurements confirm the predictions of QCD: α_s is large at low energies (large distances), but decreases asymptotically with increasing Q (smaller distances).

This breakthrough explained why quarks, though nearly free in high-energy collisions, can never be observed in isolation. Their new quantum number of **colour charge** – coming in three types (red, green, blue) and governed by a local SU(3) symmetry – implies that only *colourless* combinations can exist as physical states. This resolves puzzles such as the spin-$\frac{3}{2}$ baryons like $\Omega^- = |s\uparrow s\uparrow s\uparrow\rangle$ and $\Delta^{++} = |u\uparrow u\uparrow u\uparrow\rangle$, which would violate the Pauli exclusion principle if quarks carried only flavour and spin. QCD thus predicts a dual behaviour: quarks interact almost as free particles at short distances (asymptotic freedom), but are inseparably bound at long distances, giving rise to **confinement**. The observable spectrum consists solely of colour-singlet states: three quarks of different colours form a baryon, while a quark–antiquark pair forms a meson (see Figure 1.6).

Once such colour-neutral combinations are admitted, the familiar multiplet structure of the Eightfold Way naturally emerges. Quarks transform in the fundamental representation of flavour SU(3), denoted as a **colour triplet 3**. Tensor products of these triplets decompose into irreducible multiplets, reproducing the observed hadron patterns. In particular, mesons arise from $\mathbf{3}\otimes\overline{\mathbf{3}} = \mathbf{8}\oplus\mathbf{1}$ (Figure 1.2), while baryons follow from $\mathbf{3}\otimes\mathbf{3}\otimes\mathbf{3} = \mathbf{10}\oplus\mathbf{8}\oplus\mathbf{8}\oplus\mathbf{1}$ (Figure 1.3).

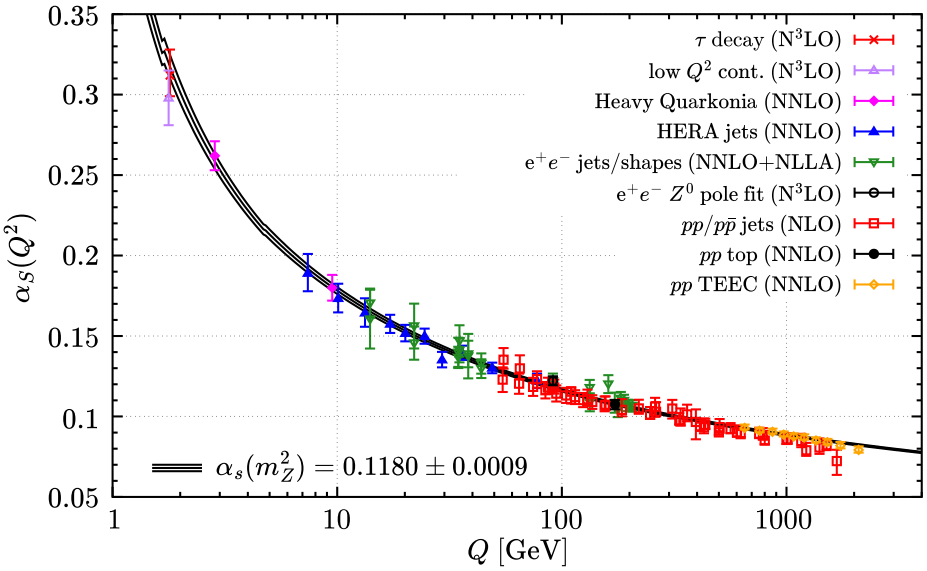


Figure 1.5. QCD coupling strength α_s as a function of energy scale Q with the current average of $\alpha_s(m_Z^2)$ from the Review of Particle Physics of 2023 as input. Coloured points show the calculated values based on NNLO QCD with experimental input. Adapted from [66, Fig. 9.5].

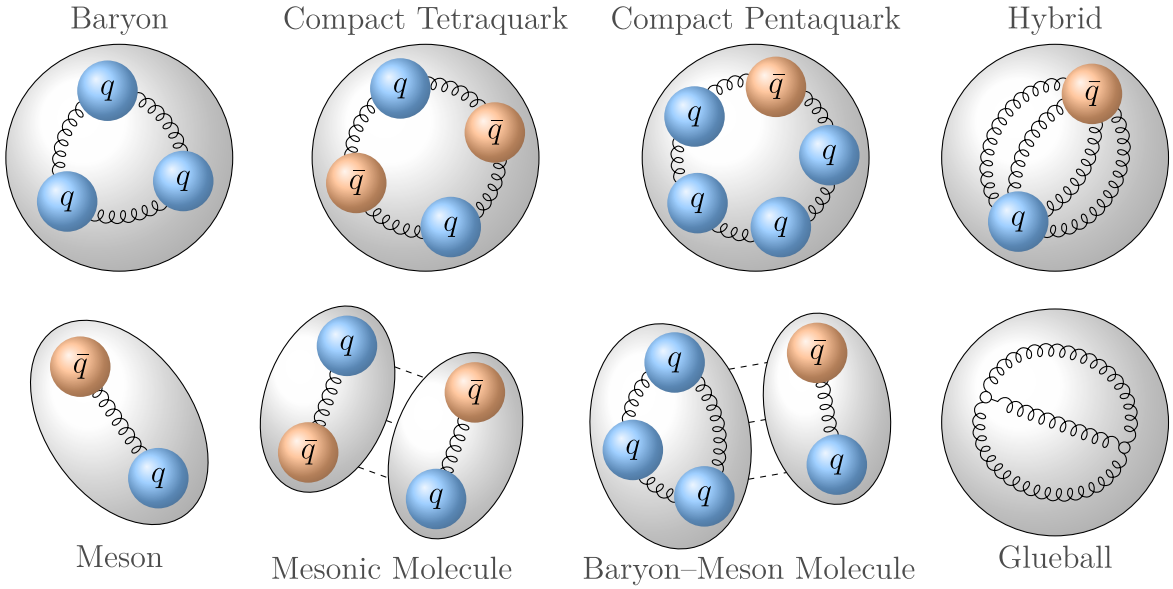


Figure 1.6. Artistic representation of how quarks and gluons combine to form different types of hadrons. Adapted from [73].

Gell-Mann also predicted other colourless combinations, such as **tetraquarks** (two quarks and two antiquarks) and **pentaquarks** (four quarks and one antiquark). In recent decades, other exotic configurations have been discussed, including **glueballs** (bound states consisting of gluons only), **hybrids** (quark–antiquark pairs with gluonic excitations), and **hadronic molecules** (loosely bound states of mesons and baryons). Unlike photons in QED, gluons themselves carry colour charge and can interact with each other, which causes the QCD coupling to grow stronger as the energy decreases. This strong coupling makes quarks and gluons inseparable, but marks the **non-perturbative regime** at low energies, where symmetry principles, phenomenology, and experiment are our only tools for mapping the rich spectrum of bound states emerging from QCD.

Today, we know there are six quark flavours within three generations of increasing masses: up and down, strange and charm, and bottom and top. The lightest pair build stable protons and neutrons, while strange, charm, and bottom quarks give rise to heavier, short-lived hadrons. By contrast, the top quark is so massive and short-lived that it decays before hadronising. Alongside three families of leptons (the electron, muon, tau, and their neutrinos), the force carriers (photon, gluons, W and Z bosons), and the Higgs boson, these quarks complete the **Standard Model** (Figure 1.7), our most fundamental description of nature's building blocks. Yet in the strongly interacting regime, where quarks remain confined, hadron physics investigates the emergent spectrum of composite states, with the aim of understanding the non-perturbative dynamics of QCD and uncovering mechanisms beyond current models.

1.3. Nucleon excitations

Having seen how quantum numbers and symmetries organise hadrons into meson and baryon multiplets, it is natural to ask how these patterns extend beyond the ground states. This question is particularly pressing for nucleons, because they, unlike strange or charmed baryons, are built solely from three light quarks of nearly equal mass. With no heavy constituent to dominate the dynamics, the complex interplay of all three quarks and of the gluons that bind them becomes fully exposed. Historically, scattering experiments revealed an increasing number of baryon resonances whose squared masses and spins follow an approximately linear relationship. These patterns became known as **Regge trajectories** [75]. This invited the question of how such excitations arise in the quark model. Just as atoms have discrete excited states when their electrons occupy higher orbitals, hadrons can exist in configurations where their constituent quarks rearrange their relative motion and spin alignments. These radial and orbital excitations manifest as heavier, unstable resonances that populate the observed Regge trajectories [76].

An example is shown in Figure 1.8, where excitations of the nucleon, also called N^* **resonances**, align along straight lines when plotted as their squared mass M^2 versus orbital angular momentum L. The left panel displays the radial excitation spectrum of positive-parity nucleons, where successive radial quantum numbers n generate parallel trajectories. The right panel uses a "light-front holographic classification" [77], where grouping excitations by parity results in two nearly parallel trajectories.

Non-relativistic quark model

To classify these possible states, the idea of SU(3) flavour symmetry for light hadrons – which applies to hadrons composed of up, down, and strange quarks – can be extended by including the

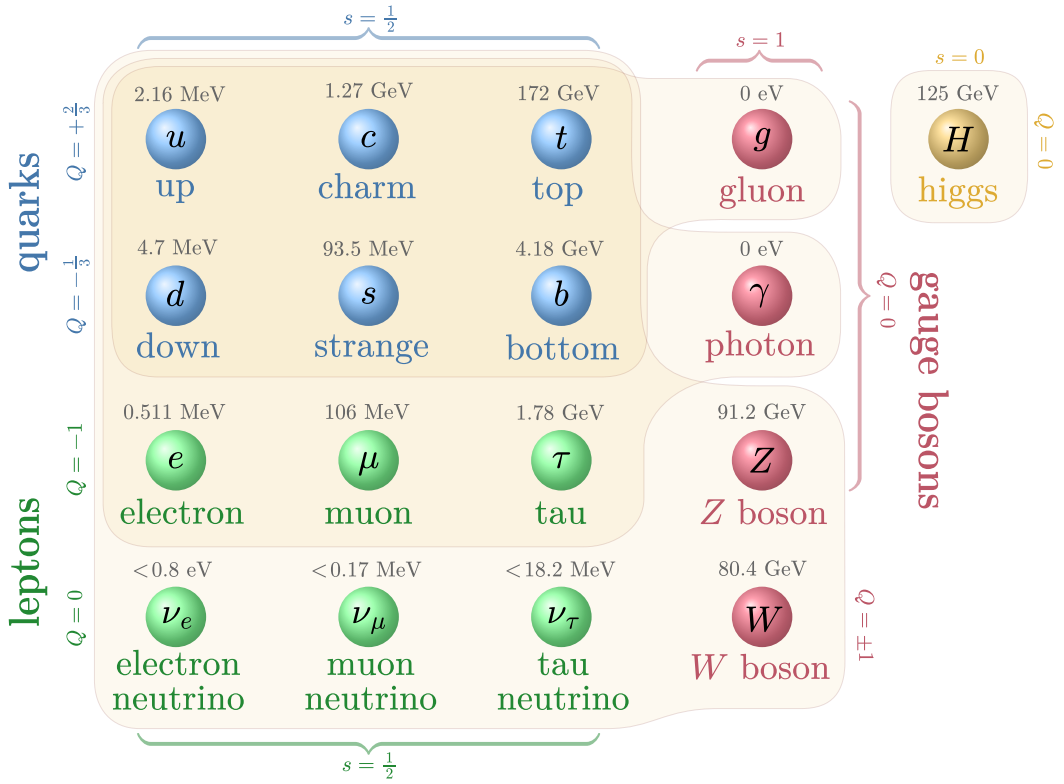


Figure 1.7. The Standard Model of particle physics, showing all known elementary particles and force mediators, coloured by category and generation. The mass, intrinsic spin s, and charge Q of each particle is also indicated. The light-yellow shading in the background indicates which force carriers interact with which matter particle. Inspired by [74].

spin degrees of freedom, which are described by an SU(2) symmetry. In the **non-relativistic quark model** [78], this leads to an approximate SU(6) spin–flavour symmetry,

$$\mathrm{SU}(6) \supset \mathrm{SU}(3)_{\text{flavour}} \times \mathrm{SU}(2)_{\text{spin}} \,, \tag{1.2}$$

where each quark transforms as a 6-dimensional spin–flavour multiplet. Light baryons are then described by the tensor product of three such quark states:

$$\mathbf{6} \otimes \mathbf{6} \otimes \mathbf{6} = \mathbf{56}_{\mathcal{S}} \oplus \mathbf{70}_{\mathcal{M}} \oplus \mathbf{70}_{\mathcal{M}} \oplus \mathbf{20}_{\mathcal{A}} \,.$$

The subscript of each multiplet indicates whether its spin–flavour wave function is symmetric ($\mathcal{S}$), antisymmetric ($\mathcal{A}$), or mixed-symmetric ($\mathcal{M}$) under exchange of any two quarks. Because quarks are fermions, the Pauli exclusion principle requires the total baryon wave function Ψ_{total} to be antisymmetric. In the non-relativistic quark model, this wave function factorises into three parts – colour, spin–flavour, and orbital (spatial):

$$\Psi_{\text{total}} = \Psi_{\text{colour}} \times \Psi_{\text{spin–flavour}} \times \Psi_{\text{space}} \,.$$

The colour part of a baryon is always totally antisymmetric, because $\Psi_{\text{colour}} = \epsilon_{abc} \, |q_a q_b q_c\rangle$, ensuring that the state is colourless. It follows that the product of the spin–flavour and space components must be symmetric. Hence, once the symmetry of the spin–flavour part is specified

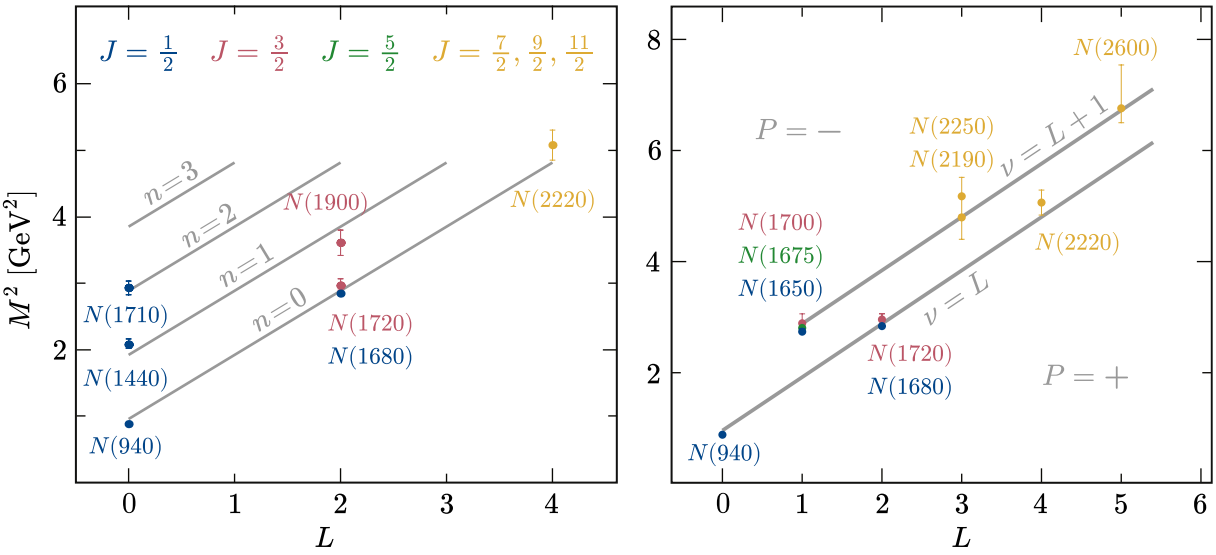


Figure 1.8. Regge trajectories of orbital and radial nucleon excitations. Left: positive-parity nucleons plotted along fixed radial quantum number n. Right: nucleons organised by "light-front" orbital angular momentum ν, giving two trajectories for positive-parity and negative-parity nucleons. The indicated masses are the averages from the PDG of 2014 [66]. Adapted from [77, Fig. 5.5].

(as indicated by the subscripts), the orbital part is fixed to carry the complementary symmetry required to maintain an overall symmetric product.

In particular, the symmetric **56**-plet must combine with a symmetric orbital wave function, which means that all three quarks occupy the lowest S-wave orbital (no orbital angular momentum). Indeed, this **56**-plet contains the familiar ground-state baryons. These are the spin-1/2 octet and spin-3/2 decuplet shown in Figure 1.3 of the $\mathrm{SU}(3)_{\text{flavour}} \times \mathrm{SU}(2)_{\text{spin}}$ subgroup of Equation (1.2). By contrast, the two $\mathbf{70}_{\mathcal{M}}$ multiplets have spin–flavour wave functions that are mixed-symmetric under quark exchange. To maintain the required total antisymmetry, the orbital part must then also be mixed-symmetric, which naturally occurs when the three quarks are excited into states with non-zero orbital angular momentum. For example, a P-wave ($L = 1$) orbital configuration introduces mixed symmetry into the spatial part, allowing these $\mathbf{70}_{\mathcal{M}}$ states to describe many of the excited baryons with negative parity for $L = 1$. The $\mathbf{20}_{\mathcal{A}}$ multiplet has a spin–flavour wave function that is fully antisymmetric, so its orbital part must be fully symmetric, like the **56**. However, because of its antisymmetric spin–flavour structure, it does not appear in the simplest non-relativistic quark model for ground-state baryons and is regarded mainly as a theoretical curiosity, since no clear experimental evidence exists for such states.

In addition to these orbital excitations, the quark model also predicts **radial excitations**, where the quarks remain in the same spin–flavour and orbital configuration, but their spatial wave function acquires additional nodes, much like atomic orbitals. The Roper resonance $N(1440)$ is a classic example, which was traditionally interpreted as the first radial excitation of the nucleon within the $\mathbf{56}_{\mathcal{S}}$ multiplet. Its structure, however, may be more intricate: strong coupling to meson–baryon channels suggests the presence of a surrounding meson cloud, which could also account for its unexpectedly low mass [79]. These radial excitations add even more predicted nucleon states to the spectrum, many of which have yet to be observed.

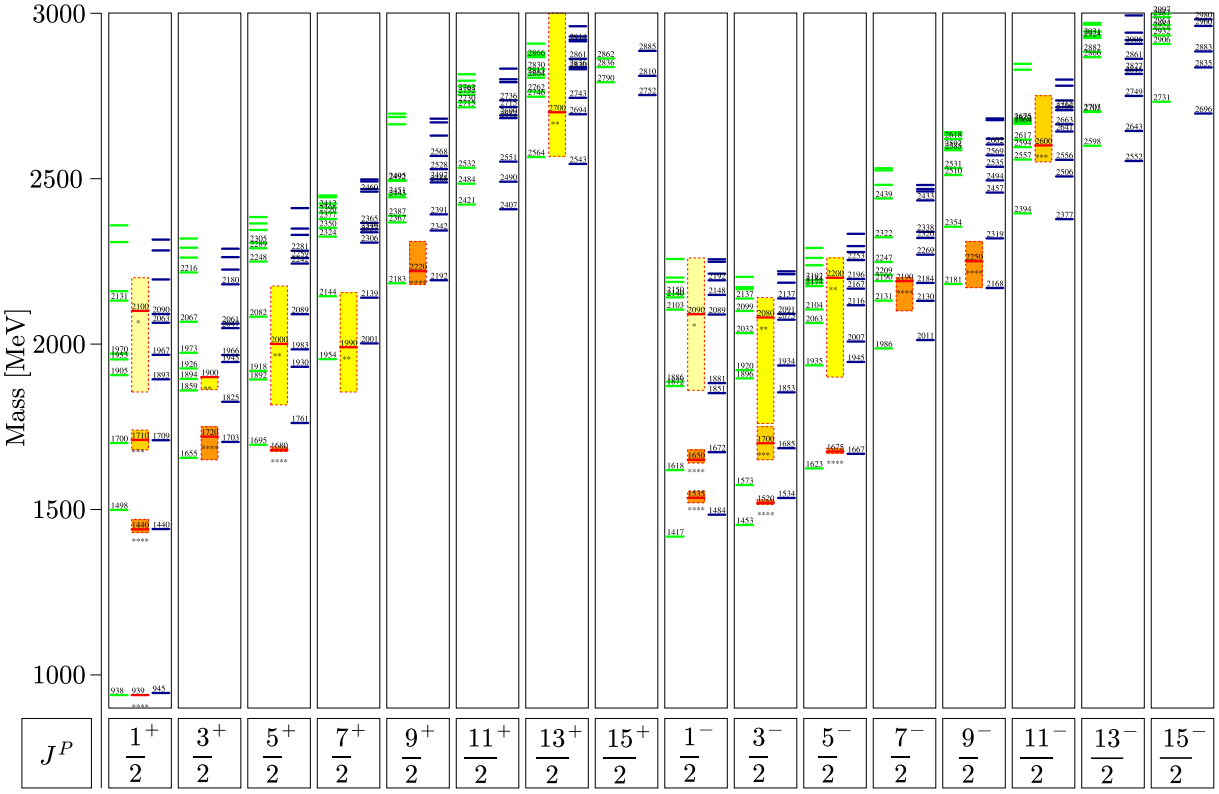


Figure 1.9. Comparison between theoretical predictions and experimental data for the N^* spectrum. In each J^P bar, the left and right sides show predictions from [81] and [82], respectively. The middle column shows mean experimental values Review of Particle Physics of 2010 [66]. Adapted from [76, Fig. 2; 82, Fig. 3].

The non-relativistic quark model is just one of several frameworks used to describe the nucleon excitation spectrum. Other approaches include effective field theories, lattice QCD calculations, and light-front holography [80]. These frameworks aim to reproduce the masses of experimentally observed resonances, but typically predict many more states than have been confirmed. Figure 1.9 shows the measured N^* spectrum placed between two sets of predictions from theoretical models. While the overall structure is reproduced, clear mismatches remain. Some predicted states, such as the many excited states predicted in the $\mathbf{70}_{\mathcal{M}}$ sector, have not been observed in experiment, while some observed states cannot easily explained by theory. This "**missing resonance** puzzle" underscores that the nucleon spectrum reflects dynamics beyond the simplest constituent-quark picture and provides a testing ground for non-perturbative QCD.

N^* resonances in scattering processes

Unlike strange or charmed baryons, where a heavier quark dominates the system and helps simplify the dynamics, the nucleon consists solely of up and down quarks with nearly identical masses (approximately 2.2 MeV and 4.7 MeV, respectively). This absence of an internal mass hierarchy means that all three quarks must be treated on equal footing, making the nucleon an ideal system for testing how confinement and dynamical chiral symmetry breaking emerge from the underlying quark–gluon interactions [83]. As we saw in Figure 1.3, light baryons fall into

spin-$\frac{1}{2}$ or spin-$\frac{3}{2}$ multiplets. Excitations of the latter are the Δ **resonances**, while excitations of the former are the N **resonances** or N^*'s [66, §81], which are the focus of this study.

Most N^*'s are unstable and have to be produced and observed indirectly. Instead, they appear as intermediate states in reactions where an incoming probe excites a nucleon target (usually a proton). Historically, **pion–nucleon scattering** (πN) in fixed-target experiments provided the first systematic evidence for nucleon excitations, starting in 1957 [76]. Examples are experiments at the Los Alamos Meson Physics Facility (LAMPF), at Brookhaven National Laboratory (BNL) with the Crystal Ball detector, and at the Paul-Scherrer-Institute (PSI) in Switzerland. In these experiments, a pion beam excites the nucleon to a Δ^* or N^* state, which then decays into final states such as πN, $\pi\pi N$, or ηN. An advantage of pion–nucleon scattering is that the number of contributing partial waves (the different angular-momentum components in which the scattering process can be decomposed) remains relatively limited, which makes it easier to extract resonance properties, although this also restricts the range of excitations that are produced [76].

Photo- and electro-production experiments such as the Crystal Barrel detector at the CB-ELSA experiment (Bonn), the A2 experiment with the Crystal Ball detector at the Mainz Microtron (MAMI), and the CEBAF Large Acceptance Spectrometer (CLAS) at Jefferson Laboratory (JLab) later extended the range of possible analyses. **Photo-production experiments** use a beam of real photons, while in **electro-production experiments**, electrons are scattered off the nucleon, producing virtual photons that excite the nucleon. These experiments not only have a higher luminosity than pion beams, but also allow for a wider energy range and more complex final states. This gives access to higher-mass resonances and provides additional information through polarisation observables. However, both photo- and electro-production involve a more complex initial state than hadronic scattering, giving rise to a much larger number of contributing partial waves.

The dominant decay channel for both N^* ($I = \frac{1}{2}$) and Δ^* ($I = \frac{3}{2}$) resonances is the single-pion mode ($N\pi$), often accompanied by multi-pion decays ($N\pi\pi$). In addition, N^* states can decay into channels such as ηN – which cannot couple to Δ^* and thus serves as an isospin filter – or into kaon–hyperon final states (e.g. $K\Lambda$, $K\Sigma$). Single-pion decays were studied first and have the lowest energy threshold, giving access to lower excitations. They are also the most straightforward to analyse, as they involve only two particles in the final state and have limited numbers of partial waves. Multi-pion decays, on the other hand, often involve intermediate resonances like the Δ.

Since N^*'s are short-lived, they appear as broad resonances in the invariant mass distribution of the final state particles. However, a characteristic feature across all these reactions is the appearance of sharp structures when new channels open. In Figure 1.10, the total cross section for $\gamma N \to \eta N$, measured by the A2 experiment, shows resonance peaks interrupted by sudden drops or kinks at the thresholds for $K\Sigma$, ωN, and $\eta' N$. These threshold effects indicate that the reaction strength is redistributed once additional final states become kinematically accessible.

To fully disentangle broad, overlapping N^* resonances, modern studies therefore rely on coupled-channel analyses, which combine data from the different production mechanisms and from multiple final states into a single, consistent parametrisation of the energy-dependent dynamics. We discuss the techniques that these programmes employ in Section 2.4. The most well-known research programs are (in chronological order) the Scattering Analysis Interactive Dial-in (**SAID**) of George Washing University [85; 86], the Kent State University model [87], the Mainz Unitary Isobar Model (**MAID**) [88], the **Bonn–Gatchina** model [89], the **Jülich–Bonn** and Jülich–Bonn–Washington model [90; 91], and the **Laurent–Pietarinen** formalism [92]. An

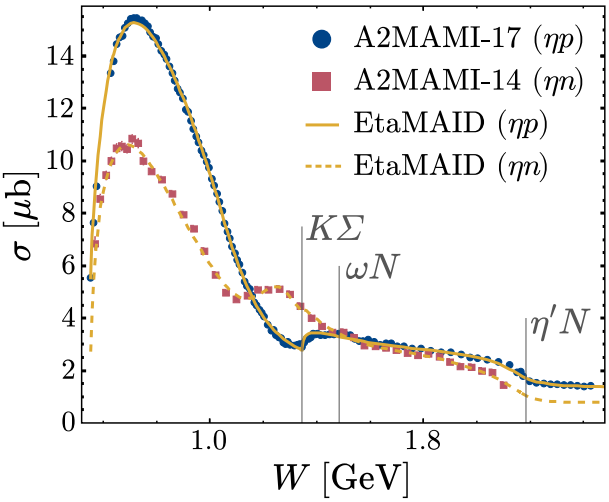


Figure 1.10. Total cross section as a function of center-of-mass energy W in $\gamma N \to N^* \to \eta N$ reactions studied by the A2 collaboration at MAMI, compared with EtaMAID predictions. Dips or edges in the cross section reveal threshold openings into other channels. Adapted from [84].

example by predictions of the MAID model for η and η' photo-production (EtaMAID) is shown in Figure 1.10.

The success of these models lies in their ability to describe the growing body of experimental data. By fitting a wide range of reactions simultaneously, they allow resonance signals to be disentangled from background channels and from one another. The outcome is reflected in Table 1.3, which shows the N^* excitation spectrum currently determined from partial-wave analyses of experimental data. The table uses star ratings that indicate the degree of confidence assigned by the Review of Particle Physics [66]. As can be seen from the colours that highlight newer findings since 2004, the understanding of N^* excitations in the higher mass range has improved significantly. However, as discussed, not all states can be explained well by theory, and many states that are predicted, have not yet been observed.

N^* resonances in charmonium decays

Scattering experiments have long been the main source of information about nucleon excitations, but they are not the only way to study these states. Some of the higher-mass excitations listed in Table 1.3 have also been observed in decays of the J/ψ **meson** and higher charmonium states at the BESIII experiment. **Charmonium** denotes bound states of a charm quark and an anticharm quark ($c\bar{c}$) and includes states like J/ψ. In these decays, the charmonium state produces an antinucleon and a nucleon, one of which can be an excited state. The main disadvantage is that the recoiling antinucleon limits the available phase space, and the multi-body final state can cause interference between different subsystems of completely different hadronic states. However, these decays have clear advantages: the J/ψ meson provides a precisely known production energy and well-defined initial quantum numbers, which reduces the number of partial waves and polarisation observables. Moreover, Δ resonances are heavily suppressed: because the J/ψ meson carries no isospin, the $\bar{N}N^*$ pair must couple to $I = 0$, which excludes Δ^* states

	J^P	mass	width	overall	$N\gamma$	$N\pi$	$\Delta\pi$	$N\eta$	ΛK	ΣK	$N\rho$	$N\omega$	$N\eta'$	$N\sigma$	$N_{1440}\pi$	$N_{1520}\pi$	$N_{1535}\pi$	$N_{1680}\pi$
p, n	$1/2^+$			****														
$N(1440)$	$1/2^+$	1366±3	192±4	****	****	****	***				**			**				
$N(1520)$	$3/2^-$	1506±2	112±3	****	****	****	****	****	****		****							
$N(1535)$	$1/2^-$	1494±7	115±10	****	****	****	**	****	***	**	**			*	*			
$N(1650)$	$1/2^-$	1664±10	105±6	****	****	****	***	*	****	**	**			*	*			
$N(1675)$	$5/2^-$	1662±5	130±8	****	****	****	****	*	*	*	****			*				
$N(1680)$	$5/2^+$	1676±6	117±4	****	****	****	****	*	*	*	**	***			**	*		
$N(1700)$	$3/2^-$	1745±40	420±50	***	**	***	***	*	**	*	**	**		*	*			
$N(1710)$	$1/2^+$	1696±10	155±15	****	****	****	**	***	**	**	****	*		**	*			
$N(1720)$	$3/2^+$	1710±20	180±30	****	****	****	***	*	**	***	**	*	*	**	*	**	**	
$N(1860)$	$5/2^+$	1870±25	290±40	***	*	**	**	*			***			*				
$N(1875)$	$3/2^-$	1855±17	260±20	***	**	**	*	**	*	*	*	*	*	**	**	*	*	
$N(1880)$	$1/2^+$	1860±25	270±30	***	**	*	**	*	**	**	**	**		**			**	
$N(1895)$	$1/2^-$	1905±15	185±25	****	****	*	*	****	**	**	**	*	****	*	*			
$N(1900)$	$3/2^+$	1925±25	270±30	****	****	**	**	*	**	**	**		**	**	*		**	
$N(1965)$	$3/2^+$	1935±60	640±100	*	*	*					*							
$N(1990)$	$7/2^+$	2005±25	270±30	**	**	**	***	*	*	**	**							
$N(2000)$	$5/2^+$	1990±40	480±60	**	**	*	**	*	*	*	**		*	**		*		**
$N(2040)$	$3/2^+$	2040±25	230^{+15}_{-40}	*		*												
$N(2060)$	$5/2^-$	2020±30	410±50	***	***	**	*	*	*	*	**	*	*	*	*	*		*
$N(2100)$	$1/2^+$	2055±25	430±60	***	**	***	**	*	*		**	*	**	***				
$N(2120)$	$3/2^-$	2130±40	350±35	***	***	**	***		**	*	***	*	*	*	*	*		
$N(2190)$	$7/2^-$	2130±35	370±35	****	****	****	****	*	**	*	*	*		**				
$N(2220)$	$9/2^+$	2165±30	440±40	****	**	****	*	*	*	*	*							
$N(2250)$	$9/2^-$	2220±40	470±40	****	**	****	*	*	*	*	*							
$N(2300)$	$1/2^+$	2300^{+116}_{-30}	340^{+116}_{-65}	*		*												
$N(2570)$	$5/2^-$	2570^{+39}_{-14}	250^{+70}_{-32}	*		*												

Table 1.3. Overview of the known nucleon resonances with their spin and parity J^P, adapted from [80]. Masses and total widths are given in MeV. The right columns show the studied decay channels and the stars indicate how certain the evidence is, with four stars indicating that existence is certain. Black values come from the Review of Particle Physics (RPP) of 2004, while blue values come from the RPP 2022 [66]. Newer results are indicated in red. Yellow resonances have only been observed in J/ψ or $\psi(2S)$ decays [93], although it appears that $N(2040)$ has also been observed in πN scattering [94].

($I = \frac{3}{2}$). J/ψ decays therefore provide a clean isospin filter that isolates N^* resonances without contamination from Δ's.

So far, the BESIII collaboration has performed only a few dedicated studies of nucleon excitations. The main challenge has been to account for interference effects between different subsystems in multi-body decays involving baryons. Recently, however, correct solutions for this "spin alignment" problem (Section 3.3) have been developed and applied to studies at BESIII and LHCb. In this study, we take first steps towards applying these solutions to the decay $J/\psi \to \bar{p}\,(N^* \to \Sigma^+ K_S^0)$. In addition, we have prepared the computational techniques to couple this channel to $J/\psi \to \bar{p}\,(N^* \to p\eta)$ and other future selections, so that broad resonances can be studied properly in this higher-mass region where simple Breit–Wigner parametrisations are insufficient. Eventually, this paves the way to combining these results with the extensive knowledge from scattering experiments, moving closer to a unified description of the nucleon excitation spectrum.

2 Scattering theory

The ultimate goal in particle physics is to understand how particles interact. Generally speaking, particle properties can only be studied by measuring cross sections in particle accelerator experiments, such as colliders or fixed-target experiments. We therefore need to relate the cross sections that are measured experimentally to a theory that describes the underlying particle reactions. In this chapter, we will see how **scattering theory** is the bridge that connects theory and experiment by giving us a means to formulate observable transition amplitudes.

2.1. The scattering matrix

Any particle reaction in an experiment can be reduced to a scattering process consisting of three stages [95]:

1. approach of the incoming particles (**initial state**),
2. a process of interaction, during which the incoming particles may scatter, temporarily form intermediate states, or transform into new sets of outgoing particles,
3. and the scattered or formed particles moving away (**final state**).

When the interaction stage (2.) occurs within a relatively short time, both the initial and final states can be treated as collections of free, non-interacting particles that are unaffected by the interaction potential. This is a reasonable assumption, because the interaction region is almost point-like in comparison to the detector size [95]. The non-interacting incoming and outgoing states may therefore be approximated as asymptotic states. The challenge then is to relate the incoming asymptotes to the outgoing ones by describing the interaction process. Importantly, whether describing a classical or a non-relativistic interaction potential, the scattering process is fully determined once the incoming and outgoing asymptotic states are specified. This requirement is known as the **asymptotic condition**.

In the classical picture, the scattering process can be visualised as in Figure 2.1. Far from the interaction region, the trajectory of the particle approaches a well-defined asymptote, which can be characterised by its incoming or outgoing momentum. Within the interaction region, the path of the particle is fully determined by the interaction potential. However, for most practical purposes, it is unnecessary to determine the trajectory to learn about the potential. Instead, one can define a continuous mapping that relates each incoming momentum to a corresponding outgoing momentum. Importantly, this mapping is expressed in terms of measurable momenta, offering insight into the underlying potential and, by extension, the Lagrangian that governs the system.

In the quantum-mechanical case, the incoming and outgoing states are described by wave functions. Figure 2.2 is an artistic representation of that: an incoming plane wave travels through

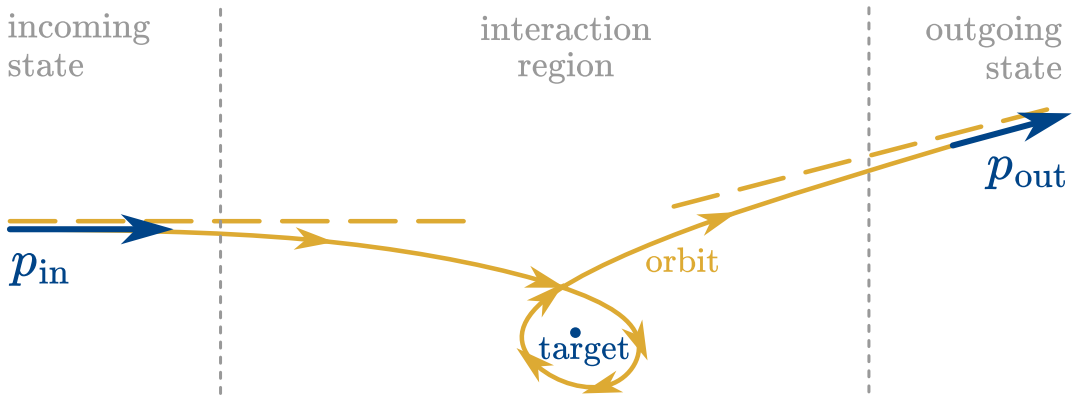


Figure 2.1. Sketch of the asymptotic condition in a classical scattering process. The connected continuum of incoming and outgoing momenta can be used to understand the interaction potential. Inspired by [95, Fig. 2.1].

an interaction region where an approximately localised potential scatters it into a superposition of outgoing waves. In quantum mechanics, the interaction process is described in terms of an operator rather than a classical orbit. Wheeler first introduced the concept of a "collision matrix" [96], which Heisenberg formalised in the form of an **S**-matrix operator [97] (not to be confused with the spin operator). By abandoning the notion that the scattering process is to be described as a positional wave function, the scattering process is translated to a model of exact momentum states that can be related through operators in the Heisenberg picture. This better matches the observables that we measure in experiment [98; 99], where the experimentalist essentially prepares initial state wave packets in momentum space and measures the outgoing state as a distribution of four-momenta [100].

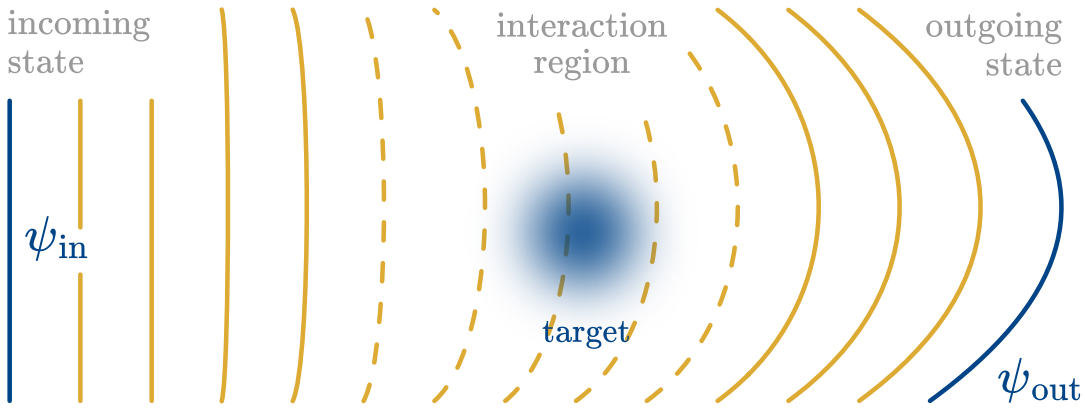


Figure 2.2. Scattering process of quantum mechanical wave functions. Incoming state ψ_{in} and outgoing state ψ_{out} are comparable to the asymptotes in Figure 2.1, but are defined as a superposition of quantum states in a specific basis of observables.

The idea of a scattering operator is quite general and can be applied to any scattering process,

including particle decays, even if multiple decay channels are involved [101]. Consider such an arbitrary incoming state $|\psi_{\text{in}}\rangle$. This can be any configuration of particles, such as e^-e^+ about to collide. We want to find a description for the resulting outgoing state $|\psi_{\text{out}}\rangle$ that can be a collection of decay products. Here, we introduce an **S-matrix** that maps the incoming state to the outgoing state by

$$|\psi_{\text{out}}\rangle \;=\; \mathbf{S}\,|\psi_{\text{in}}\rangle\;. \tag{2.1}$$

This expression encodes the entirety of the interaction. Although the microscopic details of the dynamics may be complex or even unknown, the **S**-matrix summarises their net effect on the asymptotic particle states, which are accessible by experiments.

The **S**-matrix is an extremely general operator and has to be further specified depending on the particle reaction that is being studied. For instance, the dimension of the **S**-matrix represents the number of possible states of the initial and final state, which makes the **S**-matrix suitable for coupled channel studies. The **S**-matrix and the incoming and outgoing states also have to be further parametrised in terms of variables and quantum numbers that are of relevance to the experiment, such as collision energy, four-momenta, spin, parity, et cetera.

In practice, we separate the *trivial* part of the scattering process, where the incoming state passes through unaffected by the interaction potential, from the *non-trivial* part that contains the effects of the interaction. This leads to the decomposition

$$\mathbf{S} \;=\; \mathbf{1} + i\mathbf{T}, \tag{2.2}$$

where $\mathbf{1}$ is the identity operator (corresponding to no scattering) and $\mathbf{T}$ is the **transition operator**. In some literature, the **T**-matrix is multiplied by a factor two, or a minus sign is used.

To relate these expressions back to experimental observables, we express the incoming and outgoing states in terms of observable basis states $|p_1, \dots, p_m; \alpha\rangle$ and $|p'_1, \dots, p'_n; \beta\rangle$, where $p_i^{(\prime)}$ are the observable four-momenta of the particles, and α, β label the set of discrete quantum numbers (such as spin, isospin, parity, and rest masses) that characterise the initial- and final-state channels. This leads to the **transition matrix elements** $\mathcal{M}_{\beta\alpha}$ describing the process $\alpha \to \beta$,

$$\begin{aligned}&\langle p'_1, \dots, p'_n; \beta \,|\, i\mathbf{T} \,|\, p_1, \dots, p_m; \alpha\rangle \\ &\quad=\; i(2\pi)^4\, \delta^{(4)}\!\left(p_\beta - p_\alpha\right)\, \mathcal{M}_{\beta\alpha}\left(p_1, \dots; p'_1, \dots\right)\,,\end{aligned} \tag{2.3}$$

where the $\delta^{(4)}$-function ensures conservation of incoming and outgoing four-momenta, $p_\alpha = p_\beta$, with $p_\alpha = \sum_{i=1}^{m} p_i$ and $p_\beta = \sum_{j=1}^{n} p'_j$. The quantity $\mathcal{M}_{\beta\alpha}$ is called a "matrix element" because it represents a component of the transition operator $\mathbf{T}$: it is the (discrete) projection into the transition $\alpha \to \beta$ as a (continuous) function of their asymptotic incoming and outgoing four-momenta $p_i^{(\prime)}$.

While the full **S**-matrix is a unitary operator acting on the entire Hilbert space of asymptotic momentum states [100], $\mathcal{M}_{\beta\alpha}$ captures the physical content of a particular transition and is the quantity that enters into the calculation of cross sections, decay rates, and other observable quantities. It is a complex-valued amplitude function that encodes the likelihood of the process, with its modulus squared appearing in measurable rates. For instance, for a scattering process with n final-state particles, the **differential cross section** is given by

$$\mathrm{d}\sigma_{\beta\alpha} \;=\; \frac{1}{F_\alpha}\left|\mathcal{M}_{\beta\alpha}\right|^2 \mathrm{d}\Phi_n\,, \tag{2.4}$$

where F_α is a flux factor determined by the initial state α and $\mathrm{d}\Phi_n$ is the Lorentz-invariant phase space volume element [66, §49]

$$\mathrm{d}\Phi_n \;=\; (2\pi)^4\,\delta^{(4)}\big(p_\alpha - p_\beta\big)\prod_{i=1}^{n}\frac{\mathrm{d}^3 p_i'}{(2\pi)^3\,2E_i'} \tag{2.5}$$

for the n-body final state. In decay processes, the cross section is often denoted with $\mathrm{d}\Gamma$ rather than $\mathrm{d}\sigma$ and is referred to as the partial or differential **decay rate**. The differential cross section quantifies the probability density for the transition from the initial state α to the final state β, as a function of the kinematic degrees of freedom in the final state and of the configuration of the initial state.

The matrix elements $\mathcal{M}_{\beta\alpha}$ denote the momentum projection of the transition operator $\mathbf{T}$ and are, as such, continuous functions of the incoming and outgoing momenta. In practice, we prefer to parametrise transition matrix elements as a function of other kinematic variables that are more relevant to the reaction process. The implicit convention in literature is to call such functions **transition amplitudes** and we follow the conventional notation to denote these functions with A or a. Amplitudes are like the *response functions* that tell how a system responds to external perturbations [102, p. 274], and the fact that they are directly related to observable cross sections makes them a powerful experimental tool for extracting information about the underlying dynamics of particle interactions.

2.2. S-matrix constraints

In theory, it is impossible to derive proper parametrisations of scattering amplitudes directly from the full QCD Lagrangian. The non-perturbative nature of the strong coupling at low energies makes exact solutions intractable. We can, however, derive a few basic constraints for the $\mathbf{S}$-matrix from fundamental principles of quantum mechanics and special relativity. These constraints are universal and model-independent: they must be satisfied by any theory of scattering, including QCD.

From this perspective, the $\mathbf{S}$-matrix provides an alternative but complementary framework to QCD – even though it historically preceded the concept of QCD itself [98, §7.1]. Rather than describing the microscopic dynamics of quarks and gluons, it focuses on observable relations between asymptotic states. The $\mathbf{S}$-matrix formalism enables amplitude models that enforce general physical principles while remaining agnostic about underlying dynamics, and provides a bridge between QCD predictions and experimental observables in the hadronic regime [103]. The most commonly invoked constraints are Lorentz invariance, unitarity of the transition amplitudes, and crossing symmetry.

Lorentz invariance

Given that we describe scattering entirely in terms of the momenta of incoming and outgoing particles, we must ensure that this description remains consistent under any change of inertial frame, so that we can relate it to the laboratory frame in which we perform our measurements. This requirement of **relativistic invariance** reflects the symmetry of spacetime as described by special relativity. Any theoretical description of scattering must respect the fact that the outcome of a physical process should not depend on the inertial frame in which it is described.

In relativistic quantum mechanics, this requirement is implemented through the **Poincaré group**, which comprises both spacetime translations and Lorentz transformations (rotations and

boosts). The **S**-matrix must commute with the generators of this symmetry group to ensure that they transform appropriately under changes of frame. In particular, Lorentz invariance implies that the **S**-matrix must be invariant under Lorentz transformations. If we define **U** as the unitary transformation that maps a Lorentz transformation $\boldsymbol{\Lambda}$ to the corresponding operator acting on the Hilbert space of quantum states, Lorentz invariance implies that

$$\mathbf{U}[\boldsymbol{\Lambda}]\,\mathbf{S}\,\mathbf{U}^{\dagger}[\boldsymbol{\Lambda}] \;=\; \mathbf{S}\,.$$

This constraint has important practical implications. First, it means that scattering amplitudes can only be a function of Lorentz-invariant combinations of the asymptotic four-momenta. Second, Lorentz invariance constrains how spin degrees of freedom must be treated. In a non-relativistic regime, spins can be added using standard angular momentum algebra, but the transformation of spin states under boosts becomes non-trivial (Chapter 3).

Unitarity

The most natural constraint is that the transition preserves probability, that is, 'what comes in, comes out'. In the notation of Equation (2.1), this means

$$\begin{aligned}\langle\psi_{\text{in}}\,|\,\psi_{\text{in}}\rangle \;&=\; \langle\psi_{\text{out}}\,|\,\psi_{\text{out}}\rangle \\ &=\; \langle\psi_{\text{in}}\,\big|\,\mathbf{S}^{\dagger}\mathbf{S}\,\big|\,\psi_{\text{in}}\rangle\;,\end{aligned}$$

which gives us the **unitarity condition** that ensures that the total probability for any process remains normalised,

$$\mathbf{S}^{\dagger}\mathbf{S} \;=\; \mathbf{S}\mathbf{S}^{\dagger} \;=\; \mathbf{1}\,,$$

with $\mathbf{1}$ the identity matrix. Inserting Equation (2.2) gives us the unitarity condition for the **T**-matrix,

$$\mathbf{T} - \mathbf{T}^{\dagger} \;=\; i\mathbf{T}^{\dagger}\mathbf{T}\,. \tag{2.6}$$

This equation shows that the imaginary part of the transition amplitude is constrained by its own modulus. Physically, it implies that the scattering process may involve not just direct deflection, but also the temporary formation of intermediate states such as resonances (**elastic scattering**), or transitions into entirely different final states (**inelastic scattering**) that are not accounted for in the model. In elastic scattering, the imaginary part of the amplitude reflects the finite lifetime of the resonant state, while in inelastic scattering, it indicates a genuine loss of flux from the initial channel into other open channels. Both effects are captured by the complex structure of the **T**-matrix, and its imaginary part plays a central role in encoding how probability is redistributed across scattering channels. In Section 2.4, we will see how this leads to observable phase shifts in the scattering amplitude.

Analyticity

In any physical process, causes must precede their effects. In scattering theory, causality requires that the wave function of a particle cannot "respond" before the interaction has taken place. This causal requirement leads to a mathematical condition: the transition matrix elements $\mathcal{M}_{\beta\alpha}$ must be **analytic** in energy and momentum, meaning that the energy-momentum domain can be

continued into the complex plane and that it varies in a way that is smooth, except at well-defined singularities associated with physical thresholds or states.

The connection between causality and analyticity becomes more transparent when we recognise that the **S**-matrix relates asymptotic momentum states in a way that is analogous to how a Fourier transform relates position and momentum space. Just as a time-localised function has a smooth (analytic) Fourier transform in frequency space, a localised interaction in spacetime leads to an analytic amplitude function in energy space [104; 102, p. 390]. This analyticity implies that the amplitude as a function of energy can be extended to a complex plane, where singularities such as poles appear in specific regions. These poles correspond to physical phenomena like bound states or resonances. While bound states appear as poles on the real axis, unstable states (resonances) correspond to poles with a negative imaginary part. Causality ensures this sign: it guarantees that the amplitude decays as time increases. A pole in the upper half-plane (positive imaginary part) would imply that the amplitude grows with time, violating the principle that effects cannot precede causes.

Analyticity gives the scattering amplitude an internal structure that causes it to "feed back" into itself – its absorptive component is determined by how likely it is to scatter into other channels, which is itself encoded in the amplitude. This enables powerful tools such as **dispersion relations**, which relate the value of an amplitude at one energy to its values elsewhere [105]. Once we know the amplitude in one location, we can analytically continue it through the complex plane into other regions of interest. Since the interaction potential is itself governed by the underlying fundamental forces, these relations allow us to extract information about the dynamics from measured cross sections, and compare them to those predicted by the formulated amplitude model [106].

Crossing symmetry

Scattering processes can be visualised with Feynman diagrams, which represent terms in the perturbative expansion of the **S**-matrix. A striking feature is that a single Feynman diagram can often describe multiple processes. For instance, the diagram representing a 2-to-2 scattering process $A+B \to C+D$ can, under certain momentum substitutions, also describe $A+\bar{C} \to \bar{B}+D$. This observation reflects a property of QFT known as **crossing symmetry**. Once this additional constraint is combined with the unitarity and analyticity conditions, the transition amplitudes of seemingly unrelated reactions become connected representations of a single analytic object that is a manifestation of the underlying field theory [102, p. 274; 107; 108].

Crossing symmetry was first described by Mandelstam [109], in a paper that also introduced the **Mandelstam variables**, or **invariants**, which provide a Lorentz-invariant description of two-body scattering kinematics. For a process $A + B \to C + D$, they are defined as

$$\begin{aligned} s &= (p_A + p_B)^2 = (p_C + p_D)^2 \\ t &= (p_A - p_C)^2 = (p_D - p_B)^2 \\ u &= (p_A - p_D)^2 = (p_C - p_B)^2 , \end{aligned} \tag{2.7}$$

where p_A, p_B are the four-momenta of the incoming particles and p_C, p_D those of the outgoing particles. Here, s combines the incoming momenta as a *sum*, giving the squared center-of-mass energy, while t and u involve momentum *differences* and measure the momentum transfers between initial and final states. The difference in structure indicates two distinct reaction types: s characterises a "direct" reaction, in which the two incoming momenta combine, whereas t and

u characterise "exchange" reactions that pair an incoming with an outgoing leg and describe momentum exchange (see Figure 2.3).

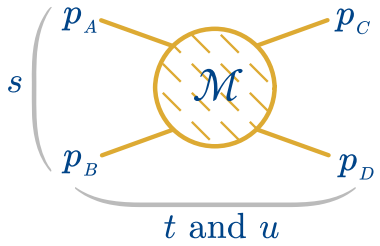


Figure 2.3. The two reaction types that can be distinguished in a 2-to-2 scattering process.

Each of these variables is often associated with a distinct channel topology or Feynman diagram (see Figure 2.4), which complements their broader role as analytic variables of the full 2-to-2 amplitude. The s**-channel** corresponds to the configuration in which the two incoming particles are combined, with s equal to their squared invariant mass. The t**-channel** corresponds to the configuration in which an incoming particle is paired with one of the outgoing particles, characterised by the squared momentum transfer t. The u**-channel** corresponds to the alternative pairing, where each incoming particle is combined with the other outgoing particle, giving the squared momentum transfer u. By crossing symmetry, an incoming particle can be reinterpreted as an outgoing antiparticle, which reverses the sign of its momentum.

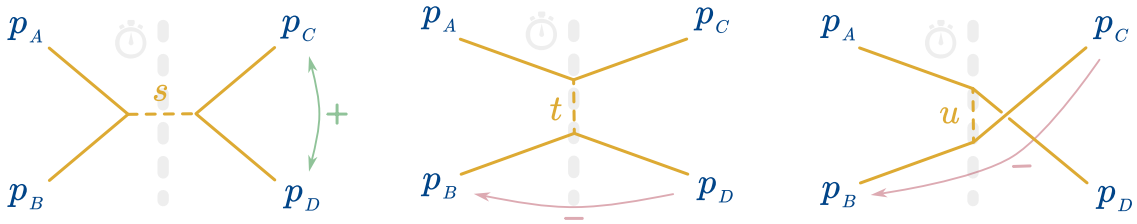


Figure 2.4. Channel topologies that are often associated with the three Mandelstam variables s, t, and u in a 2-to-2 scattering process in terms of the four-momenta of the incoming particles A and B and outgoing particles C and D. Momenta on the same side of the diagram contribute with a plus sign, while those on opposite sides contribute with a minus sign.

This perspective also clarifies how Mandelstam invariants appear in three-body decays. If a three-body final state is viewed as a sequence of two-body subprocesses, the same diagrammatic topologies reappear when deforming the scattering diagrams of Figure 2.4 into the decay diagrams of Figure 2.5. In this case, particle B is no longer incoming but part of the final state, so its momentum enters with the opposite sign. The resulting invariants are just the squared masses of two-body subsystems, which play the same role in Dalitz-plot analyses of decays as the Mandelstam variables do in scattering.

Because of four-momentum conservation, the Mandelstam variables are not independent but satisfy

$$\begin{aligned} s+t+u &= p_A^2 + p_B^2 + p_C^2 + p_D^2 \\ &= m_A^2 + m_B^2 + m_C^2 + m_D^2 \,, \end{aligned} \tag{2.8}$$

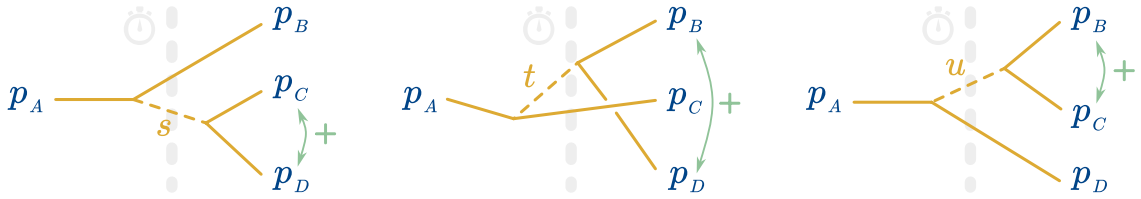


Figure 2.5. Mandelstam variables defined in the two-body subsystems of a three-body decay, obtained from the scattering topologies of Figure 2.4 by reinterpreting incoming particle B as part of the final state. In this decay picture, all momenta that enter a given invariant lie causally on the same side of the process and therefore appear with a plus sign.

A 2-to-2 scattering process can therefore be described in terms of only two independent invariants. This already hints that the processes sketched in Figure 2.4 and Figure 2.5 are not isolated but intrinsically connected.

These connections are visualised in Figure 2.6, which plots the Mandelstam variables over a real-valued, two-dimensional **Mandelstam plane**. The plot uses a barycentric coordinate system, with $x = \frac{s-t}{(s+t+u)\sqrt{2}}$ and $y = \frac{s+t-2u}{(s+t+u)\sqrt{6}}$, to emphasise that the choice of s, t, and u is arbitrary and that the three variables are symmetrically related to each other through Equation (2.8). The yellow regions mark the values kinematically allowed for given particle masses m_A, m_B, m_C, m_D, that is, the regions where experimental measurements can be made. They are determined by the condition that the Kibble function [110]

$$\phi(s,t,u) \;=\; \lambda(\lambda(s,m_B^2,m_A^2),\, \lambda(t,m_C^2,m_A^2),\, \lambda(u,m_D^2,m_A^2)) \tag{2.9}$$

is negative [102, pp. 175–177]. Here, λ denotes the Källén function [111, pp. 11–12], a symmetric polynomial in three variables, defined as

$$\lambda(x,y,z) \;=\; x^2+y^2+z^2-2xy-2yz-2zx\,. \tag{2.10}$$

Now, although these kinematically allowed regions are disjoint, and their Feynman diagrams correspond to distinct physical reactions, crossing symmetry tells us that they are analytically connected. Each scattering or decay channel corresponds to a different real slice of the same complex analytic function $A(s,t,u)$. What appear experimentally as separate processes are therefore a mere set of boundary values of one underlying analytic structure ("Mandelstam Hypothesis" [109; 112; 113, §III; 102, p. 274]).

Figure 2.6 also shows that, unlike in scattering, the three topologies of a three-body decay all occupy the same physical region. As we will see later, the amplitude can be expressed in terms of leading-order two-body subsystems, which are realised simultaneously in the Dalitz plot. This decomposition simplifies the analysis, but only approximates the full crossing-symmetric amplitude, since truncating the two-body partial-wave expansion (Section 2.3) inevitably violates exact crossing symmetry.

By modelling three-body decays as a sequence of two-body decays (Figure 2.5), the transition amplitude can be split into separate amplitudes that are parametrised as a function of a Mandelstam variable and a corresponding scattering angle θ (Chapter 3), giving us

$$A_{\text{total}}(s,t,u) \;=\; A^{(s)}(s,\theta_s) + A^{(t)}(t,\theta_t) + A^{(u)}(u,\theta_u)\,, \tag{2.11}$$

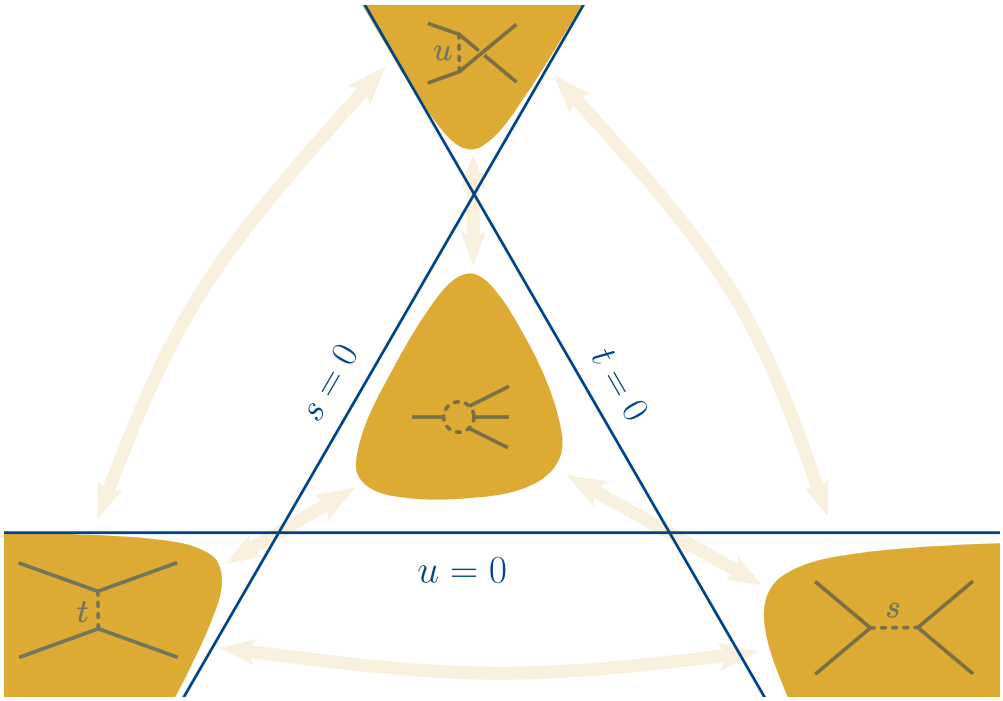


Figure 2.6. Barycentric visualisation of the kinematically allowed regions (shaded yellow) in the Mandelstam plane. The arrows indicate that the transition amplitudes for each reaction are connected through analytic continuation in the complexified plane. In this example, the regions are asymmetric, because each of the rest masses are all chosen differently.

with $A^{(s)}, A^{(t)}, A^{(u)}$ the amplitudes of the individual subsystems and $\theta_s, \theta_t, \theta_u$ the corresponding scattering angles. Since θ can itself be written in terms of another Mandelstam variable (Equation (2.26)), the angular part acquires a dynamical component that is analytically connected to the other channels.

A fully crossing-symmetric amplitude $A(s,t,u)$ that is analytic in the three Mandelstam variables simultaneously is in principle the ideal objective, but constructing such an object is extremely challenging [114]. The Khuri–Treiman formalism [115] starts from Equation (2.11) and refines the partial amplitudes through dispersion-relation corrections constrained by analyticity and two-body scattering data. The method has no fundamental restriction to non-zero spin, but its extension to a spinful final state is technically demanding. Regge theory is also relevant to the broader programme of constructing crossing-symmetric amplitudes, as it analytically continues the discrete angular momenta in the partial-wave expansion to complex values [116].

2.3. Partial-wave expansion

In Section 1.1, we saw that many of the quantum numbers used to classify hadrons – such as spin, parity, and total angular momentum – stem from the symmetries of space and time. For example, spin and orbital angular momentum arise from rotational invariance, while parity is associated with spatial inversion symmetry. These symmetry-related quantum numbers remain relevant in scattering theory, where they constrain the allowed transitions between incoming and outgoing states.

This is the key idea: if the system is rotationally invariant, then the total angular momentum must be conserved. Each amplitude component with a specific total angular momentum therefore scatters independently. The case of two-particle scattering $A + B \to C + D$ again makes this particularly clear. The sum of the momenta of the incoming particles defines a preferred direction, typically taken to be the z axis. Once we boost into the center-of-mass (CM) frame, and consider a fixed total energy s (Equation (2.7)), the geometry of the scattering process can be captured purely in terms of two angles, the **scattering angle** θ (angle between p'_C and z axis, see Figure 2.7) and the **azimuthal angle** ϕ (angle between the $p'_A p'_B$ plane and $p'_C p'_D$ plane). Indeed, once s is fixed, only rotational degrees of freedom remain, so any structure in the amplitude at fixed energy must reflect how the interaction redistributes angular momentum across different scattering angles.

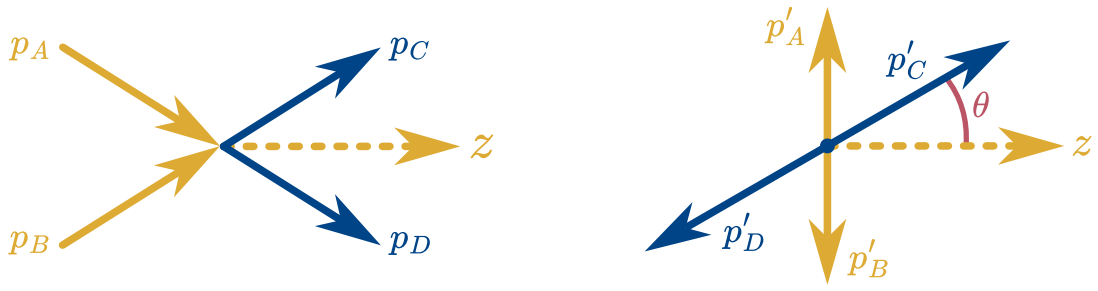


Figure 2.7. The sum of the incoming four-momenta can be used to define a z axis in the laboratory frame (left) and to boost into the CM frame (right). For a given total energy s, the four-momenta in the CM frame define the scattering angle θ (shown in the figure) and the azimuthal angle ϕ (not shown).

Similar to how a sound wave can be decomposed into pure tones of different frequencies, the amplitude $A(s,t)$ can be decomposed into independent (orthogonal) functions over the scattering angles. The result is called a **partial-wave expansion** [102, §4.4; 95, §6–c]. When the scattered particles have no intrinsic spin, these functions are given by the **Legendre polynomials** $P_\ell(\cos\theta)$ that depend only on the scattering angle θ. The amplitude at fixed total energy s can then be written as a sum over the orbital angular momenta ℓ,

$$A(s,\theta) \;=\; \sum_{\ell=0}^{\infty}(2\ell+1)\,P_\ell(\cos\theta)\,a^{(\ell)}(s)\,. \tag{2.12}$$

Here, $a^{(\ell)}(s)$ are the **partial-wave amplitudes** that encode the strength of the interaction in the orbital angular-momentum channel ℓ purely as a function of energy s. Overall, this is a **Fourier–Legendre expansion**, where P_ℓ plays the role of orthogonal basis functions.

In order to generalise the partial-wave expansion to particles of arbitrary spin, we need to introduce a few additional quantum numbers. If the incoming and outgoing particles $i=\{A,B,C,D\}$ have spin state $|J_i,\lambda_i\rangle$, with λ_i the spin projection of particle i along some quantisation axis, the amplitude can be projected into independent amplitudes $A_{\{\lambda\}}(s,\theta,\phi)$, with $\lambda=\{\lambda_A,\lambda_B;\lambda_C,\lambda_D\}$. These amplitudes can be expressed in terms of *total* angular momenta J (coupled basis of total spin and orbital angular momentum), so that the expansion coefficients become a matrix of functions of both θ and ϕ [117, p. 413],

$$\begin{aligned} A_{\{\lambda\}}(s,\theta,\phi) \;&=\; \sum_J (2J+1)\,D^{J*}_{\mu\mu'}(\phi,\theta,0)\,a^{(J)}_{\{\lambda\}}(s) \\ &=\; e^{i(\mu-\mu')\phi}\sum_J (2J+1)\,d^J_{\mu\mu'}(\theta)\,a^{(J)}_{\{\lambda\}}(s)\,, \end{aligned} \tag{2.13}$$

with $\mu=\lambda_A-\lambda_B$ and $\mu'=\lambda_C-\lambda_D$. The Wigner D- and d-functions are the matrix elements of rotation operators in the spin-J representation that generalise the Legendre polynomials for the spinless case. We return to the relation between the Wigner functions, the helicity basis, and the canonical spin basis in Chapter 3. If the scattered particles have no intrinsic spin, the total angular momentum J equals the orbital angular momentum ℓ, and the familiar expansion of Equation (2.12) is recovered from Equation (2.13), because for any angles α,β,γ,

$$D^\ell_{0,0}(\alpha,\beta,\gamma) \;=\; d^\ell_{0,0}(\beta) \;=\; P_\ell(\cos\beta)\,.$$

The key point is that we have factorised the transition amplitude $A(s,t)$ into (1) *angular* components that we know how to parametrise as a function of angles and (2) *dynamic* partial-wave amplitudes $a^{(J)}_{\{\lambda\}}(s)$ that are purely energy-dependent. By orthogonality, these dynamic partial-wave projections are given by the inverse of Equation (2.12) and Equation (2.13),

$$\begin{aligned} a^{(\ell)}(s) \;&=\; \frac{1}{2}\int_{-1}^{1}\mathrm{d}(\cos\theta)\,P_\ell(\cos\theta)\,A(s,\theta) \\ a^{(J)}_{\{\lambda\}}(s) \;&=\; \frac{1}{4\pi}\int_{-1}^{1}\mathrm{d}(\cos\theta)\int_0^{2\pi}\mathrm{d}\phi\,D^J_{\mu\mu'}(\phi,\theta,0)\,A_{\{\lambda\}}(s,\theta,\phi)\,. \end{aligned} \tag{2.14}$$

For convenience, we often drop the quantum numbers ℓ, J, and $\{\lambda\}$, so that $a(s)$ refers to a partial-wave amplitude. In Section 2.4, we will model this dynamic function $a(s)$ in order to extract information about the interaction potential from experimental data. Right now, however,

we can already further constrain this energy-dependent parametrisation purely on kinematic grounds.

As a first step, the factorisation into partial waves simplifies the unitarity condition on the transition matrix. When projected onto the angular momentum eigenbasis, Equation (2.6) becomes an energy-dependent condition that works *separately* for each J. By applying Equation (2.3) to Equation (2.6), we get the unitarity condition for the matrix elements,

$$\mathcal{M}_{\beta\alpha} - \mathcal{M}^*_{\alpha\beta} \;=\; i(2\pi)^4 \sum_{\gamma} \delta^{(4)}\big(p_\alpha - p_\beta\big)\; \mathcal{M}^*_{\gamma\beta}\, \mathcal{M}_{\gamma\alpha}\,, \tag{2.15}$$

where γ runs over all considered transition channels. This is known as the **generalised optical theorem** [118, p. 512; 100, p. 147]. Reformulating Equation (2.15) in terms of Lorentz-invariant variables s, t and applying the partial-wave projection from Equation (2.14), we get an energy-dependent unitarity condition for the partial-wave amplitudes,

$$\operatorname{Im} a_{\beta\alpha}(s) \;=\; \sum_{\gamma} a^*_{\gamma\beta}(s)\, \rho_\gamma(s)\, a_{\gamma\alpha}(s)\,, \tag{2.16}$$

or, in matrix form,

$$\operatorname{Im} \boldsymbol{a}(s) \;=\; \boldsymbol{a}^\dagger(s)\, \boldsymbol{\rho}(s)\, \boldsymbol{a}(s)\,. \tag{2.17}$$

In the single-channel case, this partial-wave unitarity condition becomes a more recognisable form of the optical theorem,

$$\operatorname{Im} a(s) \;=\; \rho(s)\; |a(s)|^2\,, \tag{2.18}$$

which relates the imaginary part of the partial-wave amplitude to its cross section. Here, we have introduced a Lorentz-invariant, normalised **phase space factor** $\rho(s)$. In general, this factor is defined as the integral of the phase space element of Equation (2.5) over the angular degrees of freedom at fixed total centre-of-mass energy s. In the special case of two-body scattering or decays, this integral has an analytic solution and can be expressed in terms of the **breakup momentum** $q(s)$ [66, §50, p.8],

$$\rho(s) \;=\; 4(2\pi)^5 \int \mathrm{d}\Phi_2 \;=\; \frac{2q(s)}{\sqrt{s}} \;=\; \frac{\sqrt{\lambda(s, m_C^2, m_D^2)}}{s}\,. \tag{2.19}$$

The factor $(2\pi)^4$ is often absorbed into Φ, leaving a conventional factor 8π.The precise form of Equation (2.18) also depends on the normalisation of the scattering amplitude: with the convention of Equation (2.4), the inverse amplitude satisfies $\operatorname{Im} a^{-1}(s) = \rho(s)$, while alternative normalisations shift factors such as 16π into the definition of $\rho(s)$.

The appearance of λ from Equation (2.10) reflects the same kinematic constraint $\phi(s,t,u) < 0$ of Equation (2.9) that defines the boundary of the physical region in Mandelstam space (see Figure 2.6). The phase space factor $\rho(s)$ thus plays the role of an *energy-dependent normalisation*, arising directly from unitarity and the kinematics of the two-body system.

As a reminder, when working with multiple channels, a subscript in the phase space factor or breakup momentum indicates the channel for which it is defined. For a two-body final state α with rest masses $m_{1,\alpha}, m_{2,\alpha}$, this means

$$\begin{aligned} \rho_\alpha(s) &= \frac{\sqrt{s-(m_{1,\alpha}-m_{2,\alpha})^2}\sqrt{s-(m_{1,\alpha}+m_{2,\alpha})^2}}{s} \\ q_\alpha(s) &= \frac{\sqrt{s-(m_{1,\alpha}-m_{2,\alpha})^2}\sqrt{s-(m_{1,\alpha}+m_{2,\alpha})^2}}{2\sqrt{s}}\,. \end{aligned} \tag{2.20}$$

This form with a product of square roots is often preferred over Equation (2.19), as it has a simpler analytic structure in the complex plane (one branch cut, see Section 2.4).

Unitarity also motivates a useful parametrisation of the partial-wave amplitudes in terms of a phase shift. We can rewrite Equation (2.18) to

$$|1 + i\rho(s)\,a(s)| \;=\; 1\,,$$

which confines $\rho(s)\,a(s)$ to a circle in the complex plane. Positions on that circle can be associated to angles, which naturally leads to parametrising the amplitude in terms of a **phase shift** $\delta(s)$ via

$$\begin{aligned} a(s) &= \frac{e^{2i\delta(s)}-1}{2i\,\rho(s)} \\ \Rightarrow\; e^{i\delta(s)}\sin\delta(s) &= \rho(s)\,a(s)\,. \end{aligned} \tag{2.21}$$

Physically, $\delta(s)$ represents the delay of the outgoing wave relative to free propagation. Its behaviour as a function of energy reveals features such as resonances and bound states – a rapid increase in $\delta(s)$ indicates elastic effects during the scattering process.

As a first step in a **partial-wave analysis** (PWA), one often applies the partial-wave expansion without committing to any detailed dynamical model of $a(s)$. In practice, the experimental data is divided into bins of the energy variable s. Within each bin, one assumes that s is effectively constant and that the corresponding partial-wave amplitude $a^{(\ell)}(s)$ can be approximated by a single complex parameter. The angular distribution observed in that bin is then fitted by adjusting these complex coefficients, one for each included partial wave. Each coefficient is thus treated as a free parameter of the fit, with no energy dependence imposed across bins.

The result is a collection of complex values $\{a^{(\ell)}(s_i)\}$ at the bin centers $\{s_i\}$. If these values vary smoothly from bin to bin, they can be interpolated to define a continuous amplitude function, from which phase shifts can be extracted. Such **model-independent** or energy-independent PWAs [119] already provide valuable insight into resonant behaviour, since characteristic phase motions across the energy range can already reveal the presence of resonant structures without assuming any specific parametrisation.

2.4. Resonance identification

When we visualise measured cross sections as a function of energy, we often observe sharp peaks or dips at specific energies. An example is shown in Figure 2.8, which displays the differential cross section for three backward-angle pion–nucleon scattering processes ($\pi^+p \to \pi^+p$, and $\pi^-p \to \pi^-p$, and $\pi^-p \to \pi^0n$). The curves represent fitted amplitude analyses that combine data from several pion–proton scattering experiments [119]. The observed structures reflect the underlying dynamics of the strong interaction. Scattering experiments are therefore like probing a bell, where the peaks correspond to the characteristic frequencies at which the bell resonates.

The width of these peaks indicates how long the bell rings before returning to equilibrium, which in the scattering case is related to the lifetime of the excited state. In hadron physics, such structures are interpreted as **resonances** – unstable states that exist for a finite time before decaying into other particles.

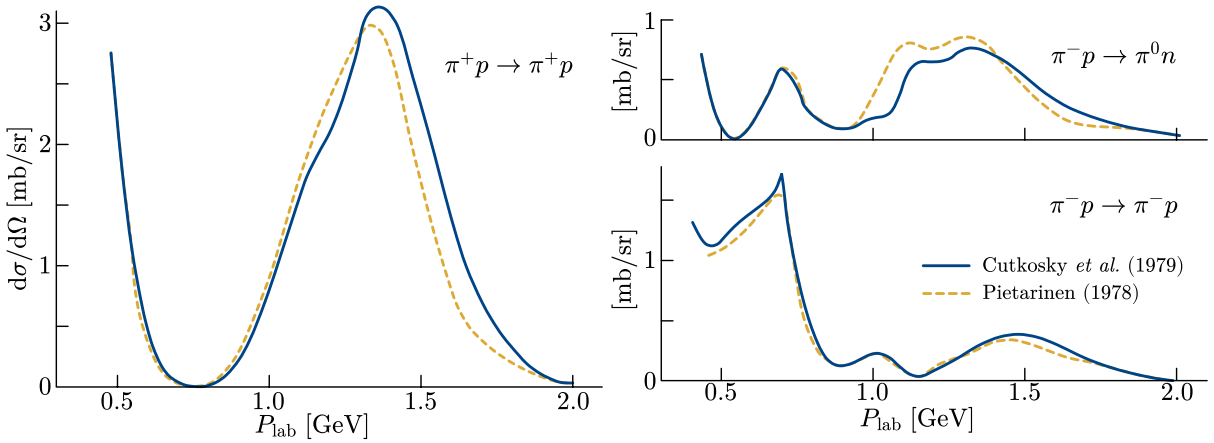


Figure 2.8. Differential cross sections $\mathrm{d}\sigma/\mathrm{d}\Omega$ for three different pion–nucleon scattering processes at the backward scattering angle ($\theta = 180°$). Shown are two fitted solutions based on amalgamated data from several pion–proton scattering experiments. Adapted from [119, Fig. 10].

Equation (2.4) allows us to relate the observable cross section distribution to the matrix elements $\mathcal{M}_{\beta\alpha}$ and Equation (2.13) helps us to further split up the cross section into partial-wave amplitudes $a^{(J)}(s)$ for each total angular momentum J. The bumps will now only appear in certain total angular momentum channels, which we can identify by looking at the phase shifts $\delta(s)$ of the partial-wave amplitudes. This suggests that the resonances are actual particles with an intrinsic spin that couples to that total angular momentum.

In certain processes, such as pion–nucleon scattering, one can go beyond spin and angular momentum and also extract the isospin of the resonances. The individual charge channels shown in Figure 2.8 are not themselves states of definite isospin, but fixed linear combinations of the $I = \frac{1}{2}$ and $I = \frac{3}{2}$ amplitudes. A partial-wave analysis of a single channel therefore cannot separate the isospin components. Only by fitting the elastic reactions π^+p and π^-p together with the charge-exchange channel $\pi^-p \to \pi^0n$ can one extract information about the isospin amplitudes. This coupled-channel treatment makes it possible to disentangle the isospin structure of the partial waves and to assign resonances their total isospin.

For πN scattering the situation is relatively simple, since the initial state can only be in an $I = \frac{1}{2}$ or $I = \frac{3}{2}$ configuration. The relation between the observable charge channels and the isospin amplitudes is

$$\begin{aligned} \pi^+p \to \pi^+p &: A_{3/2} \\ \pi^-p \to \pi^-p &: \tfrac{1}{3}A_{3/2} + \tfrac{2}{3}A_{1/2} \\ \pi^-p \to \pi^0n &: \tfrac{\sqrt{2}}{3}\left(A_{3/2} - A_{1/2}\right) , \end{aligned}$$

where $A_{1/2}$ and $A_{3/2}$ denote the partial-wave amplitudes with definite isospin. While the details of this isospin decomposition are beyond the scope of this work, it helps to understand what certain partial-wave labels mean and how these waves are extracted. For example, the P_{33}-wave

(Figure 2.9) denotes for a partial wave with orbital angular momentum $\ell = 1$, isospin $I = \frac{3}{2}$, and total angular momentum $J = \frac{3}{2}$, and how they are extracted from total differential cross sections. Similarly, the D_{15}-wave corresponds to $\ell = 2$, $I = \frac{1}{2}$, and $J = \frac{5}{2}$. The scheme follows spectroscopic letter notation $S, P, D, F, \ldots$ for orbital angular momentum $\ell = 0, 1, 2, 3, \ldots$.

Figure 2.9 shows the imaginary and real parts of the P_{33}-wave obtained from the pion–nucleon cross sections of Figure 2.8 [120]. The prominent peak around 1.23 GeV and the zero-crossing of the imaginary part reveals the $\Delta(1232)$ resonance with quantum numbers $I(J^P) = \frac{3}{2}\left(\frac{3}{2}^+\right)$. This is a striking example of how resonances emerge from the partial-wave amplitudes: once hidden in the energy spectrum, they stand out clearly in the complex amplitude. The remainder of this section introduces the techniques that make such extractions possible.

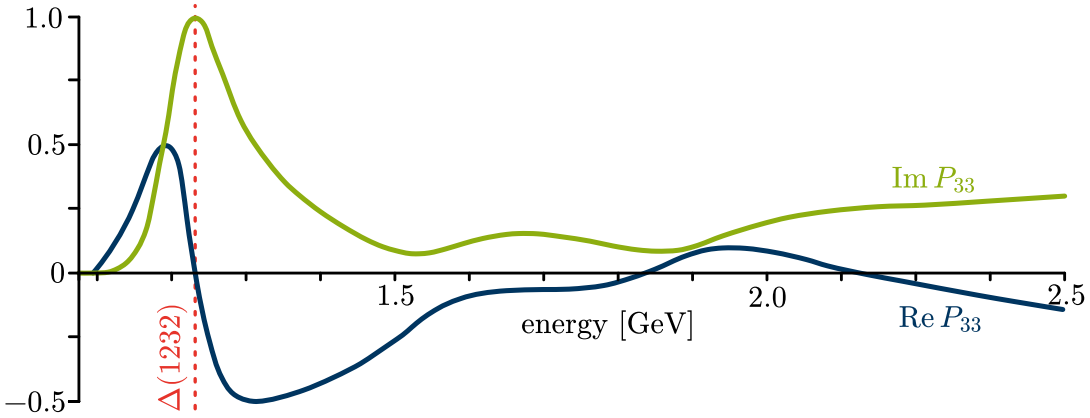


Figure 2.9. Real and imaginary part of the P_{33}-wave amplitude that was extracted from the pion–nucleon scattering cross section in Figure 2.8. The position of the peak of one of the three bumps is highlighted in red. Adapted from [120, Fig. 3].

Breit–Wigner parametrisation

To model the shape of a resonance, we need a function that captures both the peak structure and a width that is associated with the lifetime of the resonance. A simple but remarkably effective parametrisation is the **Breit–Wigner function** [121], which describes the relativistic scattering amplitude near a resonance as a function of energy s,

$$a(s) \;\cong\; \frac{1}{M_R^2 - iM_R\Gamma_R - s}\,, \tag{2.22}$$

where M_R is the mass of the resonance and Γ_R its total decay width. The numerator often includes a scaling factor or is normalised to the peak by setting it equal to $M_R\Gamma_R$, but these factors are left out here (indicated by $\cong$). This functional form mirrors the response of a driven, damped harmonic oscillator – a system that resonates when driven near its natural frequency but loses energy due to damping. Similarly, the Breit–Wigner function reflects how the system "rings" in response to a scattering event and then decays.

Figure 2.10 visualises the Breit–Wigner function through its modulus squared, real and imaginary parts, and the associated phase. The resonance appears most clearly in the left panel, where the modulus $|a_J|^2$ forms its characteristic peak at $s = M_R^2$ with a finite width controlled by the imaginary term $iM_R\Gamma_R$ in the denominator. The **Argand diagram** in the middle plots

$\operatorname{Im} a_J$ against $\operatorname{Re} a_J$ to show how an isolated Breit–Wigner traces out a circle, thereby preserving unitarity with inelasticity $\eta = 1$. The same diagram also highlights the connection between the phase shift $\delta(s)$ of Equation (2.21) and the modulus of the amplitude. The right panel shows that the corresponding phase shift $\delta(s)$ undergoes a rapid increase by approximately π across the resonance region, which indicates a relative time delay of the outgoing wave.

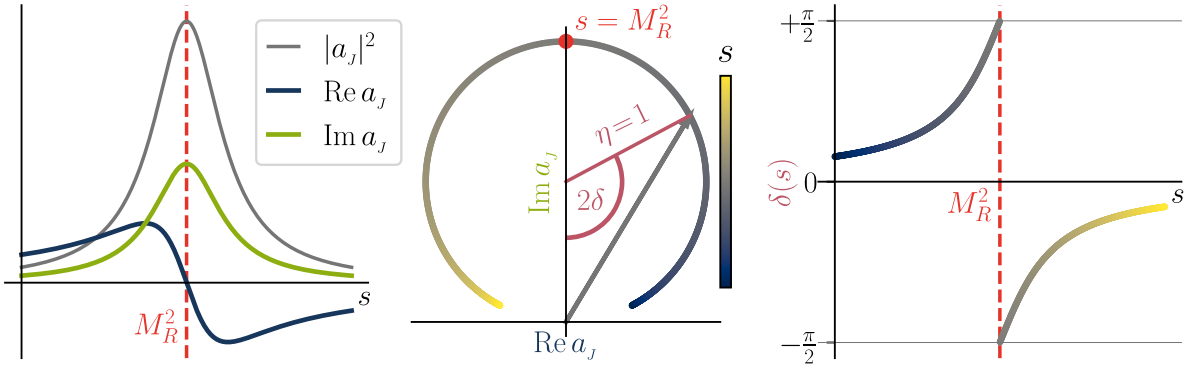


Figure 2.10. Visualisation of the real and imaginary parts, the modulus, and the phase of the relativistic Breit–Wigner function of Equation (2.22), with the position M_R of the resonance indicated in red.

The width Γ_R in Equation (2.22) is currently treated as a constant. However, since the width is related to decay probability per unit time, it should depend on the available phase space for the decay products. As the total energy s increases, the decay products carry more momentum and the phase space volume grows. This implies that the width should not be treated as a constant, but should instead vary with s. For a resonance decaying to two particles C and D, this energy dependence is therefore often encoded with an **energy-dependent width** by replacing Γ_R in Equation (2.22) with

$$\Gamma_\ell(s) \;=\; \Gamma_R \frac{\rho(s)}{\rho(M_R^2)} \frac{n_\ell^2\left(q^2(s)/q_0^2\right)}{n_\ell^2\left(q^2(M_R^2)/q_0^2\right)}\,, \tag{2.23}$$

with phase space factor ρ and breakup momentum q as defined in Equation (2.20) at total energy s or squared mass M_R^2, ℓ the orbital angular momentum between C and D, n_ℓ a factor that encodes angular-momentum suppression near threshold, and $1/q_0$ a scaling parameter that typically lies in the range $1 \sim 5\ \text{GeV}^{-1}$ [66, §50]. For brevity, we often shorten notation with $n_\ell(s) \equiv n_\ell(q^2(s)/q_0^2)$.

The factors n_ℓ are often called **centrifugal barrier factor** and they arise because partial waves with higher angular momentum are suppressed [122, §5–10] when the decay products have low relative momentum (more on that in Section 2.4). The barrier factor is usually parametrised with a (unitless, normalised) Blatt–Weisskopf formula via

$$\begin{aligned} n_\ell^2(x^2) \;&=\; \frac{\left|h_\ell^{(1)}(1)\right|^2}{x^2\left|h_\ell^{(1)}(x)\right|^2} \\ h_\ell^{(1)}(x) \;&=\; (-i)^{\ell+1}\,\frac{e^{ix}}{x}\sum_{k=0}^{\ell}\frac{(\ell+k)!}{(\ell-k)!\,k!}\left(\frac{i}{2x}\right)^k, \end{aligned} \tag{2.24}$$

where $h_\ell^{(1)}$ are the Hankel functions of the first kind, which can be written in this polynomial form if its argument x is real [123, pp. 626, 637]. This is the general form of the **Blatt–Weisskopf factors** that leads to the specific cases that are listed in most literature,

$$n_\ell^2(x^2) \;=\; \begin{cases} 1 & \text{for } \ell = 0 \\ \frac{2x^2}{x^2+1} & \text{for } \ell = 1 \\ \frac{13x^4}{x^4+3x^2+9} & \text{for } \ell = 2 \\ \cdots\ . \end{cases}$$

Figure 2.11 shows that a Breit–Wigner with energy-dependent width already captures several important decay-specific features. Near threshold, where $q(s) \to 0$, the width vanishes as the phase space closes, which introduces an asymmetry in the lineshape. The parametrisation is now also sensitive to the orbital momentum, which offers a handle to select the dominant partial wave through the damping factor. To examine whether this parametrisation respects Equation (2.18), the Argand diagram in Figure 2.11 plots the product $\rho_\ell(s)\, a_\ell(s)$, with $\rho_\ell(s) \equiv \rho(s)\, n_\ell^2(s)$, rather than the amplitude $a_\ell(s)$ itself [124]. This shows that the parametrisation with energy-dependent width still preserves unitarity for each angular momentum ℓ.

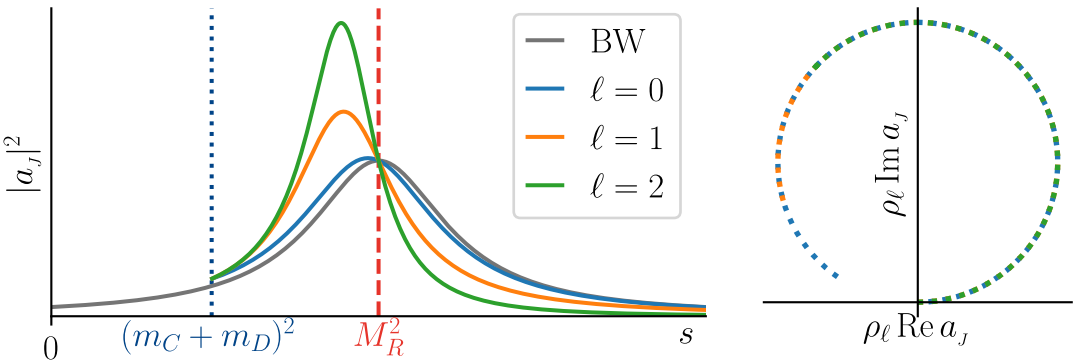


Figure 2.11. Breit–Wigner parametrisation with energy-dependent width $\Gamma_\ell(s)$ for a resonance $R \to CD$ in different orbital angular momenta ℓ. Left: squared amplitudes $|a_\ell|^2$ as a function of energy s. Right: Argand representation of $\rho_\ell(s)\, a_\ell(s)$.

Vertex parametrisation

The appearance of decay-specific parameters in the resonance parametrisation suggests a broader physical interpretation: the resonant amplitude can be viewed as the product of two interaction vertices and an intermediate propagator. This picture is already implicit in the Feynman diagram for the s-channel of a 2-to-2 scattering process shown in Figure 2.4: the intermediate particle R is first produced in the $AB \to R$ subprocess, and then decays via $R \to CD$. The Breit–Wigner function introduced in Equation (2.22) represents the **resonance propagator** (the denominator of the amplitude), but the **interaction vertices** remain to be modelled explicitly. This leads to a more general form of the amplitude,

$$a^{(J)}_{\beta\alpha}(s) \cong \frac{n_\beta(s)\, n_\alpha(s)}{M_R^2 - iM_R\Gamma_{\ell_\beta}(s) - s}\,. \tag{2.25}$$

The vertex functions n_α and n_β describe how the resonance couples to the initial and final state as a function of s. In analogy to the energy-dependent width, they are often modelled with barrier or damping factors like the Blatt–Weisskopf factor of Equation (2.24), with orbital angular momentum ℓ_α for AB and ℓ_β for CD (see left panel in Figure 2.12). Note that ℓ_α and ℓ_β can be different if the initial or final state contains particles with spin, because they couple differently to the total orbital momentum J of the partial-wave expansion. The right panel in Figure 2.12 shows the suppressive effect by barrier factors with $\ell_\alpha = 0, \ell_\beta = 1$ on a Breit–Wigner with energy-dependent width. We will see later why the barrier factor that appears in the energy-dependent width of Equation (2.23) comes from the outgoing vertex.

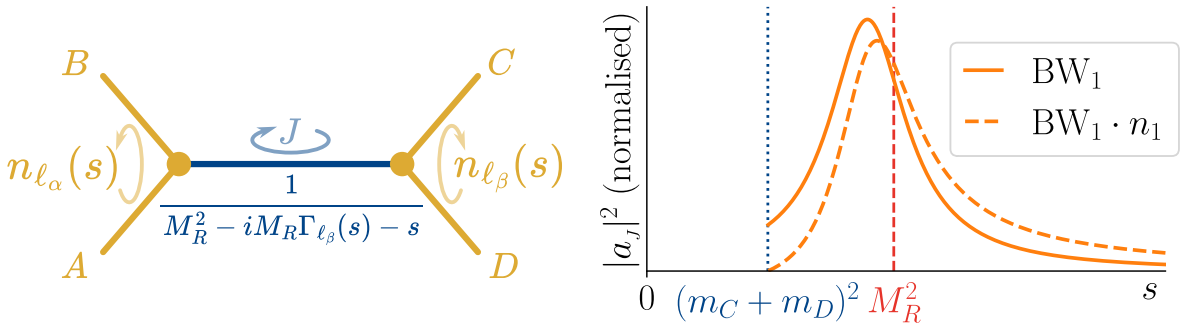


Figure 2.12. Effect of vertex barrier factors on a Breit–Wigner amplitude. Left: Resonant 2-to-2 scattering with orbital angular momenta ℓ_α and ℓ_β. Right: Normalised squared amplitude for $\ell_\alpha = 0$, $\ell_\beta = 1$, shown without (solid) and with (dashed) the production barrier factor. Vertical lines mark the CD threshold (blue) and the resonance mass M_R (red).

Confusingly, there are different conventions in literature for barrier factors. The reason is that these factors are related to angular functions like $P_\ell(\cos\theta)$ that appear in the partial-wave expansion Equation (2.12) and Equation (2.13). These factors depend on the scattering angle, which can be computed from t and s. Note that

$$\begin{aligned} t &= (p_A - p_C)^2 \\ &= m_A^2 + m_C^2 - 2E_A E_C + 2|\vec{p}_A||\vec{p}_C|\cos\theta \end{aligned}$$

with

$$\begin{aligned} |\vec{p}_A| &= \frac{\lambda^{1/2}(s, m_A^2, m_B^2)}{2\sqrt{s}} \qquad |\vec{p}_C| = \frac{\lambda^{1/2}(s, m_C^2, m_D^2)}{2\sqrt{s}} \\ E_A &= \frac{s + m_A^2 - m_B^2}{2\sqrt{s}} \qquad E_C = \frac{s + m_C^2 - m_D^2}{2\sqrt{s}} . \end{aligned}$$

Solving for $\cos\theta$, we get

$$\begin{aligned} \cos\theta &= \frac{t - m_A^2 - m_C^2 + 2E_A E_C}{2|\vec{p}_A||\vec{p}_C|} \\ &\cong \frac{\mathrm{Pol}_1(s,t)}{\lambda^{1/2}(s, m_A^2, m_B^2)\,\lambda^{1/2}(s, m_C^2, m_D^2)} , \end{aligned} \tag{2.26}$$

where the second line highlights that the numerator scales as a first-order polynomial of s, t and that the denominator depends on two Källén functions $\lambda^{1/2}(s, \dots)$. This form shows that $\cos\theta$ develops divergent behaviour as s approaches the threshold, because that causes $\lambda^{1/2}(s, \dots)$ to go to zero. This would lead to an unphysical divergence in each term $a_\ell(s)\, P_\ell(\cos\theta)$ in Equation (2.12), unless $a_\ell(s)$ compensates it. Rescaling $a_\ell(s)$ with the breakup momenta $q_\alpha^{\ell_\alpha}(s)$ and $q_\beta^{\ell_\beta}(s)$ of the initial and final state (Equation (2.20)) exponentiated with their associated orbital angular momenta does the trick.

To model higher energies correctly, an additional centrifugal factor has to be included as well, but this is a phenomenological choice that is not directly related to the corrective factor described before. The usual choice is the Blatt–Weisskopf barrier factor in *non-normalised* form, often written as $\mathcal{F}_\ell$. It is related to the normalised form of Equation (2.24) by [66, p. 12]

$$\mathcal{F}_\ell^2(x^2) \;=\; \frac{1}{x^{2(\ell+1)}\,\left|h_\ell^{(1)}(x)\right|^2} \;=\; \frac{n_\ell^2(x^2)}{x^{2\ell}\,\left|h_\ell^{(1)}(1)\right|^2}\,.$$

All in all, this separates the vertex factors into a *corrective* and a *phenomenological* component, which can be written as

$$n_\ell\left(q^2(s)\right) \;=\; q^\ell(s)\;\mathcal{F}_\ell\left(q^2(s)\right)\,.$$

The reactance matrix

Despite these improvements, the Breit–Wigner parametrisation remains limited in scope. In practice, resonances in the same partial wave rarely appear in isolation across the energy spectrum (Figure 2.9). Instead, they often overlap and interfere, especially when multiple resonant structures or open decay channels are present. Such interference effects cannot be accurately described by a simple sum of Breit–Wigner terms. This breakdown reflects the fact that the amplitude is no longer confined to a single channel: part of the flux is dissipated into other channels that are not accounted for.

As noted in Section 2.1, Heisenberg framed scattering as a kind of response problem: a system is probed by asymptotic incoming waves, and its internal structure governs the outgoing response. This perspective inspired many physicists in the 1940s and 50s to draw analogies between scattering amplitudes and the response of an electrical circuit, where the impedance characterises how the system reacts to an external driving force. Notably, Blatt and Weisskopf explicitly borrowed the concept of **reactance** from electrical engineering to formulate a more general description of scattering amplitudes [125, p. 530] – a framework now known as the ***K*-matrix formalism**.

The **reactance matrix** $\boldsymbol{K}$, or reaction matrix [126], emerges naturally when considering the unitarity condition Equation (2.18) for partial-wave amplitudes. This condition imposes a non-linear constraint on the amplitude, which is generally difficult to solve directly. However, if the amplitude is parametrised as

$$a(s) \;=\; \frac{K(s)}{1 - i\,\rho(s)\,K(s)}\,, \tag{2.27}$$

then Equation (2.18) is automatically satisfied above threshold (where $\rho(s)$ is real and positive), provided that $K(s)$ is itself a real-valued function. More generally, the $\boldsymbol{K}$-matrix arises from a

Cayley transform of the **S**-matrix. This mathematical transformation maps a unitary matrix (here **S**) to a Hermitian operator **X** via

$$\mathbf{X} \;=\; (\mathbf{S}-\mathbf{1})\,(\mathbf{S}+\mathbf{1})^{-1}\;. \tag{2.28}$$

Similarly as in Equation (2.18), we introduce an energy-dependent normalisation with the parametrisation

$$\boldsymbol{X}(s) = i\boldsymbol{\rho}(s)\,\boldsymbol{K}(s)\,,$$

where $\boldsymbol{\rho}(s)$ is a diagonal matrix of phase space factors $\rho_\alpha(s)$ for each channel α. Given that the elements in $\boldsymbol{\rho}(s)$ are real and positive for s above all thresholds, the $\boldsymbol{K}$-matrix has to be real-valued to make **X** Hermitian, and **S** unitary. This argument does not hold between or below thresholds (there is a **left-hand cut**), but this will be solved through analytic continuation later on. Using Equation (2.2), transition operator **T** and its corresponding partial-wave projection $\boldsymbol{a}(s)$ gets the same form as Equation (2.27), but in matrix form

$$\boldsymbol{a}(s) \;=\; \boldsymbol{K}(s)\;(\boldsymbol{I}-i\,\boldsymbol{\rho}(s)\,\boldsymbol{K}(s))^{-1}\;. \tag{2.29}$$

Just as reactance (the imaginary part of impedance) describes how a circuit temporarily stores and returns energy without dissipation, the $\boldsymbol{K}$-matrix captures the elastic component of scattering: energy may be temporarily trapped, for instance in a resonant state, but is not lost. Dissipative effects, such as the loss of flux into other channels, are reintroduced in a controlled way through the phase space factor, which supplies the imaginary part of the full amplitude.

Figure 2.13 shows the elements of the $\boldsymbol{K}$ matrix for two-channel nucleon scattering, πN and ηN. The $\boldsymbol{K}$-matrix describes the two-channel system as a whole by accounting for the flow between the two channels through it's off-diagonal elements.

Figure 2.13. The four elements of the $\boldsymbol{K}$-matrix for two-channel nucleon scattering, with final state πN (1) and ηN (2).

Since the $\boldsymbol{K}$-matrix plays the role of an energy-dependent kernel that captures the reactive core of the scattering process, any resonant peak structures in the amplitude must modelled through the (real-valued) elements of the $\boldsymbol{K}$-matrix. The $\boldsymbol{K}$-matrix is therefore usually parametrised as a spectral decomposition of **poles** via

$$K_{\beta\alpha}(s) \;=\; \sum_r \frac{g^r_\beta\, g^r_\alpha}{m_r^2 - s}\,, \tag{2.30}$$

where r is an index for each pole, m_r is **bare mass** of pole r, and g^r_α, g^r_β are **coupling constants** that describe how strongly pole r couples to initial state α and final state β [66, §50; 127; 128]. These poles do not correspond directly to physical particles, but represent internal excitation modes (bare states) of the multichannel system before unitarity and channel-specific thresholds dress them into observable amplitudes [129; 130]. In this sense, they play a role akin to the normal modes of an oscillator or the eigenfrequencies of a cavity: points at which the internal structure of the interaction resonates most strongly.

The partial-wave decomposition from Equation (2.13) still applies, because it is a decomposition of total angular momentum J, not of internal dynamics. However, similar to Figure 2.12, the partial-wave amplitude becomes suppressed by the centrifugal barrier factors $n_{\ell_\alpha}, n_{\ell_\beta}$ from Equation (2.24) due to the orbital angular momenta ℓ_α, ℓ_β of the incoming and outgoing vertex, respectively. Infusing Equation (2.29) and Equation (2.3) with these factors, we get a partial-wave amplitude that is "dressed" with a diagonal matrix $\boldsymbol{n}$ of vertex functions [66, §50, p.13],

$$\boldsymbol{a} \;=\; \boldsymbol{n}\,\boldsymbol{K}\,\left(\boldsymbol{I} - i\,\boldsymbol{\rho}\,\boldsymbol{n}^2\,\boldsymbol{K}\right)^{-1}\boldsymbol{n} \tag{2.31}$$

or explicitly, in terms of the matrix element functions,

$$a_{\beta\alpha}(s) \;=\; n_{\ell_\beta}(s)\sum_\gamma K_{\beta\gamma}(s)\left[\left(\boldsymbol{I} - i\,\boldsymbol{\rho}(s)\,\boldsymbol{n}(s)^2\,\boldsymbol{K}(s)\right)^{-1}\right]_{\gamma\alpha} n_{\ell_\alpha}(s)\,. \tag{2.32}$$

One can derive that the single-channel version of Equation (2.31) reduces to Equation (2.25) if there is only one pole with bare mass M_R and coupling $g_1^2 = M_R\Gamma$.

Figure 2.14 shows the resulting partial-wave amplitudes for the P-wave ($\ell_1 = \ell_2 = 1$) in πN and ηN nucleon scattering. In this example, the $\boldsymbol{K}$-matrix is modelled with one pole with a bare mass set to that of the Roper resonance $N(1440)$ [131]. There are four transition matrix elements, with the off-diagonal elements describing the flow of flux between the two channels (inelastic scattering). The amplitude a_{11} for elastic πN scattering exhibits a "cusp effect" at the threshold where the ηN channel opens up. All other elements lie above the πN threshold.

The figure shows the modulus squared, real part, and imaginary part of each amplitude. Counter to expectations, the real part does not show the characteristic shift around the bare mass position, like in Figure 2.10. This is because the couplings have been set to a high value, moving the pole position further away from the real axis.

The $\boldsymbol{K}$-matrix formalism is primarily formulated for scattering processes. In the case of a decay, however, the underlying production mechanism is not constrained in the same way and must be parametrised independently. A common approach is the **$\mathcal{P}$-vector parametrisation**, in which the amplitude is written in terms of *vector* elements $a_\beta(s)$ for each final state β, rather than *matrix* elements,

$$a_\beta(s) \;=\; n_{\ell_\beta}(s)\sum_\gamma \mathcal{P}_\gamma(s)\left[\left(\boldsymbol{I} - i\,\boldsymbol{\rho}(s)\,\boldsymbol{n}(s)^2\,\boldsymbol{K}(s)\right)^{-1}\right]_{\gamma\beta}(s)\,,$$

or, in compact matrix notation like Equation (2.31),

$$\boldsymbol{a} \;=\; \boldsymbol{n}\,\mathcal{P}\,\left(\boldsymbol{I} - i\,\boldsymbol{\rho}\,\boldsymbol{n}^2\boldsymbol{K}\right)^{-1}\,.$$

The $\mathcal{P}$-vector plays a role analogous to the incoming vertex factor in Equation (2.32), but is replaced here by a **production vector** $\mathcal{P}$. Much like the $\boldsymbol{K}$-matrix, it is usually parametrised as a sum over poles [66, §50, p.14; 127, p. 425],

$$\mathcal{P}_\gamma(s) \;=\; \sum_r \frac{\alpha_r g_\gamma^r}{m_r^2 - s}\,, \tag{2.33}$$

where α_r denotes the production coupling for each pole r. While the $\boldsymbol{K}$-matrix must be real to preserve unitarity, the production coupling α_r can be complex, since it serves to absorb undetermined relative phases of the unknown production mechanism.

In practice, both the $\mathcal{P}$-vector parametrisation of Equation (2.33) and the $\boldsymbol{K}$-matrix parametrisation of Equation (2.30) are often supplemented by additional non-pole terms. These background contributions are omitted from the present discussion.

Analytic continuation

While the $\boldsymbol{K}$-matrix is real-valued and primarily constructed to preserve unitarity across overlapping resonances and thresholds, it also offers a transparent way of disentangling the underlying resonance structure. Whereas a traditional Breit–Wigner parametrisation produces a peak in the cross section characterised by a mass and a width, the $\boldsymbol{K}$-matrix leads to a pole in the complex plane that serves as a channel-independent fingerprint of an underlying physical state [132, Ch. 3]. These poles are more fundamental than the apparent mass and width of a peak, because their positions are insensitive to channel-dependent distortions and provide a process-independent definition of the underlying hadronic state. This brings us back to the analyticity of the scattering amplitude discussed in Section 2.2: the challenge now is to extract information about these poles by **analytically continuing** the physical amplitude into the complex domain where they reside.

Leaving the real axis

So far, the amplitudes in Equation (2.32) are specifically constructed over the real s axis above each minimal threshold (see Figure 2.14). This is the physical, observable domain, so we call

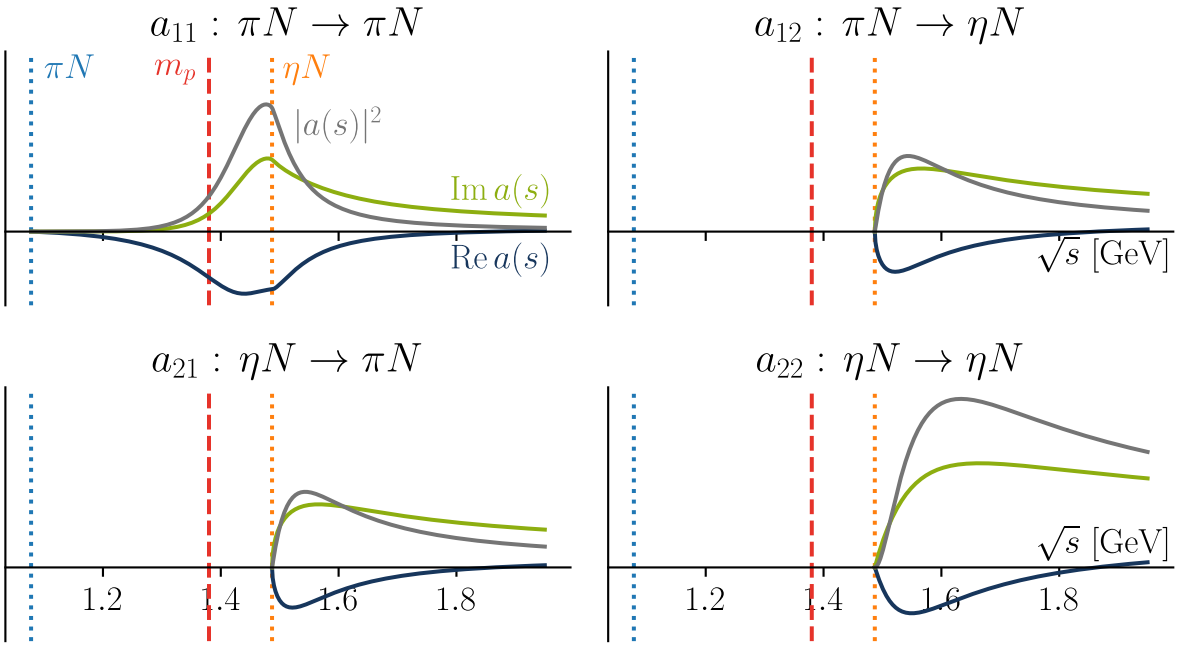


Figure 2.14. Partial-wave amplitudes (modulus squared, real part, and imaginary part) for πN (1) and ηN (2) nucleon scattering modelled with the $\boldsymbol{K}$-matrix, with one pole at the bare mass m_p set to that of the Roper resonance $N(1440)$ (P-wave with $\ell_1 = \ell_2 = 1$). The coupling constant $g_{\eta N}$ has been set to a much higher value than in reality to enhance the cusp effect at the πN threshold in a_{11}.

these constructions *physical amplitudes*. As noted in Section 2.2, we know that amplitudes have to be analytic over the full complex domain of s, but we need a recipe to continue the physical amplitude over the complex plane. Complex analysis gives us the tools to do this if we know what analytic structure the amplitude has. Once we know the location of branch cuts and poles, we can compute the amplitude at any point in the complex plane, including the domain where the poles are located. The recipe that follow applies generally to any physical partial-wave amplitude $a(s)$.

As a first step, causality in perturbation theory dictates "retarded propagation", which tells us that the physical amplitude is equal to the limit from the upper half of the complex plane of s [107, pp. 16, 90–99], that is

$$a(s) \;=\; \lim_{\epsilon \to 0^+} a(s + i\epsilon)\,. \tag{2.34}$$

This gives us a stepping stone into the upper half of the complex plane (UHP). The **Schwarz reflection principle** allows us to extend the amplitude into the lower half of the complex plane. It tells that any function f can be extended to the lower half-plane by $f(z^*) = f^*(z)$ if it and only if it [133]

1. is *continuous* on $\{z \in \mathbb{C} |\, \mathrm{Im}\, z \geq 0\}$ (closed UHP, including the real axis),
2. is *analytic* on $\{z \in \mathbb{C} |\, \mathrm{Im}\, z > 0\}$ (open UHP, excluding the real axis), and
3. returns real values on a segment of the real axis.

We already know that amplitudes are analytic in the UHP and Equation (2.34) tells us that it is continuous on the closed UHP. The third condition is satisfied through unitarity. Equation (2.18) shows that the amplitude has no imaginary part on the real axis below the smallest threshold: below the smallest threshold, there is no cross section and $|a(s)|^2$ goes to zero. This means $a(s)$ satisfies all three conditions, so that we can write $a(s^*) = a^*(s)$. The amplitude therefore has a **right-hand branch cut** that runs along the real s axis (it's imaginary part 'flips' sign when crossing the axis), starting at the first threshold opening.

The reasoning so far only reveals the analytic structure of the amplitude, but does not give a recipe to *compute* $a(s)$ for any point s in the complex plane. **Cauchy's integral formula** provides such a recipe. It tells that for any analytic function f that is analytic on or inside a closed contour C on complex plane, we can compute $f(z)$ for any point z inside C as

$$f(z) \;=\; \frac{1}{2\pi i} \oint_C \frac{f(z')}{z' - z}\, \mathrm{d}z'\,. \tag{2.35}$$

The previous argumentation already indicates that there has to be one and only one branch cut over the real axis above the threshold with a **branch point** at $s_{\mathrm{thr}} = (m_C + m_D)^2$. These branch points arise from threshold openings where a new channel becomes accessible and correspond to the boundary of the physical regions in Figure 2.6. In addition, causality tells us that the physical amplitude contains no poles. We can therefore deform any contour C that encloses a point s (see Figure 2.15) that is not on the branch cut in such a way that it only encloses the right-hand cut. Elsewhere, the contour can be deformed to infinity. If we assume that $a(s) \to 0$ as $|s| \to \infty$, Equation (2.35) over this deformed contour C becomes

$$
\begin{aligned}
a(s) &= \frac{1}{2\pi i}\left(\int_{s_\text{thr}}^{\infty}\frac{a(s'+i\epsilon)}{s'-s}\,\mathrm{d}s' + \int_{\infty}^{s_\text{thr}}\frac{a(s'-i\epsilon)}{s'-s}\,\mathrm{d}s'\right) \\
&= \frac{1}{2\pi i}\int_{s_\text{thr}}^{\infty}\frac{a(s'+i\epsilon)-a(s'-i\epsilon)}{s'-s}\,\mathrm{d}s' \\
&= \frac{1}{2\pi i}\int_{s_\text{thr}}^{\infty}\frac{\operatorname{Disc} a(s')}{s'-s}\,\mathrm{d}s' \;=\; \frac{1}{\pi}\int_{s_\text{thr}}^{\infty}\frac{\operatorname{Im} a(s')}{s'-s}\,\mathrm{d}s' \,,
\end{aligned}
\tag{2.36}
$$

where in the final step, we have used Schwarz reflection to rewrite the discontinuity as $\operatorname{Disc} a(s') = 2i \operatorname{Im} a(s')$. This is the **dispersion integral** for partial-wave amplitudes, which allows us to compute the amplitude at any point in the complex plane, provided that we know its imaginary part on the right-hand cut. An overview of more sophisticated treatments that describes subtractions in case $a(s)$ does *not* vanish as $|s| \to \infty$, see [134]. It also discusses the left-hand cut for negative s, such as the N/D approach [113].

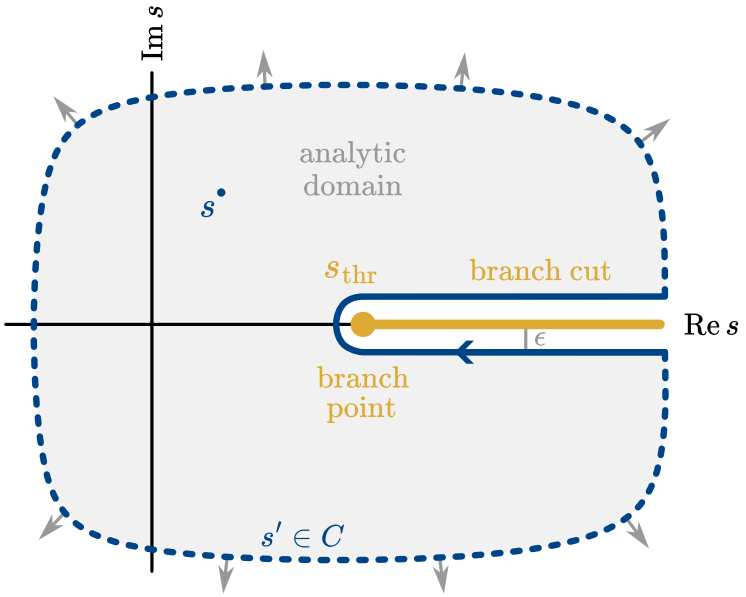


Figure 2.15. The contour C encloses the right-hand branch cut above the threshold $s_\text{thr} = (m_C + m_D)^2$ and is deformed to $|s| \to \infty$ elsewhere, so that closed integral Equation (2.35) can be used to compute the amplitude $a(s)$ for any point s in the complex plane with Equation (2.36).

Continuing with the *K*-matrix

Computing the dispersion integral for arbitrary amplitude functions is difficult and computationally intensive. This is another point where the $\boldsymbol{K}$-matrix formalism shines: amplitudes constructed via Equation (2.31) isolate the dispersive cut structure (branch cuts) from their reactive core that contains the poles. As a heuristic argument (ignoring that $\boldsymbol{K}$ is not necessarily invertible), we write Equation (2.29) as

$$
\boldsymbol{a}(s) \;=\; \left(\boldsymbol{K}^{-1}(s) - \boldsymbol{\Sigma}(s)\right)^{-1} \,, \tag{2.37}
$$

where $\boldsymbol{\Sigma}(s) = i\boldsymbol{\rho}(s)$. Since the $\boldsymbol{K}$-matrix parametrisation of Equation (2.30) is analytic (apart from poles) and does not contain cuts, the right-hand cut in the amplitude has to come from $\boldsymbol{\Sigma}$.

Notice, however, that the standard phase space factor from Equation (2.20) does not have the correct cut structure. As can be seen in Figure 2.16, it has a branch cut in its real part across the real s axis between $(m_C - m_D)^2$ and $(m_C + m_D)^2$. We therefore need to construct a function $\boldsymbol{\Sigma}(s)$ that does have the expected right-hand branch cut in the imaginary part, starting at branch point $(m_C + m_D)^2$, but that does have the same imaginary part when approaching the cut from the UHP. A similar argument holds for the more general product $i\rho(s)n_\ell^2(s)$ in Equation (2.31), which needs to be replace by an analytic **Chew–Mandelstam function** $\boldsymbol{\Sigma}_\ell(s)$ with the expected right-hand cut [113]. The barrier factors on the sides do not affect the positions of the poles and are therefore ignored in the analytic continuation of the overall partial-wave amplitude.

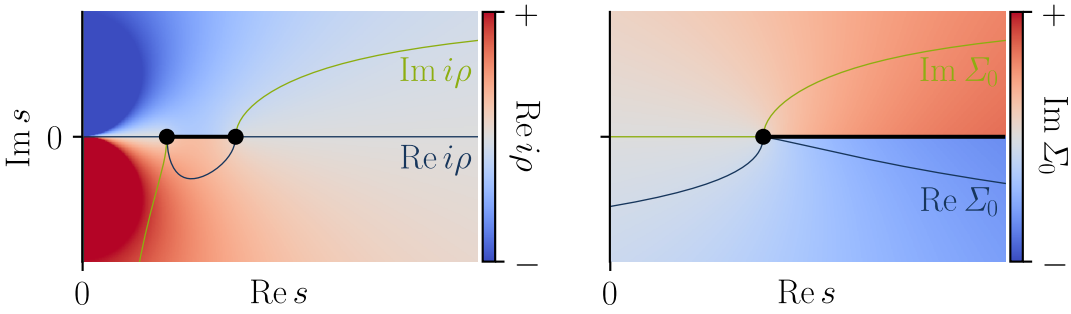


Figure 2.16. Analytic continuation of the (rotated) phase space factor $i\rho(s)$ to a Chew–Mandelstam function $\Sigma_0(s)$ that produces the expected right-hand cut (black). The real (blue) and imaginary (green) part of the respective functions evaluated just above the real s axis are also plotted.

To construct $\Sigma_\ell(s)$ for higher ℓ, we can again apply the Cauchy integral formula. This time the logic that led to Equation (2.36) reverses: we start with the assumption that we need a function $\Sigma_\ell(s)$ that has the desired right-hand cut with a discontinuity of

$$\operatorname{Disc}\Sigma_\ell(s) \;=\; 2i\,\rho(s)\,n_\ell^2(s)\,.$$

rather than its imaginary part. Since $\rho(s)\,n_\ell^2(s) \to s$ rather than 0 as $|s| \to \infty$, the dispersion integral needs to be once-subtracted [135], giving us [136; 66, p. 15]

$$\Sigma_\ell(s) \;=\; \frac{s - s_\text{thr}}{\pi} \int_{s_\text{thr}}^{\infty} \frac{\rho(s')\,n_\ell^2(s')\,\mathrm{d}s'}{(s' - s_\text{thr})(s' - s - i\epsilon)}\,. \tag{2.38}$$

An analytic solution to this integral exists if $\ell = 0$ and is given by [137]

$$\begin{aligned}\Sigma_0(s) \;=\; \frac{1}{\pi}\Bigg[&\frac{2\,q(s)}{\sqrt{s}}\log\frac{m_C^2 + m_D^2 - s + 2\,q(s)\sqrt{s}}{2\,m_C\,m_D} \\ &- (m_C^2 - m_D^2)\left(\frac{1}{s} - \frac{1}{(m_C + m_D)^2}\right)\log\frac{m_C}{m_D}\Bigg]\,.\end{aligned} \tag{2.39}$$

Simply by replacing $i\rho(s)n_\ell^2(s)$ with its analytic version $\Sigma_\ell(s)$, the physical amplitude $a_{11}(s)$ for elastic πN scattering shown in Figure 2.14 can be computed anywhere in the complex plane.

Figure 2.17 shows how the amplitude $a_{11}(s)$ for $\pi N \to \pi N$ scattering has been analytically continued from the real axis into one smooth function over the complex plane, with the expected right-hand branch cut along the real s axis starting at the πN threshold. The second branch point at the ηN threshold lies at the same position as the cusp effect seen in the lineshapes.

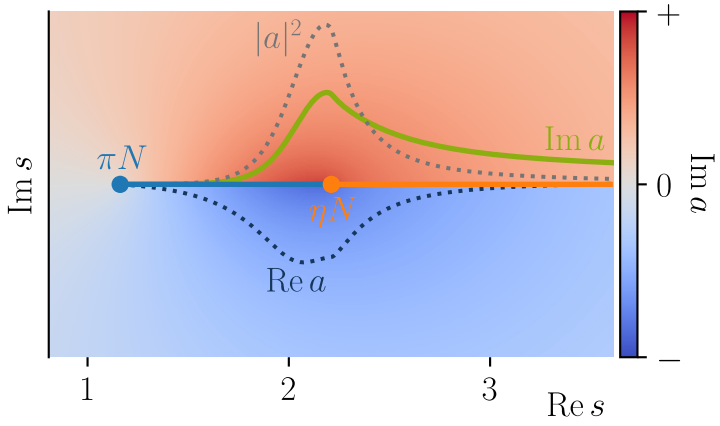


Figure 2.17. Analytic continuation into the complex plane of the physical amplitude $a_{11}(s)$ for elastic πN scattering shown in Figure 2.14. The modulus squared (dashed gray), real part (dashed blue), and imaginary part (solid green) of $a_{11}(s)$ evaluated just above the real s axis are also shown, as well as the two overlapping branch cuts for the πN and ηN thresholds.

Crossing the branch cut

The analytic continuation of the amplitude has not yet revealed any poles. This is because the amplitude function $a(s)$, while analytic and single-valued within its complex domain, becomes part of a multivalued structure when continued across its branch cuts. These continuations together form a smooth, connected complex manifold known as a **Riemann surface**. Figure 2.18 shows a three-dimensional rendering of Figure 2.17 that continues the amplitude across its branch cuts. Each continuation defines a new **Riemann sheet**, and the poles associated with resonances reside on a sheet that is not directly accessible from the initial (physical) one [138, §3.1.5]. What we typically call "the amplitude" is therefore only one branch – a single-valued function – on a particular sheet of a Riemann surface that encodes the global analytic structure of the scattering process.

The amplitude function that we have constructed is called the **physical sheet** as it is constructed from observable data on the real axis [122, pp. 344–345]. Continuing across the cut leads us to an **unphysical sheet** that contains the poles that we are interested in. When considering multiple channels, each threshold defines a new branch point with two associated sheets, leading to multiple overlapping cuts along the real axis. The Riemann surface for n channels therefore consists of 2^n sheets, each corresponding to a different combination of continuations across the cuts. The sheets are commonly denoted with Roman numerals, like $\boldsymbol{a}^{\mathrm{I}}$, $\boldsymbol{a}^{\mathrm{II}}$, et cetera, starting at physical sheet $\boldsymbol{a}^{\mathrm{I}}$.

There is no generic recipe to compute an unphysical sheet from the physical amplitude. However, we can analytically continue the physical sheet just around threshold, because the

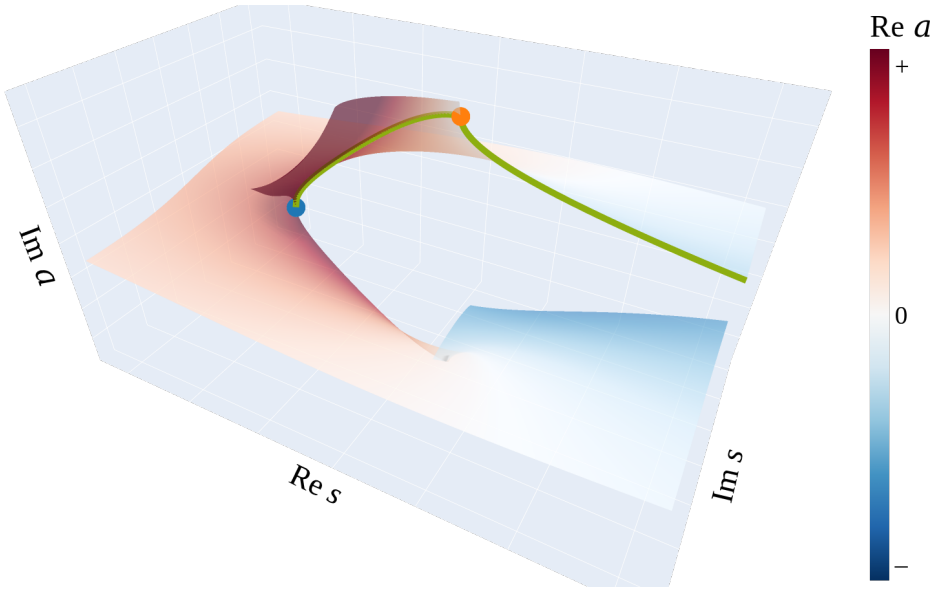


Figure 2.18. Three-dimensional rendering of the transition amplitude in Figure 2.17 with the amplitude continued further across its branch cuts to form one Riemann surface. The imaginary part of the amplitude along the real s axis is indicated in green and the πN and ηN thresholds are indicated by a blue and an orange dot, respectively.

transition between the sheets has to be continuous. The unitarity relation for partial waves Equation (2.18) can be rewritten as

$$a(s) \;=\; \frac{a(s^*)}{1 - 2i\rho(s)\,a(s^*)}\,,$$

using the fact that $2i\,\mathrm{Im}\,a = a - a^*$, $|a|^2 = a^* a$, and $a^*(s) = a(s^*)$ (Schwarz reflection). Continuity between the sheets tells that $a^{\mathrm{I}}(s + i\epsilon) = a^{\mathrm{II}}(s - i\epsilon)$ around the branch cut, which gives us

$$a^{\mathrm{II}}(s - i\epsilon) \;=\; \frac{a^{\mathrm{I}}(s - i\epsilon)}{1 - 2i\rho(s)\,a^{\mathrm{I}}(s - i\epsilon)}\,. \tag{2.40}$$

In many studies, this is taken as a general transformation rule on the whole complex plane for transitioning to the next sheet between each branch cut. In matrix notation, this would be

$$\boldsymbol{a}_{\mathrm{j}} \;=\; \left(\boldsymbol{a}_{\mathrm{i}}^{-1} - 2i\boldsymbol{\rho}_{\mathrm{j}\to\mathrm{i}}\right)^{-1} \tag{2.41}$$

with $\boldsymbol{\rho}_{\mathrm{j}\to\mathrm{i}}$ the diagonal matrix of standard phase space factors if which some elements have been set to zero for the transition from sheet i to sheet j [139]. Other sources define sheets through sign flips of the diagonal matrix of phase space elements [140, p. 666; 130, p. 6].

As an illustration of Equation (2.41), we consider the two-channel case in Figure 2.17. Applying the matrix $\boldsymbol{\rho}_{\mathrm{j}\to\mathrm{i}} = \mathrm{diag}(\rho_1, 0)$ continues $\boldsymbol{a}^{\mathrm{I}}$ into $\boldsymbol{a}^{\mathrm{II}}$ across the branch cut between the πN and ηN thresholds. The outcome is displayed in Figure 2.19: the upper half-plane and lower left of the plot show the original physical sheet, while the lower right reveals the continuation onto the adjacent sheet, where a pole emerges in the lower half-plane. This hidden pole is precisely what generates the resonant peak we previously observed in the physical amplitude a_{11} of Figure 2.14. The figure also illustrates that branch cuts are not intrinsic features of the function, but arise

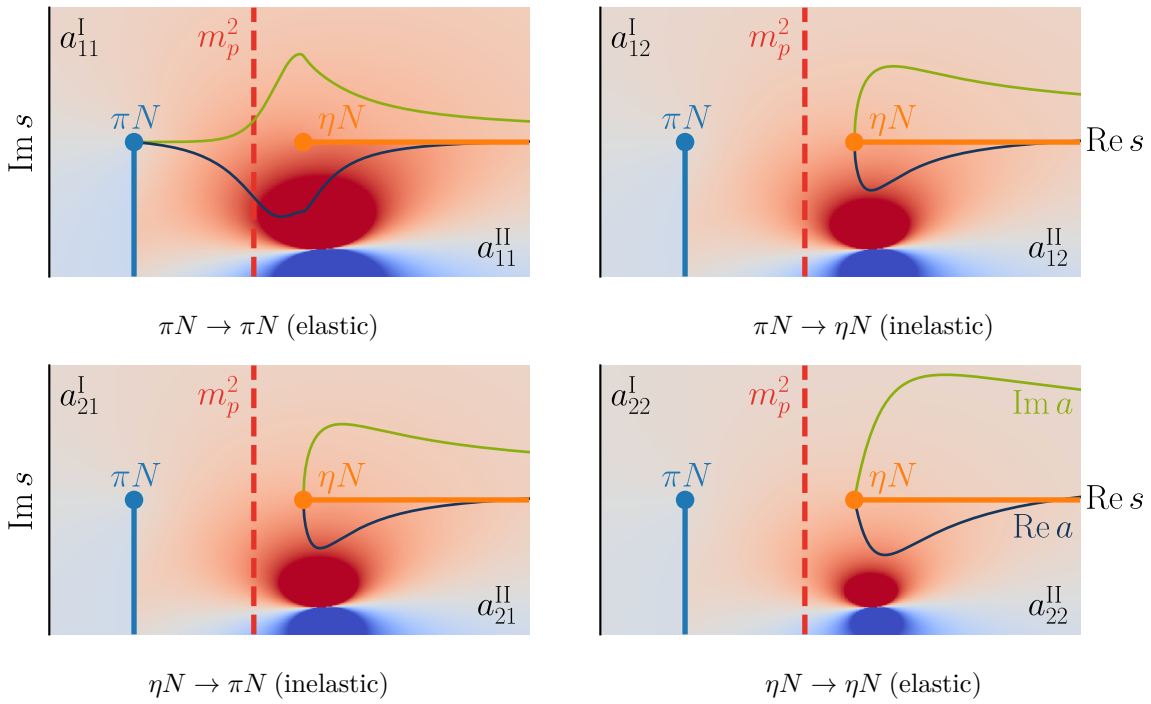


Figure 2.19. Analytic continuation of the physical amplitudes of π/η–nucleon scattering into the unphysical sheet between the two thresholds. The πN cut between sheet I and II is rotated downwards rather than to the left in order to avoid the wrong cut structure of the standard phase space factor $\rho_1(s)$ appearing in transformation Equation (2.41).

only when a multi-valued Riemann surface is represented as a single-valued function on the complex plane. Through analytic continuation – effectively "rotating" the cut – one exposes additional sheets where the underlying pole structure manifests itself.

Extracting poles and residues

Finally, we are in a position to locate pole positions. These can be found by determining where the denominator of an unphysical sheet goes to zero. Using Equation (2.41), we can translate this to the condition

$$\det\left(\boldsymbol{a}_{\mathrm{I}}(s)^{-1} - 2i\boldsymbol{\rho}_{\mathrm{j}\to\mathrm{i}}(s)\right) = 0\,.$$

This equation cannot be solved analytically, even given the simple pole parametrisation of Equation (2.30). A common trick is to numerically minimise the modulus $\left|\boldsymbol{a}_{\mathrm{I}}(s)^{-1} - 2i\boldsymbol{\rho}_{\mathrm{j}\to\mathrm{i}}(s)\right|$ with regard to s, for instance with a gradient-descent algorithm. This gives us a number of **pole positions** s_r that are potentially located in different sheets. With the $\boldsymbol{K}$-matrix parametrisation described so far, these positions are affected by threshold parameters, coupling constants, and bare masses.

Poles have a **residue** associated with each channel transition. It reveals how strongly the state couples to specific decay or production channels. The coupling constants g_α^r in Equation (2.27) encode this interaction strength and determine where and how the resonance becomes visible in

the cross section of each channel. The residue can be computed numerically with the Cauchy integral formula over a small closed contour C around the pole position s_r, giving us

$$\text{Res}(a, s_r) \;=\; \frac{1}{2\pi i} \oint_\gamma a(s')\, ds' \,.$$

Pole positions characterise the analytic structure of the entire system and define hadronic states independently of the specific process or channel in which they appear. Unlike peaks in measured cross section, which may vary with kinematic effects and interference between channels, a pole's location in the complex energy plane is a property of the underlying dynamics. It provides a channel-independent reference point that can be compared directly with non-perturbative consequences of QCD, such as those obtained from lattice calculations, dispersive analyses, or effective field theories. In this way, pole positions offer a direct connection between experimental observations and the fundamental structure of the strong interaction.

2.5. Connecting to experiment

The theory developed in this chapter provides the foundation for constructing amplitude models that respect fundamental physical principles of Section 2.2. These models encode dynamical information in terms of physically meaningful parameters, such as coupling constants and bare pole positions, and can be directly linked to observable quantities. Specifically, amplitude models give us an **intensity function**, a real-valued function of measurable kinematic variables, via the expression for the differential cross section (see Equation (2.4)). This intensity function serves as the interface between theory and the event distributions measured by experiment, enabling the extraction of model parameters from experimental data [112].

In practice, the input to an amplitude analysis consists of reconstructed four-momenta of the final-state particles for each event. These four-momenta are the raw, Lorentz-covariant observables measured in the detector. From them, one computes the relevant **kinematic variables** – such as helicity angles and Mandelstam invariants – which serve as arguments to the intensity function. Denoting the set of kinematic variables as x, the intensity function gets the form $I(x; \boldsymbol{\theta})$, with $\boldsymbol{\theta}$ the set of model parameters.

The function $I(x; \boldsymbol{\theta})$ determines the expected density of events in phase space and can be interpreted as a **probability-density function** after normalisation

$$P(x; \boldsymbol{\theta}) \;=\; \frac{I(x; \boldsymbol{\theta})}{\int_\Phi I(x'; \boldsymbol{\theta})\, \mathrm{d}x'}$$

over the physically allowed region of phase space Φ (often implicitly weighted by detector acceptance). Given a sample of N observed events $\{x_i\}$, the **likelihood function** is defined as the joint probability of the data for a given set of model parameters $\boldsymbol{\theta}$ as [141]

$$\mathcal{L}(\boldsymbol{\theta}) \;=\; \prod_{i=1}^{N} P(x_i; \boldsymbol{\theta}) \;=\; \prod_{i=1}^{N} \frac{I(x_i; \boldsymbol{\theta})}{\int_\Phi I(x; \boldsymbol{\theta})\, \mathrm{d}x} \,.$$

In experimental data, N is so large that it makes the product computationally infeasible. We therefore instead take the logarithm, which turns the likelihood into a sum: the **log-likelihood** function

$$\log \mathcal{L}(\boldsymbol{\theta}) \;=\; \sum_{i=1}^{N} \log I(x_i;\boldsymbol{\theta}) - N \log \left[\int_{\Phi} I(x;\boldsymbol{\theta})\,\mathrm{d}x \right] .$$

This has the additional benefit that the log-likelihood is additive over independent datasets, which is useful for combining results from different analyses or experiments (coupled analysis).

The integral over phase space generally has no analytical solution, particularly in multi-body decays or when accounting for detector effects. Instead, it is estimated using Monte Carlo (MC) integration, typically by generating a simulated sample uniformly distributed in phase space and summing over weights [142], giving us

$$\int_{\Phi} I(x;\boldsymbol{\theta})\,\mathrm{d}x \approx \frac{1}{M} \sum_{j=1}^{M} I(x_j^{\mathrm{MC}};\boldsymbol{\theta}) .$$

The (log-)likelihood function provides a measure (**estimator**) for how well an amplitude model with parameters $\boldsymbol{\theta}$ describes *all* observed events, even across different channels. The goal of an amplitude analysis is to find the parameter values that maximise this likelihood, which corresponds to minimising the **negative log-likelihood** (NLL) function. Finding the global minimum in the parameter space of the NLL for a specific model and dataset is often referred to as "fitting" the model to the experimental data sets.

Model fitting is a high-dimensional, non-linear optimisation problem, often complicated by strong correlations between parameters and the presence of local minima in parameter space. Standard optimisation tools such as **Minuit** are widely used due to their robustness and built-in handling of parameter boundaries and uncertainties [143]. These algorithms typically rely on gradient-based minimisation (e.g. MIGRAD), combined with error estimation techniques such as MINOS or Hessian-based covariance extraction. The computational cost of these fits is dominated by the repeated evaluation of the intensity function over all measured data events and MC-generated integration points. The efficient, vectorised, and parallelisable evaluation of arbitrary amplitude models will be the focus of Chapter 4.

3 Helicity formalism

Chapter 2 developed scattering theory techniques starting from 2-to-2 processes. In particular, this led to the partial-wave expansion (Section 2.3), which separates the amplitude into a universal angular dependence and a reduced amplitude that contains the full information about the interaction as a function of energy. Extending this framework to *multi-body* decays is more challenging, because a proper relativistic treatment of spin states must account for their dependence on the chosen reference frame. Several spin formalisms have been developed to describe transitions involving spinful particles, such as the Rarita–Schwinger formalism [144], Lorentz-*covariant* tensor approaches [145–148], and spin-projection formalisms. Among these, spin-projection formalisms – particularly the helicity formalism – are most widely used in amplitude analyses, as they are computationally convenient and align naturally with how experimental observables are expressed. However, as we will see, amplitudes constructed from helicity states are inherently frame-dependent and do not transform covariantly under Lorentz transformations. This *non-covariant* nature requires careful treatment when combining amplitudes from different spin projections across different decay chains.

The helicity formalism assumes that a multi-body decay can be described as a sequence of two-body decays [117]. In the rest frame of the parent particle, constructing the amplitude for each two-particle spin state is relatively straightforward, because conservation of angular momentum fixes how the spins couple at each vertex. By choosing convenient spin quantisation axes, the amplitudes for each decay step can then be combined into a single expression for the full decay chain, with the Lorentz-invariant parametrisations developed in Chapter 2 inserted at each two-body amplitude.

Using the helicity formalism effectively requires a clear understanding of how Lorentz transformations (boosts and rotations) affect spin states. For this reason, treatises on the helicity formalism typically follow the following structure [149–151; 102; 100; 152]:

1. Review how single-particle spin states at rest change under spatial rotations.
2. Boost and rotate these states to a convenient quantisation axis.
3. Investigate the coupling of two-particle spin states using conservation of angular momentum.
4. Construct the amplitude for a chain of two-body decays.

In this chapter, we follow the same build-up and extend the discussion to multi-body decays with decay chains that are topologically distinct.

3.1. Single-particle states

Rotating spin states

Consider a single massive particle with spin j (massless particles will be treated later in terms of their helicity). Its spin state can be denoted $|j, m\rangle$, where m is the spin projection along a chosen quantisation axis, typically the z axis. We seek to determine how this quantum state transforms under an arbitrary spatial rotation $\boldsymbol{R}(\alpha, \beta, \gamma)$, which rotates coordinate system (x, y, z) into (x', y', z') over **Euler angles** α, β, γ in the zyz**-convention**. In the active convention used here, the coordinate axes remain fixed while the particle three-vectors themselves are rotated: first by an "intrinsic twist" angle γ around the z axis, then by a "polar" angle β around the new y axis, and finally by an "azimuthal" angle α around the new z axis (see Figure 3.1). In this thesis, we use the symbols α, β, γ when deriving general angular properties and use other Greek letters like ϕ, θ, χ to refer to rotations between specific physical frames.

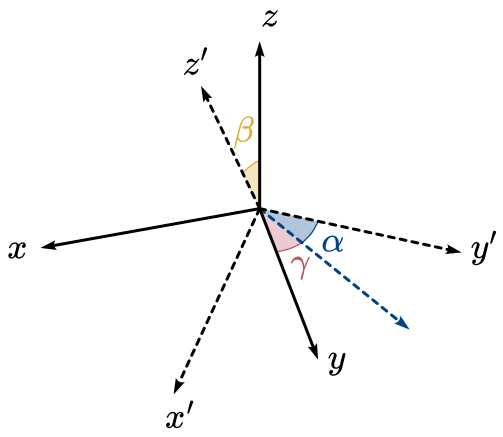


Figure 3.1. Euler angles for rotation $\boldsymbol{R}(\alpha, \beta, \gamma)$ in the zyz-convention.

In the context of spin states, the operator $\mathbf{U}[\boldsymbol{R}]$ represents the effect of the spatial rotation $\boldsymbol{R}$ on the Hilbert space in which the spin state $|j, m\rangle$ resides. Since spatial rotations are represented in quantum mechanics by unitary operators, $\mathbf{U}[\boldsymbol{R}]$ is itself unitary. For this reason, $\mathbf{U}[X]$ is often referred to as the **unitary operator** corresponding to the spatial transformation X. It now becomes important to keep track of the labels in bra-ket states like $|j, m\rangle$, as they can refer to different bases and degrees of freedom that are not always in the same Hilbert space. The operator $\mathbf{U}[\boldsymbol{R}]$ must rotate $|j, m\rangle$ according to the same group action that the spatial rotation $\boldsymbol{R}(\alpha, \beta, \gamma)$ induces in three dimensions. Because $|j, m\rangle$ is an eigenstate of the total angular momentum operators $\mathbf{J}^2$ and its z component J_z, a rotation about the z axis is generated by exponentiating J_z. Similarly, a rotation about the y axis is generated by exponentiating J_y, the y component of the total angular momentum $\mathbf{J}$. This yields the standard factorisation for a rotation,

$$\mathbf{U}[\boldsymbol{R}(\alpha, \beta, \gamma)] \;=\; e^{-i\alpha J_z}\, e^{-i\beta J_y}\, e^{-i\gamma J_z}\,. \tag{3.1}$$

The rotated state can be expanded as a linear combination of basis states $|j, m'\rangle$ defined with respect to the new quantisation axis z'. Because we have decomposed $\boldsymbol{R}$ into a sequence of rotations in which two are about the quantisation axis z, the expansion can be written explicitly as

$$\begin{aligned} \mathbf{U}[\boldsymbol{R}(\alpha,\beta,\gamma)]\,|j,m\rangle &= \sum_{m'} e^{-im'\alpha}\, d^{j}_{m'm}(\beta)\, e^{-im\gamma}\ |j,m'\rangle \\ &= \sum_{m'} D^{j}_{m'm}(\alpha,\beta,\gamma)\ |j,m'\rangle\ , \end{aligned} \tag{3.2}$$

where $d^{j}_{m'm}$ and $D^{j}_{m'm}$ are the elements of the (small) Wigner d- and Wigner D-matrices of the total angular momentum representation (German: "**D**arstellung") in spin space [153]. The Wigner d-matrix is a $2j+1$-dimensional matrix that encodes the rotation about the y axis, while the Wigner D-matrix represents the full unitary operator in the spin basis. Note that a rotation about the z axis acts only by shifting the phase of the z component of the angular momentum, since $e^{i\phi J_z}\,|j,m\rangle = e^{i\phi m}\,|j,m\rangle$, giving us

$$\begin{aligned} D^{j}_{m'm}(\alpha,\beta,\gamma) &= \langle j,m'\,|\,\mathbf{U}[\boldsymbol{R}(\alpha,\beta,\gamma)]\,|\,j,m\rangle\ , \\ d^{j}_{m'm}(\beta) &= \big\langle j,m'\,\big|\,e^{-i\beta J_y}\,\big|\,j,m\big\rangle\ . \end{aligned}$$

It is important to keep track of the direction of rotation, as in some instances, the inverse rotation will be used. This results in a conjugate of the Wigner D-matrix with flipped lower indices, giving

$$\mathbf{U}\big[\boldsymbol{R}(\alpha,\beta,\gamma)^{-1}\big]\,|j,m\rangle = \sum_{m'} D^{j\,*}_{mm'}(\alpha,\beta,\gamma)\ |j,m'\rangle\ . \tag{3.3}$$

It should be noted that the spatial rotations $\boldsymbol{R}$ belong to the **special orthogonal group** SO(3), whereas the corresponding unitary operators $\mathbf{U}[\boldsymbol{R}]$ belong to the **special unitary group** SU(2). These two Lie groups are not isomorphic. Rather, there is a two-to-one homomorphism between SU(2) and SO(3), meaning that SU(2) is a **double cover** of SO(3). As a result, while a spatial rotation in SO(3) by 2π around any axis leaves the spatial orientation unchanged, its corresponding rotation in SU(2) produces a spin state with reversed sign. Only after a 4π rotation in space does the spin state return to its original form. We will return to this subtlety in Section 3.3.

Boosting spin states

Knowing how spin states at rest transform under spatial rotations, we can now examine how they transform under general Lorentz transformations. This provides a consistent description of a massive particle with spin j and four-momentum $p = (E, p_x, p_y, p_z)$, which is an essential ingredient for constructing amplitudes in two-body decay chains.

So far, we described spin states in terms of spinors transforming under the rotation group SU(2), which is isomorphic to the three-dimensional spin group Spin(3). To include boosts as well as rotations, the symmetry group must be extended to the (orthochronous) Lorentz group $\mathrm{SO}^{+}(1,3)$. Its double cover is $\mathrm{Spin}(1,3) \cong \mathrm{SL}(2,\mathbb{C})$, under which spin states transform. This extension introduces two additional features: (1) Wigner rotations, which describe the induced rotation of spin under Lorentz boosts, and (2) the coupling of spin with four-momentum, most commonly expressed through helicity.

Lorentz transformations include both pure boosts and rotations. A boost that brings the rest-frame momentum to a spatial momentum $\vec{p} = (p_x, p_y, p_z)$ is denoted by $\boldsymbol{\Lambda}(\vec{p})$. Analogous to $\mathbf{U}[\boldsymbol{R}]$, the action of this boost on a spin state is represented by a unitary operator $\mathbf{U}[\boldsymbol{\Lambda}(\vec{p})]$. This relates the moving spin state $|\vec{p},j,m\rangle$ to the **rest-frame spin state** $|j,m\rangle \equiv |\vec{0},j,m\rangle$ via

$$|\vec{p},j,m\rangle = \mathbf{U}[\boldsymbol{\Lambda}(\vec{p})]\ |j,m\rangle\ .$$

Here, the spin projection m is defined in the particle's rest frame. As with spatial rotations (Equation (3.1)), the operator $\mathbf{U}[\boldsymbol{\Lambda}(\vec{p})]$ can be written in terms of the boost generators $\vec{\mathbf{K}}$. Together with the rotation generators $\vec{\mathbf{J}}$, the six operators $\mathbf{K}_i$ and $\mathbf{J}_i$ span the Lie algebra $\mathfrak{so}(1,3)$ of the Lorentz group $\mathrm{SO}(1,3)$. Explicitly, this gives us

$$\mathbf{U}[\boldsymbol{\Lambda}(\vec{p})] \;=\; \exp(-i\,\vec{p}\cdot\vec{\mathbf{K}}) \;=\; e^{-i\,\mathbf{K}_x p_x}\, e^{-i\,\mathbf{K}_y p_y}\, e^{-i\,\mathbf{K}_z p_z}.$$

However, unlike in Equation (3.2), this representation does not directly help to decompose the boosted state $|\vec{p},j,m\rangle$ into a linear combination of basis states with definite angular properties. Instead, we exploit the fact that a Lorentz boost along an arbitrary direction $\vec{p}=(p,\theta,\phi)$ can be written as a rotation and a boost along the z axis. This factorisation will allow us to construct one convenient description of the spin state for a particle with momentum $\vec{p}$. Later, we will see that there is an alternative but equivalent description that is often more practical for analysing angular distributions in decays.

The factorisation works as follows. To boost a state with momentum $\vec{p}$, we first rotate the momentum into the z direction with $\mathring{\boldsymbol{R}}^{-1}$, then apply the boost $\boldsymbol{\Lambda}_z(p)$ along z, and finally rotate back with $\mathring{\boldsymbol{R}}$:

$$\boldsymbol{\Lambda}(\vec{p}) \;=\; \mathring{\boldsymbol{R}}(\phi,\theta,0)\,\boldsymbol{\Lambda}_z(p)\,\mathring{\boldsymbol{R}}^{-1}(\phi,\theta,0)\,.$$

Crucially, since unitary representations are multiplicative,

$$\mathbf{U}[\boldsymbol{\Lambda}_1\,\boldsymbol{\Lambda}_2] \;=\; \mathbf{U}[\boldsymbol{\Lambda}_1]\,\mathbf{U}[\boldsymbol{\Lambda}_2]\,, \tag{3.4}$$

and satisfies $\mathbf{U}[\boldsymbol{R}^{-1}]=\mathbf{U}^{-1}[\boldsymbol{R}]$, we find

$$\mathbf{U}[\boldsymbol{\Lambda}(\vec{p})] \;=\; \mathbf{U}[\mathring{\boldsymbol{R}}(\phi,\theta,0)]\,\mathbf{U}[\boldsymbol{\Lambda}_z(p)]\,\mathbf{U}^{-1}[\mathring{\boldsymbol{R}}(\phi,\theta,0)]\,. \tag{3.5}$$

The state $|\vec{p},j,m\rangle$ defined in this way is known as the **canonical basis** of the spin state (see Figure 3.2). It is obtained by applying a pure Lorentz boost to the rest-frame state:

$$|\vec{p},j,m\rangle \;=\; \mathbf{U}[\boldsymbol{\Lambda}]\;|j,m\rangle \;=\; \mathbf{U}[\mathring{\boldsymbol{R}}]\,\mathbf{U}[\boldsymbol{\Lambda}_z]\,\mathbf{U}^{-1}[\mathring{\boldsymbol{R}}]\;|j,m\rangle\;. \tag{3.6}$$

Importantly, Equation (3.6) shows that a spin state in the canonical basis transforms under spatial rotations just like the spin state at rest in Equation (3.2). Applying a rotation $\boldsymbol{R}$ to the canonical state $|\vec{p},j,m\rangle$, we find

$$\begin{aligned}
\mathbf{U}[\boldsymbol{R}]\;|\vec{p},j,m\rangle \;&=\; \mathbf{U}[\boldsymbol{R}]\;\underbrace{\mathbf{U}[\mathring{\boldsymbol{R}}]\,\mathbf{U}[\boldsymbol{\Lambda}_z]\,\mathbf{U}^{-1}[\mathring{\boldsymbol{R}}]}_{\mathbf{U}[\boldsymbol{\Lambda}(\vec{p})]}\;\underbrace{\mathbf{U}[\boldsymbol{R}^{-1}]\,\mathbf{U}[\boldsymbol{R}]}_{1}\;|j,m\rangle \\
&=\; \underbrace{\mathbf{U}[\boldsymbol{R}\,\mathring{\boldsymbol{R}}]\,\mathbf{U}[\boldsymbol{\Lambda}_z]\,\mathbf{U}^{-1}[\boldsymbol{R}\,\mathring{\boldsymbol{R}}]}_{\mathbf{U}[\boldsymbol{\Lambda}(\boldsymbol{R}\vec{p})]}\;\mathbf{U}[\boldsymbol{R}]\;|j,m\rangle \\
&=\; \mathbf{U}[\boldsymbol{\Lambda}(\boldsymbol{R}\vec{p})]\sum_{m'} D^{j}_{m'm}(\boldsymbol{R})\;|j,m'\rangle \\
&=\; \sum_{m'} D^{j}_{m'm}(\boldsymbol{R})\;|\boldsymbol{R}\vec{p},j,m'\rangle\;.
\end{aligned} \tag{3.7}$$

In the canonical basis, the spin quantisation axis is fixed (e.g. along the laboratory z axis) and therefore not generally aligned with the momentum of the particle. An alternative is the **helicity basis** (see Figure 3.2), in which the quantisation axis is chosen along the particle's momentum direction. In this basis, the spin projection along the momentum is called the **helicity** λ, and the

spin state is denoted by $|\vec{p}, j, \lambda\rangle$. An additional advantage of the helicity basis is that it remains well-defined even for massless particles.

The relativistic helicity state can be constructed from the rest-frame spin state by first boosting along the z axis and then rotating the momentum direction into $\vec{p}$, or equivalently (using Equation (3.5)), by boosting directly along $\vec{p}$ and then rotating the spin quantisation axis accordingly. Explicitly, this yields

$$\begin{aligned} |\vec{p}, j, \lambda\rangle &= \mathbf{U}[\mathring{\boldsymbol{R}}(\phi, \theta, 0)]\, \mathbf{U}[\boldsymbol{\Lambda}_z(p)]\, |j, \lambda\rangle \\ &= \mathbf{U}[\boldsymbol{\Lambda}(\vec{p})]\, \mathbf{U}[\mathring{\boldsymbol{R}}(\phi, \theta, 0)]\, |j, \lambda\rangle \ , \end{aligned}$$

which is essentially the same as Equation (3.6), but without the final rotation $\mathbf{U}^{-1}[\mathring{\boldsymbol{R}}]$ that restores the canonical orientation. The helicity λ is invariant under rotations about the momentum direction and under boosts along $\vec{p}$. For an arbitrary rotation $\boldsymbol{R}$, we find

$$\begin{aligned} \mathbf{U}[\boldsymbol{R}]\ |\vec{p}, j, \lambda\rangle &= \mathbf{U}[\boldsymbol{R}\,\mathring{\boldsymbol{R}}]\, \mathbf{U}[\boldsymbol{\Lambda}_z]\ |j, \lambda\rangle \\ &= |\boldsymbol{R}\vec{p}, j, \lambda\rangle \ . \end{aligned} \tag{3.8}$$

Compare this to Equation (3.7) for the canonical basis, where the rotation mixes different spin projection states. Similarly, a boost $\boldsymbol{\Lambda}'$ that maps $\vec{p}$ to a new momentum $\vec{p}\,'$ *parallel* to $\vec{p}$ leaves the helicity unchanged:

$$\begin{aligned} \mathbf{U}[\boldsymbol{\Lambda}(\vec{p}\,')]\ |\vec{p}, j, \lambda\rangle &= \mathbf{U}[\boldsymbol{\Lambda}(\vec{p}\,')]\, \mathbf{U}[\boldsymbol{\Lambda}(\vec{p})]\, \mathbf{U}[\mathring{\boldsymbol{R}}]\ |j, \lambda\rangle \\ &= \mathbf{U}[\boldsymbol{\Lambda}(\vec{p}\,')]\, \mathbf{U}[\mathring{\boldsymbol{R}}]\ |j, \lambda\rangle \\ &= |\vec{p}\,', j, \lambda\rangle \ . \end{aligned} \tag{3.9}$$

In summary, helicity states are invariant under rotations and boosts along the momentum direction, which makes the helicity basis especially convenient for describing multi-body decays. In this basis, the sequential factorisation into two-body amplitudes becomes transparent, with each two-body decay carrying an angular structure that arises from the coupling of spin and orbital angular momenta.

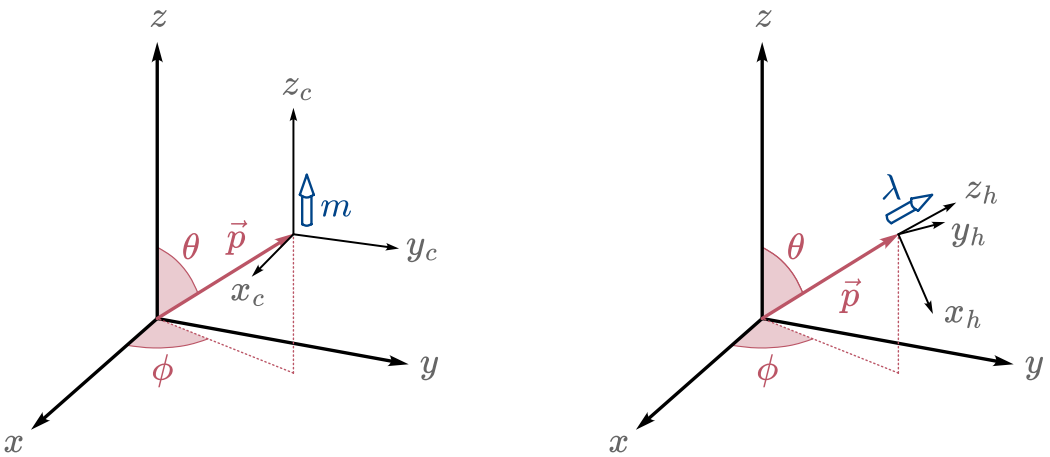


Figure 3.2. The canonical basis $|\vec{p}, j, m\rangle$ (left) and the helicity basis $|\vec{p}, j, \lambda\rangle$ (right). The quantisation axes in each frame are indicated with a blue arrow.

3.2. Two-particle states

We now construct the **two-particle spin state** for a decay $a \to 1\,2$, in which a parent particle a decays into two daughter particles 1 and 2. We first review the standard theory of how spin states couple in the rest frame, which is necessary to correctly extend the construction to the relativistic case. Just like in the single-particle spin state (Section 3.1), this leads to two practical spin bases for describing the two-particle state: the canonical basis and the helicity basis. The canonical basis is more intuitive, as it is built from the canonical spin states of the decay products. The helicity basis is more convenient for constructing amplitudes in decay chains, because, like the single-particle helicity basis, it is invariant under rotations (up to phases coming from the SU(2) group, which must be tracked carefully to avoid apparent contradictions).

Joint tensor product state

Two-particle spin states can be constructed from the tensor product of two single-particle spin states. This is the next step towards formulating the amplitude of a multi-body decay as a chain of two-body decays. The tensor product combines the two states into a single state that lives in a Hilbert space of higher dimension. If J is the (total) angular momentum of a state $|J, M\rangle$ with spin projection M, then the dimension of its Hilbert space is $2J + 1$. The dimension of the tensor product space is the product of the dimensions of the individual spaces, i.e. $(2J_1 + 1)(2J_2 + 1)$, and its possible total angular momentum quantum numbers range from $|J_1 - J_2|$ to $J_1 + J_2$.

Imagine we have two such states, $|J_1, M_1\rangle$ and $|J_2, M_2\rangle$, where J_1 and J_2 are the intrinsic spins (or total angular momenta) of the two particles, and M_1, M_2 are their respective projections along a chosen quantisation axis. In the helicity formalism, these spins J_1, J_2 are assumed to be known: each two-body decay product is either a known final-state particle or an intermediate resonance, in which case its spin state is itself treated as a definite $|J, M\rangle$ state. The composite two-particle state, or **joint state**, is then given in the uncoupled basis by the **tensor product**

$$|J_1, M_1; J_2, M_2\rangle \equiv |J_1, M_1\rangle \otimes |J_2, M_2\rangle \ .$$

At this stage, we only know that this is a mathematical object living in a Hilbert space that is the tensor product of the two Hilbert spaces of the individual particles. For our purposes, it is more convenient to represent this higher-dimensional state in terms of **total spin** basis vectors, denoted $|S, M_S\rangle$, which form the irreducible representation basis (see also Section 1.2) for the combined spin. The individual spin states $|J_i, M_i\rangle$ are eigenstates of the spin operators $\mathbf{J}_i$, while the coupled states $|S, M_S\rangle$ are eigenstates of the total spin operator $\mathbf{S} = \mathbf{J}_1 + \mathbf{J}_2$.

The relation between these two bases is given by a unitary transformation, whose matrix elements are the **Clebsch–Gordan coefficients**,

$$C^{S,M_S}_{J_1,M_1;J_2,M_2} \ = \ \langle J_1, M_1; J_2, M_2 | S, M_S\rangle \ .$$

Written as a matrix (for fixed J_1, J_2, S), the rows correspond to the uncoupled basis $\langle J_1, M_1; J_2, M_2|$ and the columns to the coupled basis $|S, M_S\rangle$ [66, §46]. These coefficients show which combinations of individual spin projections contribute to each allowed total spin state. A familiar consequence is that the total spin S can only take values between $|J_1 - J_2|$ and $J_1 + J_2$ in integer steps, and that $M_S = M_1 + M_2$.

Applying this unitary transformation explicitly yields:

$$|S, M_S\rangle \;=\; \sum_{M_1,M_2} C^{S,M_S}_{J_1,M_1;J_2,M_2} \left(|J_1, M_1\rangle \otimes |J_2, M_2\rangle\right) , \tag{3.10}$$

or, equivalently,

$$|J_1, M_1\rangle \otimes |J_2, M_2\rangle \;=\; \sum_{S,M_S} C^{S,M_S}_{J_1,M_1;J_2,M_2} \, |S, M_S\rangle \; . \tag{3.11}$$

The inverse relation takes the same form, because the Clebsch–Gordan coefficients are real in the standard convention, so the same coefficients appear in both directions of the basis change.

Coupling to the decaying particle

Once we assume that two particles are produced in the decay of a parent state $|J_a, M_a\rangle$, their joint state is described as a **coupled state**. The decay process must conserve total angular momentum, so the spin J_a of particle a is carried by the combination of the two final-state spins and their relative motion. To describe this, we introduce the **orbital angular momentum** $|L, M_L\rangle$, which accounts for the relative motion of the two particles in the decay. The **total angular momentum** $|J, M_J\rangle$ is then formed by coupling the joint total spin state $|S, M_S\rangle$ to the orbital part $|L, M_L\rangle$. In operator terms, if $|L, M_L\rangle$ are eigenstates of the orbital angular momentum operator $\mathbf{L}$, then the total angular momentum operator is given by angular-momentum addition, $\mathbf{J} = \mathbf{L} + \mathbf{S}$ (see Figure 3.3).

Applying the same Clebsch–Gordan machinery as before, we have

$$|J, M_J\rangle \;=\; \sum_{\substack{L,M_L \\ S,M_S}} C^{J,M_J}_{L,M_L;S,M_S} \left(|L, M_L\rangle \otimes |S, M_S\rangle\right) .$$

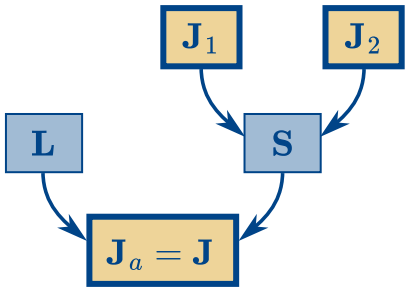


Figure 3.3. The two spin-coupling stages, $\mathbf{S} = \mathbf{J}_1 + \mathbf{J}_2$ and $\mathbf{J} = \mathbf{L} + \mathbf{S}$, for the decay $a \to 1\,2$.

Unlike in Equation (3.10), we sum over all allowed values of L, M_L, S, M_S, because the intrinsic spins J_1, J_2 are known, but the intermediate total spin S and orbital angular momentum L are *not* fixed in advance. Alternatively, we can keep L and S explicit as known quantum numbers and write

$$|J, M_J, L, S\rangle \;=\; \sum_{M_L,M_S} C^{J,M_J}_{L,M_L;S,M_S} \left(|L, M_L\rangle \otimes |S, M_S\rangle\right) . \tag{3.12}$$

Combining this with Equation (3.10) yields a fully expanded form in terms of the known intrinsic spin states,

$$
\begin{aligned}
|J, M_J, L, S\rangle \;=\; & \\
& \sum_{M_L, M_S, M_1, M_2} C^{J,M_J}_{L,M_L;S,M_S}\, C^{S,M_S}_{J_1,M_1;J_2,M_2} \\
& \qquad (|L, M_L\rangle \otimes |J_1, M_1\rangle \otimes |J_2, M_2\rangle) \ .
\end{aligned}
\tag{3.13}
$$

Finally, conservation of angular momentum requires that the quantum numbers match those of the initial parent particle: $J = J_a$ and $M_J = M_a$. The resulting state defines the **canonical basis** for the two-particle system. It plays a role analogous to the canonical basis of a single-particle spin state, but now the pair (L, S) serves as the two-particle analogue of the single spin projection m. Unlike the single-particle case (see Figure 3.2), these internal quantum numbers do not have a simple interpretation as directions in physical space. We will revisit this when introducing the helicity basis, which reformulates the spin states in a relativistic way using the four-momenta of the decay products.

Relativistic, coupled states

In the rest frame, two spin states are coupled with Clebsch–Gordan coefficients. To describe decays where the particles carry momentum, this tensor-product construction must be extended to account for Lorentz boosts. As in the single-particle case (Section 3.1), we first obtain the canonical basis from the boosted canonical spin states of the decay products, and then the helicity basis through spin projections along the momentum of each particle. In both cases, the kinematic degrees of freedom are separated step by step(see Figure 3.4).

Canonical basis

The "relativistic" spin states of the decay products take the form $|\vec{p}_i, J_i, M_i\rangle$, as introduced in Equation (3.6). The relativistic analogue of the coupled spin state from Equation (3.10) can then be written as

$$
|\vec{p}_a, S, M_S\rangle \;=\; \sum_{M_1, M_2} C^{S,M_S}_{J_1,M_1;\,J_2,M_2} \left(|\vec{p}_1, J_1, M_1\rangle \otimes |\vec{p}_2, J_2, M_2\rangle\right) . \tag{3.14}
$$

Here, $\vec{p}_a$ denotes the total three-momentum of the two-particle system. While formally $\vec{p}_a = \vec{p}_1 + \vec{p}_2$, it is crucial to note that the individual spin states depend on the reference frame. The coupling in Equation (3.14) strictly applies only in the rest frame of the two-body system. The necessary Lorentz transformations introduce Wigner rotations of the spin states, as discussed in Section 3.3.

In the center-of-mass (CM) frame, where $\vec{p}_a = 0$, the two-particle state of Equation (3.14) is fully characterised by the magnitude of the relative momentum, $p = |\vec{p}_1| = |\vec{p}_2|$, and its orientation given by the polar and azimuthal angles θ, ϕ (see Figure 3.5). The magnitude p is related to the CM energy s via the breakup momentum $q(s)$ defined in Equation (2.20).

Factoring out the total-momentum eigenstate $|p_a\rangle$ of the four-momentum operator $\mathbf{P}$, the two-particle state becomes

$$
|\vec{p}_a, \theta, \phi, S, M_S\rangle \;=\; (2\pi)^3 \left[\frac{4\sqrt{s}}{p}\right]^{\frac{1}{2}} \left(|\theta, \phi, S, M_S\rangle \otimes |p_a\rangle\right) .
$$

Since we work in the CM frame with $\vec{p}_a = 0$, the dependence on $|p_a\rangle$ can be omitted, leaving only the energy-dependent normalisation factor.

To construct the relativistic equivalent of $|L, M_L\rangle \otimes |S, M_S\rangle$ as in Equation (3.12), we integrate over the angular variables (θ, ϕ). This procedure mirrors the discrete Clebsch–Gordan decomposition but now employs spherical harmonics $Y^L_{M_L}(\theta, \phi)$ as continuous basis functions. The resulting integral reads

$$|L, M_L, S, M_S\rangle \;=\; N_L \iint d\phi\, d\theta\, Y^L_{M_L}(\theta, \phi)\, |\theta, \phi, S, M_S\rangle \;, \tag{3.15}$$

with normalisation $N_L = \sqrt{\frac{2L+1}{4\pi}}$ [154, §4.2]. This integration removes the angular dependence, yielding states that are labelled by fixed orbital angular momentum L. These states $|L, M_L, S, M_S\rangle$ transform under rotations just like tensor-product states of angular momentum eigenstates. Specifically, applying a rotation $\boldsymbol{R}$ yields [150, p. 9]

$$\mathbf{U}[\boldsymbol{R}]\; |L, M_L, S, M_S\rangle \;=\; \sum_{M'_L, M'_S} D^L_{M'_L M_L}(R)\, D^S_{M'_S M_S}(R)\; |L, M'_L, S, M'_S\rangle \;.$$

Likewise, for the tensor-product states, we have

$$\mathbf{U}[\boldsymbol{R}]\, (|L, M_L\rangle \otimes |S, M_S\rangle) \;=\; \sum_{M'_L, M'_S} D^L_{M'_L M_L}(R)\, D^S_{M'_S M_S}(R)\; |L, M'_L\rangle \otimes |S, M'_S\rangle \;.$$

For all practical purposes, we can therefore identify $|L, M_L, S, M_S\rangle \equiv |L, M_L\rangle \otimes |S, M_S\rangle$. In analogy with Equation (3.12), we can couple these and form states of definite total angular momentum J using

$$\begin{aligned} |J_a, M_a, L, S\rangle \;&=\; \sum_{M_L, M_S} C^{J_a, M_a}_{S, M_S; L, M_L}\, |L, M_L, S, M_S\rangle \\ &=\; N_L \sum_{M_L, M_S} C^{J_a, M_a}_{S, M_S; L, M_L} \iint d\phi\, d\theta\, Y^L_{M_L}(\theta, \phi)\, |\theta, \phi, S, M_S\rangle \;. \end{aligned}$$

This completes the construction of the **canonical basis** of Equation (3.12) from two-particle momentum states, serving as the two-body analogue of Equation (3.6). Note that L and S label rotationally invariant properties, meaning that

$$\mathbf{U}[\boldsymbol{R}]\; |J_a, M_a, L, S\rangle \;=\; \sum_{M'} D^J_{M' M_a}(R)\; |J_a, M', L, S\rangle \;.$$

Expressing two-particle states in this form provides access to the angular-momentum content explicit, which facilitates the inclusion of dynamical factors, such as the centrifugal suppression factors discussed in Section 2.4.

Helicity basis

One limitation of the canonical basis is that it is not invariant under Lorentz boosts. This becomes problematic when constructing amplitudes for cascades of two-body decays, where intermediate frames vary along different decay chains. As we saw in the single-particle case (Section 3.1), it is often more convenient to choose the particle's direction of motion as the spin quantisation axis. For a two-particle system, we typically define the quantisation axis along the momentum of particle 1 (consistent with how the angle θ, ϕ were defined). This leads naturally to the helicity quantum numbers λ_1, λ_2 and their corresponding helicity states $|J_1, \lambda_1\rangle$ and $|J_2, \lambda_2\rangle$. The difference between the canonical and helicity constructions is shown in Figure 3.5.

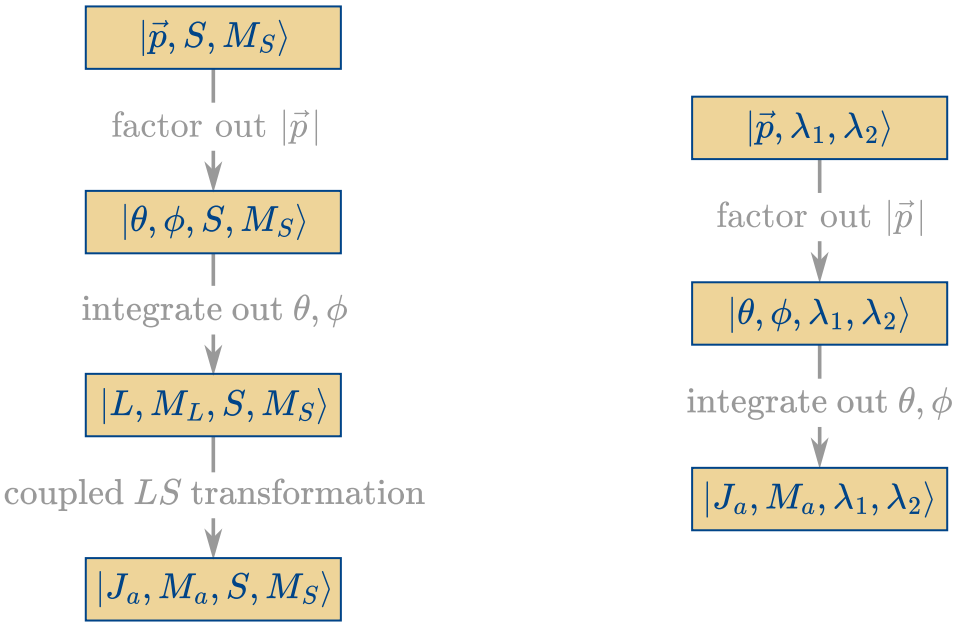


Figure 3.4. Construction schemes for the canonical basis (left) and helicity basis (right) of the two-particle system.

In analogy with Equation (3.14), to construct the two-particle state in the helicity basis, we again start from the tensor product of the individual helicity states. However, because helicity is defined as the projection of spin along the particle's own momentum direction, we must first rotate our standard frame's z axis into the direction of $\vec{p}_1$. This is accomplished by a rotation $\mathring{\boldsymbol{R}}(\theta, \phi, 0)$, which aligns the z axis with the momentum of particle 1.

Starting from a reference state $|0, 0, \lambda_1, \lambda_2\rangle$, where both particles in the direction of the z axis along which the helicities are defined, we obtain the physical state via

$$\begin{aligned} |\theta, \phi, \lambda_1, \lambda_2\rangle &= \mathbf{U}[\mathring{\boldsymbol{R}}(\theta, \phi, 0)] \left(|s_1, \lambda_1\rangle \otimes |s_2, \lambda_2\rangle\right) \\ &\equiv \mathbf{U}[\mathring{\boldsymbol{R}}] \; |0, 0, \lambda_1, \lambda_2\rangle \; . \end{aligned}$$

This ensures that the helicities λ_1, λ_2 are defined with respect to the physical momentum directions of the particles.

Similar to Equation (3.15), we integrate over the solid angle $\mathrm{d}\Omega' = \sin\theta' \, \mathrm{d}\theta' \, \mathrm{d}\phi'$ to obtain states of definite total angular momentum J. Conservation of angular momentum implies $J_a = J$ and $M_a = M_J$, leading to the **helicity basis** state

$$|J_a, M_a, \lambda_1, \lambda_2\rangle = \frac{N_J}{2\pi} \iint \mathrm{d}\Omega' \, D^{J*}_{M_a, \lambda_1 - \lambda_2}(\theta', \phi') \, \mathbf{U}[\mathring{\boldsymbol{R}}] \; |0, 0, \lambda_1, \lambda_2\rangle \; . \tag{3.16}$$

The appearance of the difference $\lambda_1 - \lambda_2$ in the Wigner D-function reflects the fact that the two decay products emerge back-to-back in the parent rest frame (Figure 3.5, right). Under a rotation with angle ϕ about the parent's momentum axis, particle 1 picks up a phase $e^{-i\lambda_1\phi}$ while particle 2 picks up $e^{+i\lambda_2\phi}$. The combined state therefore transforms with net phase $e^{-i(\lambda_1 - \lambda_2)\phi}$, which fixes the lower index of the Wigner D-function. The rotation angles (θ, ϕ) are defined as the polar and azimuthal angles of particle 1 in the parent rest frame, so the convention of "$\lambda_1 - \lambda_2$" is tied to choosing particle 1 as the helicity reference axis.

Importantly, these states transform under rotations just like the single-particle helicity states introduced in Equation (3.8), with J and M_J being invariant under any rotation $\boldsymbol{R}$ [150, p. 10], meaning that

$$\mathbf{U}[\boldsymbol{R}]\ |J_a, M_a, \lambda_1, \lambda_2\rangle \ = \ \sum_{M'_a} D^{J}_{M'_a M_a}(R)\ |J_a, M'_a, \lambda_1, \lambda_2\rangle \ .$$

This property will later enable the construction of amplitudes for decay chains involving sequential two-body decays.

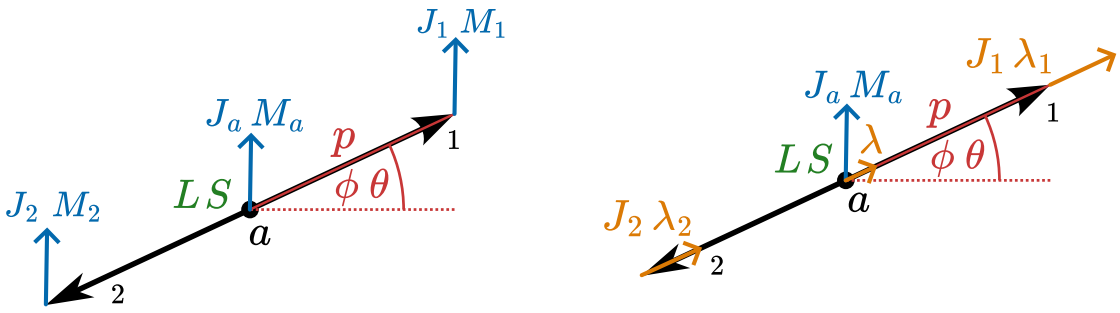


Figure 3.5. Two-particle state in the canonical basis (left) and the helicity basis (right).

Transformation between the two bases

The two formulations serve as basis vectors to the Hilbert space of spin states. One can therefore find a transformation to switch between the helicity and canonical bases [154, §4.3]

$$\begin{aligned} |J, M, \lambda_1, \lambda_2\rangle \ &= \ \sum_{L,S} \sqrt{\frac{2L+1}{2J+1}} C^{J,\lambda}_{L,0,S,\lambda} C^{S,\lambda}_{J_1,\lambda_1,J_2,\text{-}\lambda_2} |J, M, L, S\rangle \\ |J, M, L, S\rangle \ &= \ \sum_{\lambda_1,\lambda_2} \sqrt{\frac{2L+1}{2J+1}} C^{J,\lambda}_{L,0,S,\lambda} C^{S,\lambda}_{J_1,\lambda_1,J_2,\text{-}\lambda_2} |J, M, \lambda_1, \lambda_2\rangle \end{aligned} \tag{3.17}$$

with $\lambda = \lambda_1 - \lambda_2$. Note how the coefficients of the transformations are the same in both directions. Equation (3.17) allows us to construct the amplitude for a sequential two-body decay chain with the helicity basis and transform it to the canonical basis in order to get access to the angular-momentum content of each decay node.

3.3. Amplitude construction

Finally, we have the mechanisms to construct the amplitude of a general multi-body decay of the form $0 \to 1\,2\,\ldots\,n$. As discussed in Chapter 2, the transition amplitude for a transition from initial state $|\psi_{\text{in}}\rangle$ to final state $|\psi_{\text{out}}\rangle$ comes from the application of the transition operator $\mathbf{T}$ to these states using Equation (2.3). In the helicity formalism, the initial state is a single spin state $|\psi_{\text{in}}\rangle = |p_0, j_0, \lambda_0\rangle$. The final state is a set of n spin states that are assumed to be stable and non-interacting and can therefore be written as a tensor product,

$$|\psi_{\text{out}}\rangle \ = \ |p_1, j_1, \lambda_1\rangle \times |p_2, j_2, \lambda_2\rangle \times \ldots |p_n, j_n, \lambda_n\rangle \ .$$

To construct the amplitude explicitly, we exploit the fact that conservation laws (in particular angular momentum conservation at each vertex) allow us to decompose the full multi-body transition into a product of two-body transitions, analogous to internal lines in a Feynman diagram, connected through sums over intermediate helicities. This leads naturally to the sequential-decay parametrisation of the amplitude.

Sequential-decay parametrisation

The strategy for formulating the transition amplitude is to split up the decay into two-body decays, which we can describe using our knowledge of relativistic, coupled two-particle spin states. This approach is often referred to as the *isobar model*, but a better term would be **sequential-decay** [150; 149; 151; 155] or *cascade-decay* [156] parametrisation.

In the sequential-decay parametrisation, we consider the multi-body decay $0 \to 1\,2\,\ldots\,n$ to proceed via a cascade of two-body decays. For $n > 2$, there are multiple two-body decay topologies that can be constructed. We call such a topology a **chain**, or path, and denote it with a Fraktur index $\mathfrak{c}$ [156]. A decay chain can be visualised in the form of a directed graph of two-body decay nodes labelled by κ. The naming scheme for each node κ is shown in Figure 3.6: the digits of the final-state particles to which it decays are listed and grouped with parentheses depending on how it decays further. In each node κ, the parent state is denoted by κ_0, and the two decay products by κ_1 and κ_2. Decay chains can be named with the same labeling scheme by taking the label of its first decay node. For instance, the left chain in Figure 3.6 would be labelled 1(2(34)).

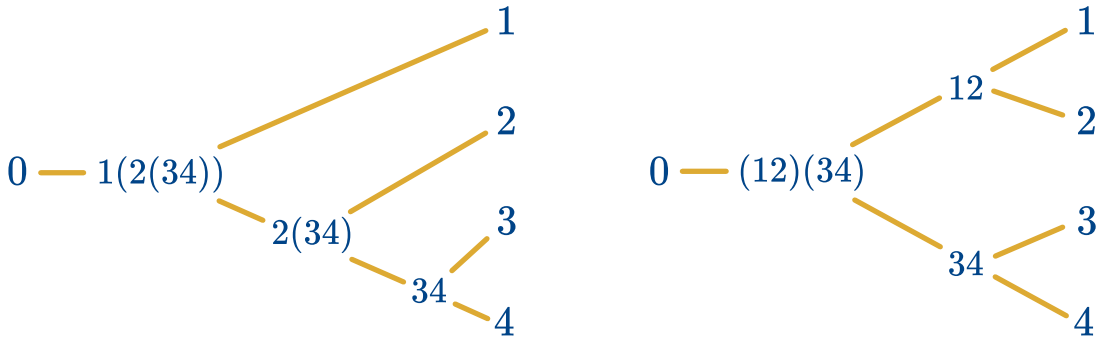


Figure 3.6. Vertex notations for two decay chains, 1(2(34)) and (12)(34), in a sequential four-body decay. Topologies and nodes are labelled by grouping the children of each node in parentheses.

Parametrisations of the scattering matrix can be found with path-integral methods [100, §9.2]. The decay topologies in the sequential-decay parametrisation can be seen as a 'discrete' path integral from the initial to the final state once we assume that each of the two-body decay nodes in the chain takes place independently. The **helicity amplitude for chain** $\mathfrak{c}$ can therefore be constructed as the product of the helicity amplitudes of all its two-body decay nodes κ, that is

$$\begin{aligned} A^{\mathfrak{c}}_{\lambda_0,\{\lambda\}}(\tau_{\mathfrak{c}}) \;=\; & \sum_{\{\lambda_{\text{int.}}\}} \prod_{\kappa}^{n-1} D^{j_{\kappa_0}\,*}_{\lambda_{\kappa_0},\lambda_{\kappa_1}-\lambda_{\kappa_2}}(\Omega^{\mathfrak{c}}_{\kappa}) \\ & \times\; X_{\kappa}\big(\mu_{\kappa_0}\big)\; H^{\kappa_0\to\kappa_1,\kappa_2}_{\lambda_{\kappa_1},\lambda_{\kappa_2}}\big(\mu_{\kappa_0},\mu_{\kappa_1},\mu_{\kappa_2}\big)\;. \end{aligned} \tag{3.18}$$

with $\tau_{\mathfrak{c}}$ the set of relevant kinematic variables for the chain $\mathfrak{c}$, such as the helicity angles $\Omega^{\mathfrak{c}}_{\kappa}$ for node κ and the invariant masses $\mu_{\kappa_0}, \mu_{\kappa_1}, \mu_{\kappa_2}$ of its parent and decay products. We have denoted the set of allowed helicities of the final-state particles with $\{\lambda\} = \{\lambda_1, \dots, \lambda_n\}$ and those of the intermediate spin states with $\{\lambda_{\text{int.}}\}$. The latter helicities cannot be observed and have to be summed over, but this has to be done carefully, as some decay products of each node appear as the parent of the next node. Also note that $\lambda_{\kappa_0}, \lambda_{\kappa_1}, \lambda_{\kappa_2}$ can be any of λ_0, $\{\lambda\}$, or $\{\lambda_{\text{int.}}\}$, depending on which node κ is considered. The Wigner D-functions originate from the rotations over the helicity angle $\Omega^{\mathfrak{c}}_{\kappa} = (\theta^{\mathfrak{c}}_{\kappa}, \phi^{\mathfrak{c}}_{\kappa})$ applied before boosting into the rest frame of the next helicity state in the decay chain. The function X_{κ} parametrises the lineshape of the decaying resonance κ_0 (propagator) in the two-body decay κ, whereas the function $H^{\kappa_0 \to \kappa_1, \kappa_2}_{\lambda_{\kappa_1}, \lambda_{\kappa_2}}$ parametrises its vertex (see Section 2.4). The angular-momentum content can be applied by inserting the transformation of Equation (3.17).

Each symbol μ_i indicates the *invariant* mass of state i. The initial and final state are considered stable, that is, their masses are not variables of the dynamics functions. For example, in single-channel analyses, $X_k\big(\mu_{\kappa_0}\big)$ is often parametrised as Equation (2.25). For non-zero angular momenta, the energy-dependent width depends on the masses of the decay products, $\mu_{\kappa_1}, \mu_{\kappa_2}$, but these are taken to be constant. From a computational point of view, one has to assume these masses to be fixed if one implements a proper analytic continuation, otherwise the dispersion integral Equation (2.38) has to be solved for each each iteration during a fit.

The Wigner D-functions in Equation (3.18) originate from the rotations over the helicity angle $\Omega^{\mathfrak{c}}_{\kappa}$ applied before boosting into the rest frame of the next helicity state in the decay chain. As explained in Equation (3.16), their lower index $\lambda_{\kappa 1} - \lambda_{\kappa 2}$ encodes this helicity-difference phase, which ensures the correct angular-momentum coupling at each decay node.

Wigner rotations

In a sequential-decay amplitude that contains topologically distinct decay chains, the same set of final-state helicities can be reached through different sequences of Lorentz boosts and rotations. Because the spin quantisation axis that defines the helicity of each final-state particle (Figure 3.2) is determined by the specific boost sequence of its decay chain, the helicity labels from different chains correspond to different physical axes. In this case, the **total transition amplitude** $A_{\lambda_0,\{\lambda\}}$ cannot be obtained by simply summing all chain amplitudes $A^{\mathfrak{c}}_{\lambda_0,\{\lambda\}}$ of Equation (3.18). To combine them meaningfully, the final-state helicities from each chain must first be rotated into a common reference frame, ensuring that all spin quantisation axes are aligned.

The mismatch between the different chains can be corrected by an alignment factor $W^{\mathfrak{c}(\mathfrak{o})}_{\{\lambda'\},\{\lambda\}}(\tau_{\mathfrak{c}} \mid \tau_{\mathfrak{o}})$ for each chain $\mathfrak{c}$ and a selected **reference chain** $\mathfrak{o}$. The total transition amplitude for all chains is then given by

$$A_{\lambda_0,\{\lambda\}}(\tau_{\mathfrak{o}}) \;=\; \sum_{\{\lambda'\}} \sum_{\mathfrak{c}} A^{\mathfrak{c}}_{\lambda_0,\{\lambda'\}}(\tau_{\mathfrak{c}})\, W^{\mathfrak{c}(\mathfrak{o})}_{\{\lambda'\},\{\lambda\}}(\tau_{\mathfrak{c}} \mid \tau_{\mathfrak{o}})\,. \tag{3.19}$$

The alignment factor is a product of (inverse) **Wigner rotations** in spin space, one for each final-state particle i with spin j_i,

$$W^{\mathfrak{c}(\mathfrak{o})}_{\{\lambda'\},\{\lambda\}}(\tau_{\mathfrak{c}} \mid \tau_{\mathfrak{o}}) \;=\; \prod_{i=1}^{n} D^{j_i\,*}_{\lambda'_i \lambda_i}\left(\phi^i_{\mathfrak{c}(\mathfrak{o})}, \theta^i_{\mathfrak{c}(\mathfrak{o})}, \chi^i_{\mathfrak{c}(\mathfrak{o})}\right)\,. \tag{3.20}$$

Here, $\phi^i_{\mathfrak{c}(\mathfrak{o})}, \theta^i_{\mathfrak{c}(\mathfrak{o})}, \chi^i_{\mathfrak{c}(\mathfrak{o})}$ are the Euler angles (Figure 3.1) describing the spatial rotation $\boldsymbol{R}^i_{\mathfrak{c}(\mathfrak{o})}$ that connects the two helicity frames for particle i. Computing these angles is a non-trivial task that generally requires a numerical procedure [156]. The key idea is that the product of Lorentz transformations from the initial state 0 to a final-state particle i along chain $\mathfrak{c}$, followed by the inverse transformations along reference chain $\mathfrak{o}$, cancels the net boost and yields a pure (Wigner) rotation,

$$\boldsymbol{R}^i_{\mathfrak{c}(\mathfrak{o})} = \left(\prod_a^{0\to i} \boldsymbol{\Lambda}^{\mathfrak{c}}_{a\leftarrow a_{\text{next}}}\right)^{-1} \times \left(\prod_b^{0\to i} \boldsymbol{\Lambda}^{\mathfrak{o}}_{b\leftarrow b_{\text{next}}}\right) . \tag{3.21}$$

The Euler angles of $\boldsymbol{R}^i_{\mathfrak{c}(\mathfrak{o})}$ can be extracted by applying Equation (3.21) to a four-momentum in a suitably chosen coordinate system. Although $\boldsymbol{R}^i_{\mathfrak{c}(\mathfrak{o})}$ is an element of the Lorentz group $\mathrm{SO}^+(1,3)$, its action on spin states is represented in the spinor space $\mathrm{SL}(2,\mathbb{C})$, the double cover of $\mathrm{SO}^+(1,3)$ (see Section 3.2). In this representation, a spatial rotation of 2π changes the sign of a half-integer spin state, so spinors exhibit a 4π periodicity. This phase behaviour must be tracked when combining amplitudes from different decay chains. To determine whether a sign flip occurs, the same transformation sequence can be applied with Lorentz transformation matrices in $\mathrm{SL}(2,\mathbb{C})$ to a reference Dirac spinor, and the resulting phase compared to the original.

Figure 3.7 illustrates this procedure for final-state particle 1 in two topologically distinct four-body decay chains, $\mathfrak{c} = (1(23))4$ and $\mathfrak{o} = ((12)3)4$, showing how the Lorentz transformations differ and how their combination yields the required Wigner rotations for the spin-alignment factor.

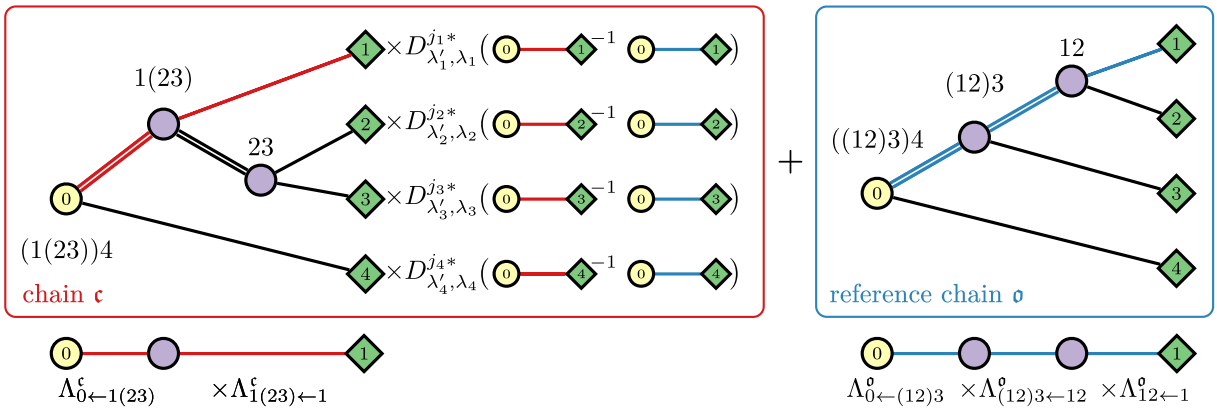


Figure 3.7. Sketch of how aligning a four-body decay chain $\mathfrak{c} = (1(23))4$ (left) to a reference chain $\mathfrak{o} = ((12)3)4$ (right) leads to four Wigner D rotation functions. The symbols in the arguments of the Wigner D-functions indicate how angles for these rotations are obtained: boost and rotate along the reference chain from the initial state (yellow) to each final state (green), then back along chain $\mathfrak{c}$. The lower row shows the sequence of Lorentz transformations that lead to the rest frame of particle 1 and that is coloured in the chains. Adapted from [156, Fig. 2].

Differential decay rate

Finally, the total transition amplitudes from Equation (3.19) can be combined into an expression for the differential decay rate of a general multi-body decay. This is the analogue of Equation (2.4)

for differential cross sections, but now applied to a decay process with an arbitrary number of final-state particles. To write down this expression, we must first identify the relevant degrees of freedom in the differential. The decaying particle 0 is produced with a certain spatial orientation, which is typically described relative to the production plane in terms of Euler angles $\Omega = (\phi, \theta, \chi)$. In addition, the n-body decay process itself has $k = 3n - 7$ internal degrees of freedom: $3n$ for each three-momentum, minus 4 for the overall energy-momentum conservation, and minus 3 for the spatial orientation of the decay. These k variables are collected in the set τ, and they govern the internal dynamics of the decay, that is, they only affect the helicity amplitudes.

In contrast to the internal decay variables τ, the angles $\Omega = (\phi, \theta, \chi)$ encode the orientation of an external spin quantisation axis of the decaying particle 0 with respect to the production frame. An example of this is discussed in Section 7.1, particularly Figure 7.2. Since the production process may leave 0 in a polarised state, we must account for the statistical mixture of spin projections m_0 that it can occupy. This information is encoded in the elements $\rho_{m_0' m_0}$ of **spin-density matrix $\boldsymbol{\rho}$**, which describe the coherence between spin projections m_0 and m_0' along a quantisation axis defined by the production mechanism.

However, the helicity formalism expresses decay amplitudes in terms of spin projections λ_0 along the momentum direction of the decaying particle. This direction is determined by the first decay node of reference chain $\mathfrak{o}$ in Equation (3.19), that is, by the two states to which particle 0 decays in chain $\mathfrak{o}$. To relate the two frames, we rotate the total transition amplitudes $A_{\lambda_0,\{\lambda\}}$ *back* over the angles Ω using Equation (3.3), giving

$$\widetilde{A}_{m_0,\{\lambda\}}(\Omega, \tau) \;=\; \sum_{\lambda_0} D^{j_0\,*}_{m_0\lambda_0}(\phi, \theta, \chi)\, A_{\lambda_0,\{\lambda\}}(\tau) \tag{3.22}$$

The resulting sum over matrix elements $D^{j_0\,*}_{m_0\lambda_0}$ implement the change of spin basis from the decay frame to the production frame, allowing us to coherently sum over all possible spin orientations in the decay process. This gives the **differential decay rate**,

$$\begin{aligned}
\frac{\mathrm{d}\Gamma}{\mathrm{d}^3\Omega\,\mathrm{d}^k\tau} \;&\cong\; \sum_{\{\lambda\}} \sum_{m_0, m_0'} \widetilde{A}^*_{m_0',\{\lambda\}}(\Omega, \tau)\, \rho_{m_0' m_0}\, \widetilde{A}_{m_0,\{\lambda\}}(\Omega, \tau) \\
&= \sum_{\{\lambda\}} \sum_{m_0, m_0'} \rho_{m_0' m_0} \\
&\qquad \times \sum_{\lambda_0} D^{j_0\,*}_{m_0\lambda_0}(\phi, \theta, \chi)\, A_{\lambda_0,\{\lambda\}}(\tau) \\
&\qquad \times \sum_{\lambda_0'} D^{j_0}_{m_0'\lambda_0'}(\phi, \theta, \chi)\, A^*_{\lambda_0',\{\lambda\}}(\tau)\,.
\end{aligned} \tag{3.23}$$

The spin-density matrix $\rho_{m_0' m_0}$ is a Hermitian matrix of dimension $2j_0 + 1$. Its diagonal elements give the distribution of the spin projections along a chosen quantisation axis in the production plane and its off-diagonal elements encode any coherences between them. As discussed in Section 2.5, the differential cross section serves as an intensity function that describes the measured data distributions over each degree of freedom. Since this function is fit to data using maximum likelihood techniques, any normalisation factors can be left out (indicated by $\cong$) in practical applications.

In many analyses, the polarisation of the decaying particle is not of interest and the Euler angles ϕ, θ, χ in Equation (3.23) are effectively integrated out. This results in an **unpolarised** intensity function

$$
\begin{aligned}
I_0(\tau) &= \int_0^{2\pi} \mathrm{d}\phi \int_0^{\pi} \sin\theta \, \mathrm{d}\theta \int_0^{2\pi} \mathrm{d}\chi \, \frac{\mathrm{d}\Gamma}{\mathrm{d}^3\Omega \, \mathrm{d}^k\tau} \\
&\cong \sum_{\{\lambda\}} \sum_{m_0, m_0'} \rho_{m_0' m_0} \delta_{m_0 m_0'} \sum_{\lambda_0, \lambda_0'} \delta_{\lambda_0 \lambda_0'} A_{\lambda_0, \{\lambda\}}(\tau) \, A^*_{\lambda_0', \{\lambda\}}(\tau) \\
&= \sum_{\{\lambda\}} \mathrm{Tr}\ \rho \sum_{\lambda_0} A_{\lambda_0, \{\lambda\}}(\tau) \, A^*_{\lambda_0', \{\lambda\}}(\tau) \\
&\cong \sum_{\lambda_0} \sum_{\{\lambda\}} \left| A_{\lambda_0, \{\lambda\}}(\tau) \right|^2 .
\end{aligned}
\tag{3.24}
$$

Here, we have used the fact that the Wigner D-functions form a set of orthogonal functions over the Euler angles, meaning that

$$
\begin{aligned}
&\int_0^{2\pi} \mathrm{d}\alpha \int_0^{\pi} \sin\beta \, \mathrm{d}\beta \int_0^{2\pi} \mathrm{d}\gamma \, D^j_{mn}(\alpha, \beta, \gamma) \, D^{j'\,*}_{m'n'}(\alpha, \beta, \gamma) \\
&\qquad = \frac{8\pi^2}{2j+1} \, \delta_{jj'} \, \delta_{mm'} \, \delta_{nn'} .
\end{aligned}
$$

Dalitz-Plot Decomposition

In the case of a three-body decay $0 \to 123$, the three-momenta of final-state particles $\vec{p}_1, \vec{p}_2, \vec{p}_3$ in the rest frame of the decaying particle span a unique plane, since momentum conservation enforces $\vec{p}_1 + \vec{p}_2 + \vec{p}_3 = 0$ (see Figure 3.8). We refer to the plane spanned by these momenta as the **Dalitz plane**, as it fully captures the internal kinematic degrees of freedom of the decay, analogous to the Dalitz plot. As before, the remaining degrees of freedom are the Euler angles $\Omega = (\phi, \theta, \chi)$ that specify the orientation of the Dalitz plane with respect to the production frame.

Since all kinematically relevant variables lie within the Dalitz plane, the amplitude construction only requires Wigner d-functions to describe rotations within the plane, without an 'off-plane' azimuthal component. The **Dalitz-Plot Decomposition** (DPD) method exploits this fact to derive analytic expressions for the remaining polar rotation angles in terms of Mandelstam invariants $\sigma_1 = (p_2 + p_3)^2$, $\sigma_2 = (p_3 + p_1)^2$, $\sigma_3 = (p_1 + p_2)^2$ and rest masses $\mu_0, \mu_1, \mu_2, \mu_3$ [157]. Using Equation (2.8), that leaves only $k = 2$ degrees of freedom within the Dalitz plane, as expected.

Sequential three-body decays have only one type of topological structure with two decay nodes: particle 0 decays to a spectator k and an intermediate state i, j (the **production node**), and then the intermediate state decays to final-state particles i, j (the **decay node**). Such a decay chain $\mathfrak{c} = (ij)k$ is often referred to as a **subsystem** and is sometimes simply labelled by index k. The decay chain with regard to which we align the other decay chains (chain $\mathfrak{o}$ in Equation (3.19)) is called the **reference subsystem**, and we label it with index l of its spectator. With this notation, the three-body analogue of Equation (3.19) is the **Dalitz-plot function**,

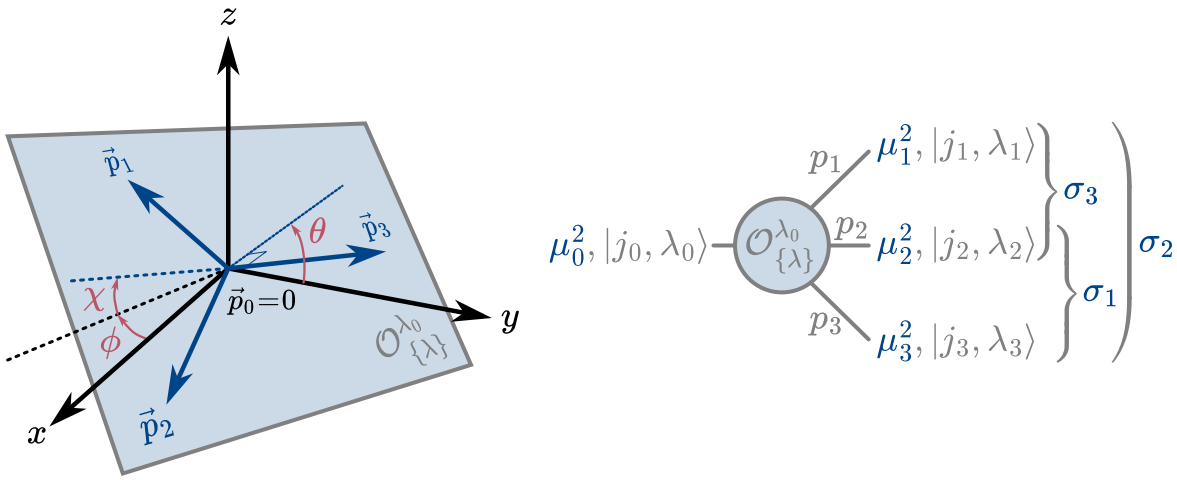


Figure 3.8. In the rest frame of decaying particle 0 in a three-body decay, the momenta $\vec{p}_1, \vec{p}_2, \vec{p}_3$ of the final state span the Dalitz plane (left). The orientation of this plane is specified by the Euler angles ϕ, θ, χ. Within the plane (right), the Mandelstam invariants $\sigma_1, \sigma_2, \sigma_3$ and the rest masses $\mu_0, \mu_1, \mu_2, \mu_3$ fully describe the internal dynamics of Dalitz-plot function $\mathcal{O}^{\lambda_0}_{\{\lambda\}}$.

$$
\begin{aligned}
\mathcal{O}^{\lambda_0}_{\{\lambda\}}(\tau) \;=\; & \sum_{\{\lambda'\}} \sum_{(ij)k} \sum_{s}^{(ij)\to i,j} \sum_{\lambda_s} n_{j_0}\, n_s \\
& \times\; d^{j_0}_{\lambda_0,\lambda_s-\lambda'_k}(\hat{\theta}_{k(l)})\, d^{s}_{\lambda_s,\lambda'_i-\lambda'_j}(\theta_{ij}) \\
& \times\; H^{0\to(ij),k}_{\lambda_s,\lambda'_k}(\mu_0^2,\sigma_k,\mu_k^2)\, X_s(\sigma_k)\, H^{(ij)\to i,j}_{\lambda'_i,\lambda'_j}(\sigma_k,\mu_i^2,\mu_j^2) \\
& \times\; d^{j_1}_{\lambda'_1\lambda_1}(\zeta^1_{k(0)})\, d^{j_2}_{\lambda'_2\lambda_2}(\zeta^2_{k(0)})\, d^{j_3}_{\lambda'_3\lambda_3}(\zeta^3_{k(0)})\,.
\end{aligned}
\tag{3.25}
$$

Here, $(ij)k$ sums over all decay chains (23)1, (31)2, and (12)3, label s indicates all expected intermediate spins in the two-body decay $(ij) \to i, j$, and λ_s are the helicities of that intermediate spin state. The normalisation factors $n_{j_0} = \sqrt{2j_0+1}$ and $n_s = \sqrt{2s+1}$ are similar to those appearing Equation (3.15). Two Wigner d-functions follow: one for the production node with angle $\hat{\theta}_{k(l)}$ measured in the rest frame of 0, and one for the decay node with scattering angle θ_{ij} measured in the rest frame of $(ij) \to i, j$. The vertex functions H and propagator function X_s were introduced in Equation (3.18). Finally, the three Wigner d-functions with angles $\zeta^i_{k(0)}$ describe the Wigner rotations required to align the three decay chains $(ij)k$ with regard to the production frame.

Since $\hat{\theta}_{k(l)} = \zeta^0_{k(l)}$, the Wigner d for the production node can be interpreted as a Wigner rotation. In addition, for computations it is more efficient to compute the Wigner rotation angles with regard to the reference subsystem l, meaning that $\zeta^i_{k(0)} \to \zeta^i_{k(l)}$, so that the Wigner rotations in the amplitude for that subsystem vanish (this choice affects the definition of the Euler angles Ω of the decay plane orientation in Equation (3.23)). This results in the alternative form of Equation (3.25),

$$
\begin{aligned}
\mathcal{O}^{\lambda_0}_{\{\lambda\}}(\tau) \;=\; & \sum_{\lambda_0'} \sum_{\{\lambda'\}} \sum_{(ij)k} \sum_{s}^{(ij)\to i,j} \sum_{\lambda_s} n_{j_0}\, n_s \\
& \times\; \delta_{\lambda_0',\lambda_s-\lambda_k'}\, d^{s}_{\lambda_s,\lambda_i'-\lambda_j'}(\theta_{ij}) \\
& \times\; H^{0\to(ij),k}_{\lambda_s,\lambda_k'}(\mu_0^2,\sigma_k,\mu_k^2)\, X_s(\sigma_k)\, H^{(ij)\to i,j}_{\lambda_i',\lambda_j'}(\sigma_k,\mu_i^2,\mu_j^2) \\
& \times\; d^{j_0}_{\lambda_0\lambda_0'}(\zeta^0_{k(l)})\, d^{j_1}_{\lambda_1'\lambda_1}(\zeta^1_{k(l)})\, d^{j_2}_{\lambda_2'\lambda_2}(\zeta^2_{k(l)})\, d^{j_3}_{\lambda_3'\lambda_3}(\zeta^3_{k(l)})\,.
\end{aligned}
\tag{3.26}
$$

An unpolarised intensity function for the decay $J/\psi \to \bar{p}K^0_S\Sigma^+$ with only two subsystems, 2 and 3 as reference subsystem, is provided in Equation (7.20). The model also uses Equation (3.17) to transform the chain amplitudes to the canonical basis and get access to the angular momentum content of each decay node (Equation (7.21)).

The angles θ_{ij}, $\hat{\theta}_{k(l)} = \zeta^0_{k(l)}$, and $\zeta^i_{k(0)}$ can be computed analytically from Mandelstam invariants $\sigma_1, \sigma_2, \sigma_3$ and the rest masses $\mu_0, \mu_1, \mu_2, \mu_3$ of the initial- and final-state particles. The expressions are given in Appendix A of [157]. Taking $(ij)k \in \{\,(23)1, (31)2, (12)3\,\}$, the three relevant Wigner rotation angles in Equation (3.25) and Equation (3.26) are

$$
\begin{aligned}
\cos\theta_{ij} \;&=\; \frac{2\sigma_k(\sigma_j-\mu_k^2-\mu_i^2)-(\sigma_k+\mu_i^2-\mu_j^2)(\mu_0^2-\sigma_k-\mu_k^2)}{\lambda^{1/2}(\mu_0^2,\mu_k^2,\sigma_k)\,\lambda^{1/2}(\sigma_k,\mu_i^2,\mu_j^2)} \\
\cos\hat{\theta}_{i(j)} \;&=\; \frac{(\mu_0^2+\mu_i^2-\sigma_i)(\mu_0^2+\mu_j^2-\sigma_j)-2\mu_0^2(\sigma_k-\mu_i^2-\mu_j^2)}{\lambda^{1/2}(\mu_0^2,\mu_j^2,\sigma_j)\,\lambda^{1/2}(\mu_0^2,\sigma_i,\mu_i^2)} \\
\cos\zeta^i_{k(0)} \;&=\; \frac{2\mu_i^2(\sigma_j-\mu_0^2-\mu_j^2)+(\mu_0^2+\mu_i^2-\sigma_i)(\sigma_k-\mu_i^2-\mu_j^2)}{\lambda^{1/2}(\mu_0^2,\mu_i^2,\sigma_i)\,\lambda^{1/2}(\sigma_k,\mu_i^2,\mu_j^2)}\,.
\end{aligned}
\tag{3.27}
$$

In addition, $\zeta^i_{k(i)} = \zeta^i_{k(0)}$, and $\hat{\theta}_{l(k)}$ can be found by replacing it with $\hat{\theta}_{k(l)}$ at the cost of a sign flip, following $d^j_{\lambda\lambda'}(\hat{\theta}_{l(k)}) = (-1)^{\lambda-\lambda'} d^j_{\lambda\lambda'}(\hat{\theta}_{k(l)})$. Angles with equal indices are zero, so $\theta_{kk} = 0$, $\hat{\theta}_{k(k)} = 0$, and $\zeta^i_{k(k)} = 0$. In some decays, this fact can be used to simplify the expression for the Dalitz-plot function with a suitable choice of reference subsystem.

4 Computational techniques

In the past decades, scattering experiments have produced steadily growing datasets. These larger samples allow us to apply more sophisticated parametrisations without overfitting. However, this progress comes at a cost: the complexity of these amplitude models as well as the increased size of the data samples requires significantly more computational power.

This chapter explores some of the modern techniques for high-performance computing that have become available in recent years and that have been employed for this work within the ComPWA project (Chapter 5). It first outlines how advancements in computing hardware have driven the popularity of array-oriented programming libraries that are nowadays commonplace in data analysis and Machine Learning. It then examines how Computer Algebra Systems can simplify the formulation of amplitude models and generate code for fast numerical computations over large data arrays. The chapter concludes by showing how a dynamic programming language like Python enables the generation of publication-ready content directly from the analysis codebase, with symbolic expressions naturally complementing this approach.

4.1. Array-oriented programming

Traditionally, most analyses in high-energy physics and hadron spectroscopy have been written in performance-focused languages like C++ and Fortran. These languages dominated because they offered fine-grained control and precision at a time when particle physics began producing large volumes of collision data [158; 159], see Figure 4.1. In many cases, C++ is a good choice for high-performance computing, as it allows for low-level control over memory and processor usage and can therefore implement any kind of logic efficiently. Beyond raw performance, C++ offered stronger support for abstraction and modularity through features like namespaces and object-oriented design [160], which made it well suited for the growing complexity of large-scale physics software. Since the early 2000s, however, there has been a broader shift across scientific computing towards higher-level languages such as Python, which trade low-level control for better readability, flexibility, and easier tool integration.

Code complexity and hardware diversity

The challenge with these performant, low-level languages is that it often results in complex, verbose code. This is especially problematic in analysis code that repetitively performs similar mathematical operations over columnar data. The mathematical operations of interest have to be implemented through iterative loops over the data (often called “scalar-based loops”), which makes it is hard to recognise the physics formulas that are being implemented [162], although modern language extensions partly alleviate these issues [163]. This makes it particularly challenging to express amplitude models in these traditional high-performance languages.

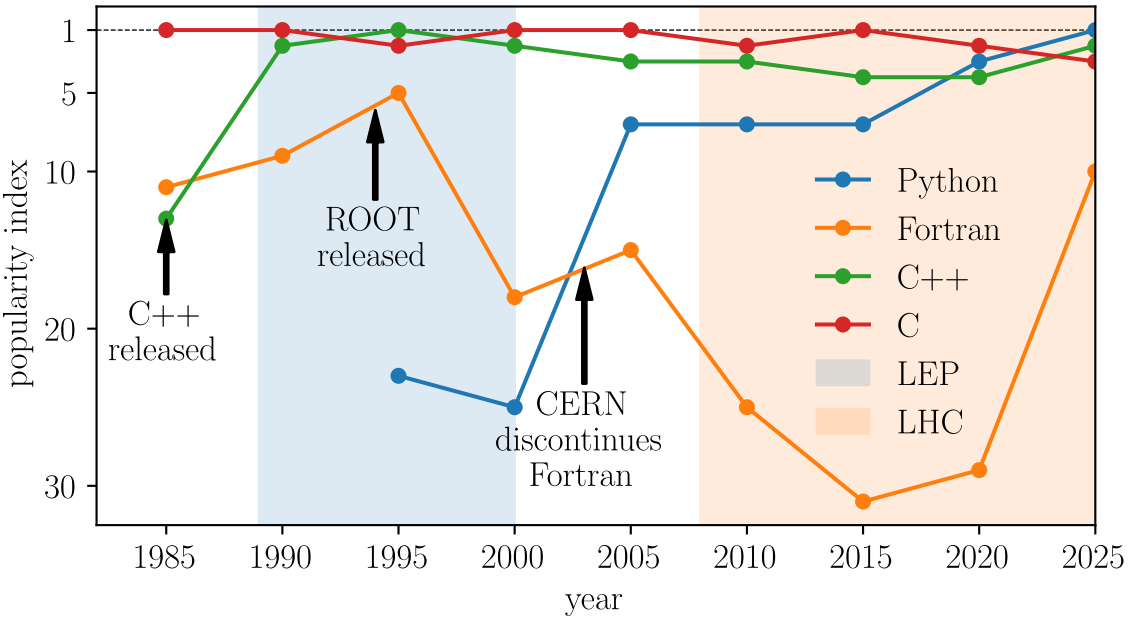


Figure 4.1. Popularity of C, C++, Python, and Fortran since 1985 according to TIOBE [161]. The lower the index, the higher the popularity of the language.

An additional difficulty in scientific computing is that low-level implementations in languages like C++ or Fortran are difficult for the broader programming community to maintain across the diversity of modern hardware architectures. In recent decades, chips are hitting physical limits, which prompts hardware manufacturers to develop a great variety of chips and accelerators that often have to be parallelised to achieve higher performance. With the advent of GPUs, TPUs, and other specialised accelerators, the need for code that can run efficiently on various types of parallelised hardware has therefore become increasingly important.

Furthermore, the widespread use of mobile devices that prioritise energy efficiency has driven the broad adoption of Reduced Instruction Set Computer (RISC) chip architectures like ARM. These chips feature a distinct memory hierarchy and instruction set compared to the more prevalent Complex Instruction Set Computer (CISC) architectures like x86-64 that are found in desktop and server CPUs. ARM chips are therefore increasingly adopted in cloud computing services as well, as they are more energy-efficient and can be scaled to larger numbers of cores at lower cost.

Amplitude analysis almost always involves those repetitive, scalar-based loops, where the amplitude model is evaluated uniformly over each event in a dataset. This process is inherently parallel, as the evaluation of the amplitude model for one event does not rely on the outcome of any other – a scenario often described as "embarrassingly parallel" [164, p. 14]. The parallel nature of amplitude model evaluations therefore allows us to distribute the data sample over the memory of different computational devices and performing the scalar evaluations concurrently over different processors, which is ideal in an age of multi-core CPUs and hardware accelerators like GPUs.

Arrays and code conciseness

Parallelism naturally lends itself to **array programming**. Instead of treating each event as an isolated scalar computation executed in a loop, array-oriented programming conceptualises the entire dataset as a single, contiguous entity – an "**array**". Arrays can be multidimensional and are therefore also called **tensors**. Each dimension corresponds to an **axis**, a direction along which the array can be indexed, and together the axis lengths define the **shape** of the array. For example, a 4×3 array has shape `(4, 3)`, meaning two axes (rows and columns), and is therefore a rank-2 tensor. As can be seen in Figure 4.2, an array consists of a continuous block of memory (typically of a uniform type), along with metadata for interpreting the data stored in it. The metadata provides a pointer to the start of the memory block, a data type, a shape, and strides that define how to move through the memory block to access the elements [165; 166]. As such, arrays are essentially recipes for nested loops over memory blocks that represent multidimensional data structures, which array libraries then optimise to improve memory locality and parallel execution [167].

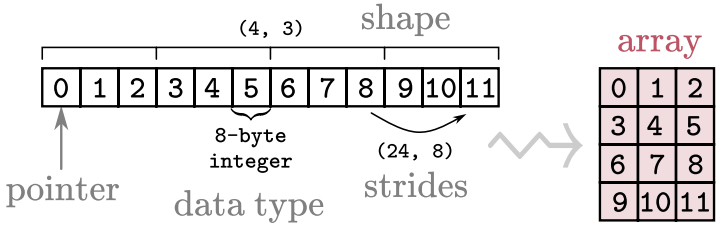


Figure 4.2. Memory layout of a rank-2 array of shape 4×3 with 12 integers. The strides and data type tell the machine how to iterate over the memory block, while the shape tells the size of the array in each dimension. Adapted from [166, Fig. 1a].

In array-oriented programming languages or libraries, the operations are defined in terms of these arrays, which allows for more concise and readable code. For example, in the following code, we see how an array of 12 integers, organised in a 4×3 shape, are defined as a single variable `x`. In this chapter, the return value of the last line in each code cell is shown as output directly beneath it.

```
import numpy as np

size = 12
x = np.arange(size).reshape((4, 3))
x
```

```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

The code to generate this array is not important for this example: the array could be also be generated with a random-number generator or imported from a data file. Rather, the fact that the array is represented as a single variable allows for concise and readable code and highlights

array-oriented thinking [168]. For example, a simple polynomial evaluation $x^2 - 7x + 3$ can be written as:

```
x**2 - 7*x + 3
```

```
array([[ 3, -3, -7],
       [-9, -9, -7],
       [-3,  3, 11],
       [21, 33, 47]])
```

Behind the scenes, the array library takes care of the nested loops over the memory block. Operations that are less arithmetic can be performed through functions provided by the library:

```
np.sqrt(x)
```

```
array([[0.        , 1.        , 1.41421356],
       [1.73205081, 2.        , 2.23606798],
       [2.44948974, 2.64575131, 2.82842712],
       [3.        , 3.16227766, 3.31662479]])
```

This also allows us to perform operations that pertain to certain axes of the array. For instance, the following example computes the mean along the first axis (that is, across the rows), using the `mean()` method of the array.

```
x.mean(axis=0)
```

```
array([4.5, 5.5, 6.5])
```

Array libraries can also automatically handle arrays of different dimension or different shapes. This is called **broadcasting** and allows us to perform operations on array objects without specifying the operation in a loop. In many cases, the array library will further optimise the computation by reusing memory and avoiding unnecessary copies. A simple example of broadcasting is multiplying an array times a scalar, as happens in the examples above. Broadcasting also happens when the value type of the array has to be modified, for instance when multiplying an integer times a float.

The following example uses broadcasting to add two arrays of different shapes. The first array is a rank-1 array of shape `(3,)` and the second is a rank-2 array of shape `(4, 3)`. The library automatically expands the first array to match the shape of the second array, so that the operation can be carried out element-wise. In this case, the rank-1 array `a` is turned into a rank-2 array of shape `(3, 1)` (column vector) and multiplied with each row of `b`, resulting in a 3×4 array where every row of `b` is scaled by the corresponding entry of `a`.

```
a = np.array([1, 2, 3])
b = np.array([[4, 5, 6, 7], [8, 9, 10, 11], [12, 13, 14, 15]])
a[np.newaxis].T * b
```

```
array([[ 4,  5,  6,  7],
       [16, 18, 20, 22],
       [36, 39, 42, 45]])
```

Broadcasting is powerful because it scales seamlessly to arrays of much larger size and applies the same operation just as well to higher-rank tensors. This flexibility makes array code agnostic to the shape of the data, removing the need to manually reorganise loops when the input changes.

Vectorisation and parallelism

Crucially, a key benefit of writing concise array-based code, is that it naturally exposes opportunities for parallel execution and other optimisations. In other words, the same design choices that improve readability also make it easier for the array library to take advantage of modern hardware. The Python code we have seen is the set of instructions that we want to apply to each element, or axis of elements, in the array. Under the hood, the computational loops over the array are automatically implemented in low-level, highly optimised routines (often written in C, C++, or Fortran). To make these routines readily available, Python extension modules are typically distributed as pre-compiled binaries ("wheels"), so the user does not need to compile them locally.

This approach of delegating repetitive work to specialised routines is generally referred to as **vectorisation** and opens the door to several optimisations. The core idea of vectorisation is that computational operations can be applied to all elements of an array without caring about their order, if computation on each element is independent of previous iterations. This allows the array library to reorder operations, execute them in parallel across multiple cores or vector units, and optimise memory access patterns.

A simple example of parallel processing is Single Instruction, Multiple Data (SIMD), which applies the same instruction to multiple data elements simultaneously on a CPU [169]. Modern CPUs have larger registers that can hold several data elements at once, and SIMD instructions operate on all elements in the register simultaneously (see Figure 4.3). In low-level languages, one typically relies on the compiler to detect which parts of the code can be vectorised (this behaviour can be tuned with compiler optimisation flags such as `-O3`). By contrast, array-oriented code naturally assumes uniform input data, making it much easier to translate the code directly into SIMD instructions. To take advantage of this, array libraries include optimisations for different vector instruction sets, such as SSE and AVX-512. Advanced Vector Extensions (AVX) like these are a CPU feature that implement the SIMD principle by allowing registers to hold more data elements at once and by introducing operations that act on all of them simultaneously, so arithmetic (such as addition or multiplication) can be carried out on many numbers in parallel. In practice, these libraries automatically detect the available CPU features and use the most efficient instructions, so the same code runs optimally on different machines.

Memory locality

The parallel nature of arrays also allows for multi-core and multi-threaded computations. In this case, the array library distributes work across several cores that share the same memory space. This is conceptually similar to SIMD in that the same operation is applied to many elements, but here it is carried out on different cores or threads rather than within a single register.

Additionally, these lower-level routines can also optimise memory access patterns, since arrays are typically stored in a compact, contiguous block of memory. Because modern processors are

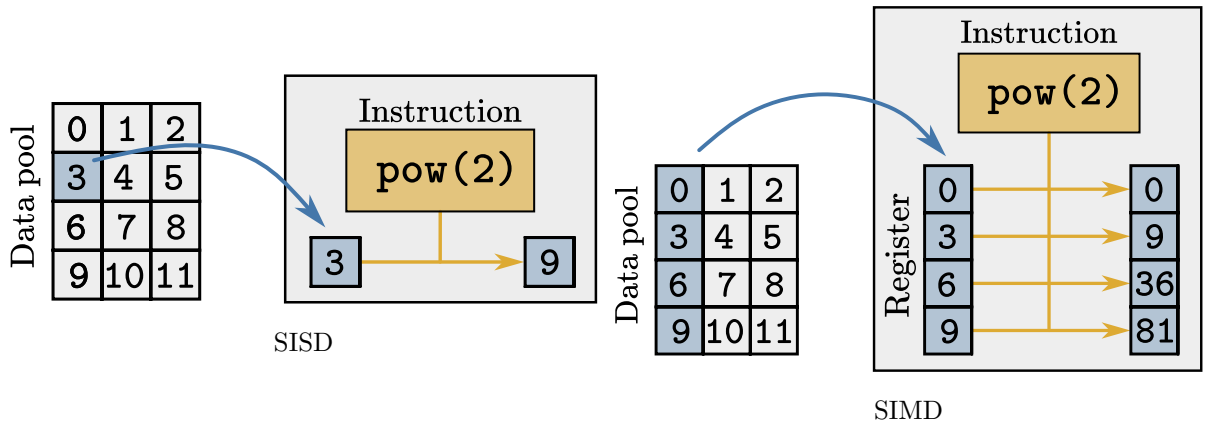


Figure 4.3. Comparison between Single Instruction, Single Data (SISD) and Single Instruction, Multiple Data (SIMD) processing. In SISD, a single instruction is executed on a single data element, while in SIMD, a single instruction is executed on multiple data elements.

built around deeply layered memory hierarchies, their performance strongly depends on keeping data close to the processing units in fast caches rather than repeatedly fetching it from slower main memory, which lies relatively far away from the processor on the scale of microchips. If data is stored in a continuous block and accessed in order, processors can cache it more efficiently. This is especially important when arrays are large, as memory bandwidth often becomes the limiting factor in numerical calculations. By contrast, naive implementations of scalar loops may scatter data in ways that are harder to optimise. The **locality** of arrays therefore inherently results in better computational performance.

Accelerated linear algebra

Many of these optimisations come together in operations that involve linear algebra operations or reductions on certain axes of the array. These operations are usually outsourced to highly optimised libraries like BLAS (Basic Linear Algebra Subprograms, see [170]) and LAPACK (Linear Algebra PACKage, see [171]). BLAS and LAPACK not only make use of SIMD and parallelism, but also optimise usage of the memory hierarchy and multi-level caches (L1, L2, and L3) while evaluating operations over subsets (tiles) of the arrays. Similarly to SIMD, these linear algebra operations also usually include hand-tuned assembly code for the specific hardware on which the compiled code runs. Some library offer additional optimisations like constant folding and algebraic simplifications.

Hardware accelerators

Finally, array-oriented libraries can offload computations to domain-specific accelerator hardware, such as GPUs. Traditionally, code is compiled for Central Processing Units (CPUs), which are ideal for general-purpose, logical tasks. CPUs have a powerful Control Unit (CU), Arithmetic Logic Units (ALUs), and hierarchical caches that can run complex, branching code efficiently (see Figure 4.4). By contrast Graphics Processing Units (GPUs) are designed for parallel, arithmetic-heavy tasks. GPUs contain thousands of smaller CUDA cores (NVIDIA) or Stream Processors

(AMD) that are optimised for running the same instruction on many uniform data elements in parallel. Both CPUs and GPUs get their input data from Random Access Memory (RAM), but GPUs use this random access to feed many cores in parallel, while CPUs use it to feed its ALUs cores with complex, branching code.

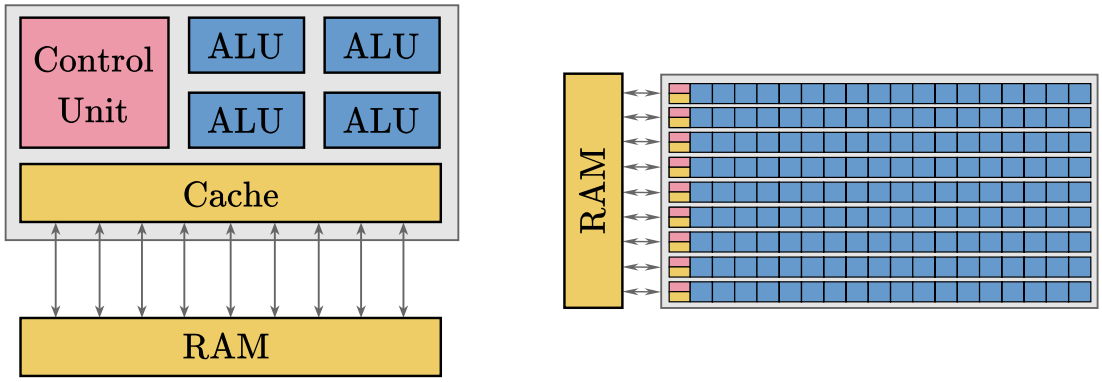


Figure 4.4. Architecture of a Central Processing Unit (CPU) on the left and a Graphics Processing Unit (GPU) on the right. Both have one or more multi-purpose Control Units (pink), Arithmetic Logic Units (blue), and caches (yellow).

The growing interest and investment into Machine Learning (ML) and training of Large Language Models (LLMs) has driven the development of even more specialised hardware accelerators. Prompted by the high operational costs of running hardware clusters, companies are starting to design their own in-house chip architectures that are designed to handle specific ML workloads. An early example of this are Google's Tensor Processing Units (TPUs), which are specialised in running matrix multiplications at even higher throughput at the cost of lower machine precision [172]. The key point is that, while high-performance computing is moving towards a highly heterogeneous landscape of chip architectures, array libraries are continuously adapting to these changes.

NumPy itself can delegate certain operations to GPUs through add-on libraries (e.g., CuPy), while other popular array-oriented frameworks (like TensorFlow, PyTorch, or JAX) are built from the ground up to harness GPU and TPU acceleration. Using such devices directly is far from trivial, as they have their own separate memory, data transfers are relatively slow, and one normally has to work with vendor-specific programming models such as CUDA or OpenCL. Array libraries take care of these complexities: thanks to the uniform array abstraction, switching between CPU and GPU can be as simple as moving your data to a GPU-backed array. They also integrate with specialised backends, ranging from device-specific libraries like cuBLAS to compiler frameworks such as XLA (Accelerated Linear Algebra), which generate optimised kernels for the selected hardware. The code itself remains unchanged, since these libraries still provide high-level instructions that operate on arrays.

JIT-compilation and lazy evaluation

In some libraries, further optimisations come from lazy evaluation – deferring certain calculations until they are strictly necessary – so that the library can fuse multiple operations into a single, optimised kernel call (**kernel fusion**). This approach reduces the overhead of running many small, separate operations and having to store the intermediate results in memory. Although

NumPy does not provide lazy evaluation, frameworks such as JAX or Dask do [173]. The core idea is consistent across these tools: writing code in terms of entire arrays makes it easier for the underlying engine to schedule, parallelise, and optimise the calculations.

A natural complement to lazy evaluation is Just-in-Time (JIT) compilation, which analyses high-level operations right before execution and compiles them into efficient, low-level machine code. In the context of array-oriented programming, this approach allows the compiler to specialise the resulting code to the exact shapes and data types of the input arrays. By knowing at compile time how large an array is and what data type it holds, JIT compilers can prepare and optimise memory access patterns based on the resources that the hardware has available. JIT compilers designed for hardware accelerators also use the incoming array information to minimise copy operations between the CPU and the accelerator, which can be a significant bottleneck in heterogeneous computations.

JIT compilation is even useful for operations that are not easily vectorised, such as if-else logic. Vectorised programs and hardware accelerators are inefficient at handling conditional logic, because they require all elements of the array to be processed in the same way. A common solution is to use **masks**, which are arrays of boolean values that select elements from another array. This produces a new view of the data block in which only the relevant elements remain, so the processor can run straight through without branching. In this way, conditional logic is turned into parallel operations, where masking operations replace explicit if-else statements – a style known as **branchless programming**. Such branchless formulations are a must for fast numerical performance on SIMD architectures and hardware accelerators, where conditional logic reduces the number of cores that can be used in parallel [166].

Automatic differentiation

Another benefit of array-oriented programming is that it naturally lends itself to automatic differentiation (AD, or "autodiff"). This technique makes it possible to compute the partial derivative of a function with respect to the input variables of interest. This is particularly useful in Machine Learning, where the derivative of the loss function with respect to the model parameters is used to update the model parameters through gradient descent.

The same technique can be applied in physics analyses, where we usually minimise the log-likelihood function for the amplitude model over large, unbinned data sets using a gradient-descent algorithm. Traditionally, the gradient at the current evaluation point is computed numerically by evaluating the function at multiple points around the current evaluation point and taking the finite difference [143]. These finite-difference approximations are easier to implement, but become computationally expensive and numerically unstable when the number of parameters or the size of the data sample grows [174]. So far it has however been the only option in amplitude analysis, as it is almost impossible to manually derive and implement the gradient of chained, nested, and non-linear amplitude formulas.

The core idea of AD is to break down the function into a series of elementary operations, for which the derivative is known. By applying the chain rule, the derivative of the function can be computed by combining the derivatives of the elementary operations. In array-oriented programming, the function is already expressed in terms of operations to uniform arrays, which makes it easier to break down the function into elementary operations. This is why many Machine Learning frameworks, like TensorFlow, PyTorch, and JAX, provide automatic differentiation as a core feature.

Automatic differentiation uses the chain rule of calculus to compute exact derivatives without symbolic manipulation in two distinct modes:

- **Forward-mode AD** propagates derivatives with respect to each function parameter "forward" through the computational graph. It does so by defining a dual space algebra, with the first element in each dual number being the function value and the second element being the derivative, and propagating the operation of each node in the function forward through the graph. This makes forward-mode efficient when the number of inputs (parameters) is small compared to the number of outputs.
- **Reverse-mode AD** (often referred to as "backpropagation" in the context of neural networks) instead starts from the outputs and propagates derivatives "backward" through the graph. At each node, the chain rule distributes the contribution of the node to all of its inputs, so that in the end the derivative of the output with respect to every input parameter has been accumulated. This approach is efficient when the number of inputs is large but the number of outputs is small, as in amplitude analysis or maximum-likelihood fits, where we optimise a single scalar function (the likelihood) with respect to many parameters. Most machine-learning libraries, including TensorFlow, JAX, and PyTorch, implement reverse-mode AD under the hood to allow efficient gradient computation at scale.

Figure 4.5 illustrates the difference between reverse-mode and forward-mode automatic differentiation for a composite function $f(g(x,y))$. In reverse-mode (left), the computation first evaluates the inner function $g(x,y)$ and then applies f to its result. The derivative $\frac{\partial f}{\partial g}$ is propagated backward to g, which is then multiplied by the partial derivatives $\frac{\partial g}{\partial x}$ and $\frac{\partial g}{\partial y}$ to yield $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$. In forward-mode (right), the procedure begins by attaching derivative information to each input (dual space), such as $\frac{\partial x}{\partial x} = 1$ and $\frac{\partial y}{\partial x} = 0$. This information is propagated forward through $g(x,y)$, producing both g and $\frac{\partial g}{\partial x}$. Finally, applying f to this dual information yields not only $f(g(x,y))$ but also the derivative $\frac{\partial f}{\partial x}$.

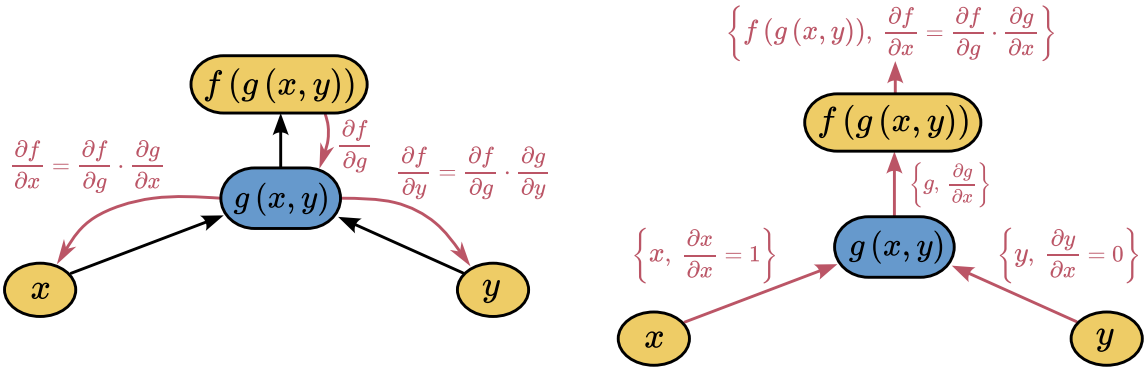


Figure 4.5. Comparison between reverse-mode AD (left) and forward-mode AD (right).

Employing AD in an array-oriented framework involves writing a function that accepts arrays and returns one or more arrays or scalars. The framework "traces" the function – either during execution (eager tracing) or via a separate compilation stage – and constructs a computational graph that records how each output depends on each input. It then applies the chain rule to this graph to compute partial derivatives.

As an example, the code below derives the gradient of

$$f(x, y, z) = x^2 \sin(y) + \log(1 + z^2)/y$$

using JAX. The function is defined in Python with arithmetic from the `jax.numpy` module, which allows JAX to trace the operations and build a computational graph. Calling `jax.jacrev` with respect to the arguments (x, y) (indices `0` and `1`) applies reverse-mode AD to evaluate the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$. The example also illustrates broadcasting: x is given as a one-dimensional array, while y is a scalar, so JAX computes the gradient element-wise. The resulting `f_gradient` function returns two arrays: a 4×4 Jacobian with respect to the input array x, since each of the four outputs depends on every element of x, and a length-4 array with derivatives with respect to the scalar y. The output is returned as JAX arrays, so every operation remains part of the computational graph and can be extended with further calculations, including gradients, for efficient execution on the chosen hardware accelerator.

```
import jax
import jax.numpy as jnp

def f(x, y, z):
    return x**2 * jnp.sin(y) + jnp.log(1 + z**2) / y

f_gradient = jax.jacrev(f, argnums=(0, 1))
x_array = jnp.array([-1.8, 3.1, -0.7, 2.5])
y_value = 0.5
z_value = -1.5
f_gradient(x_array, y_value, z_value)
```

```
(Array([[-1.726,  0.   , -0.   ,  0.   ],
        [-0.   ,  2.972, -0.   ,  0.   ],
        [-0.   ,  0.   , -0.671,  0.   ],
        [-0.   ,  0.   , -0.   ,  2.397]], dtype=float64),
 Array([-1.871,  3.719, -4.285,  0.77 ], dtype=float64, weak_type=True))
```

What makes this particularly powerful for high-performance computing is that AD benefits from all of the hardware optimisations that have been described so far. The parallelisation schemes apply equally to the constructed computational graph of the gradient function and JIT-compilation can fuse both the forward pass (function evaluation) and backward pass (derivative evaluation) into efficient kernels. Frameworks like JAX further optimise the derivative pass by reusing intermediate results from the forward pass.

In summary, abstracting away the scalar-based loops in terms of arrays allows us to outsource the work of writing machine-specific, optimised code to the array library. Driven by the surge in data analysis, Machine Learning, and training of Large Language Models (LLMs), arrays have become the standard data container for parallelisation and hardware accelerators. Array programming has become a popular paradigm in many programming languages. The popularity of array-oriented programming is therefore not limited to Python, but is also present in other languages like Julia and even C++, where libraries like JuliaDiff, SIMD.jl, Eigen, and Armadillo

provide similar functionality. The challenge for the developers of all these libraries and languages is to tailor compilation to the plethora of hardware architectures that are becoming available. This device abstraction makes it possible to keep the array-based code that formulates physics models unaffected, as the libraries that power the computations adapt to the advances in computational techniques.

4.2. Computer Algebra Systems

The previous section introduced array-oriented programming as a means to express calculations concisely, while leveraging modern hardware for performance gains. Array-oriented code already bridges the gap between theory and code quite well, but it turns out that symbolic representations offer a structured intermediate step before numerical evaluation. This is especially true in the context of amplitude analysis, where the mathematical expressions can be complicated and involve many parameters.

Computer Algebra Systems (CAS) allow us to define and manipulate amplitude models symbolically before implementing them in numerical computations. This "**CAS-assisted model building**" enables us to explore the mathematical structure of the amplitude model, simplify sub-expressions, and ensure correctness before transitioning to efficient numerical evaluation. This section explores these ideas using SymPy as a CAS, and using JAX and NumPy as numerical backends. As we will see in Chapter 7, the combination of SymPy with JAX turned out to work best for large-scale amplitude analyses.

Mathematical expressions as trees

Computer Algebra Systems emerged in the 1960s as tools for performing symbolic mathematical manipulations, mostly driven by the needs of physics and engineering [175]. Early systems like MACSYMA [176] and REDUCE pioneered symbolic computation by introducing algebraic manipulation techniques that enabled automated mathematical reasoning. Mathematica [177] later expanded these capabilities with an interactive, user-friendly environment and a broad range of built-in symbolic and numerical functions, making symbolic computation more accessible to scientists and engineers. In the world of Python, SymPy [178] has become the most popular CAS that provides similar interactive experience since the advent of Jupyter notebooks (see Section 4.3 and the examples below).

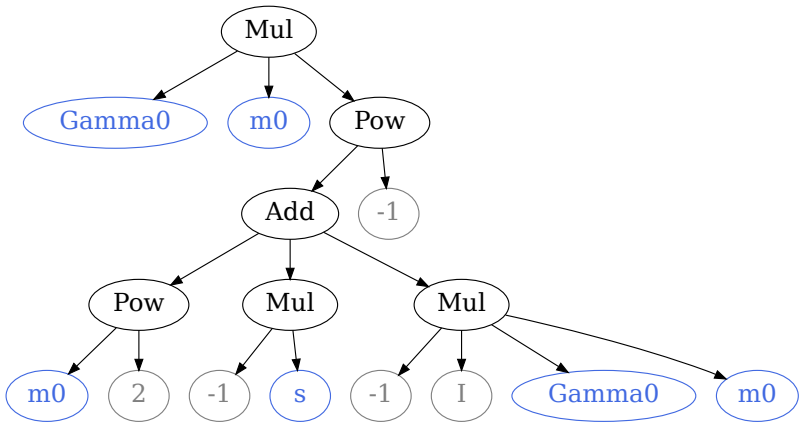
The key idea of a CAS is that mathematical expressions are treated as **expression trees**, where each node represents an operation (e.g., addition, multiplication, differentiation), and the leaves represent variables or constants. This representation allows CAS to perform exact algebraic manipulations by applying standard transformations to the leaves and branches of the tree. Crucially, such CAS operations are exact, meaning that they preserve the mathematical structure of the expression without introducing numerical errors. This is in contrast to numerical computations, which are subject to floating-point approximations and rounding errors.

The following example demonstrates how SymPy builds a simple Breit–Wigner expression as an expression tree. It shows the Python code, the automatically rendered formula, and the corresponding expression tree. The fundamental building blocks are symbols (s, m_0, Γ_0, indicated in blue), constants ($-1, 2, i$, indicated in grey), and algebraic operations (multiplication, addition, and exponentiation, indicated in black). The graph is a rendering of the hierarchical structure with which SymPy inherently represents the expression. Note that SymPy reorders commutative

terms alphabetically, which is why both the expression tree and the LaTeX rendering differ slightly from the code.

```
import sympy as sp
s, mass, width = sp.symbols("s m0 Gamma0")
expression = mass * width / (mass**2 - s - sp.I * width * mass)
expression
```

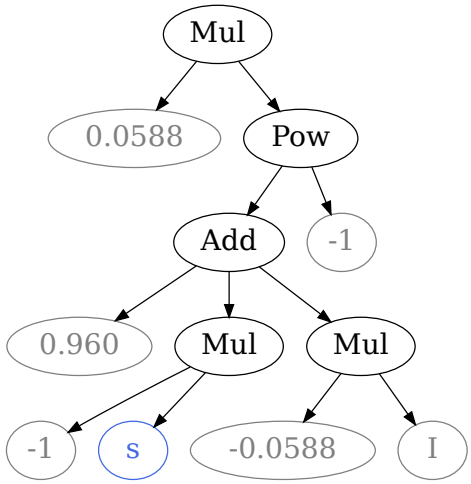
$$\frac{\Gamma_0 m_0}{-i\Gamma_0 m_0 + m_0^2 - s}$$



When substituting symbols with numerical values, the CAS applies exact, algebraic transformations to the relevant branches in the expression tree. In the following example, the symbols m_0 and Γ_0 in the Breit–Wigner expression are replaced with numerical values. Even though these values are floating-point numbers, the result of the substitution is still an exact symbolic expression, so that floating-point errors are avoided. This is achieved through the arbitrary-precision floating-point library `mpmath` [179]. Importantly, the expression tree corresponding to the result of the substitution has a simpler structure with fewer mathematical nodes.

```
expression.subs({mass: 0.980, width: 0.06})
```

$$\frac{0.0588}{-s + 0.9604 - 0.0588i}$$



Beyond algebraic computations, most CAS systems also provide tensor algebras, symbolic integration techniques, series expansions, and differential equation solvers. Within hadron spectroscopy, CAS systems like Mathematica and Maple are therefore popular with theorists in particular for verifying analytical properties of functions, such as identifying singularities in resonance models or ensuring symmetry constraints in decay amplitudes.

As a simple example, the following code snippet demonstrates how a CAS can derive the simple Breit–Wigner expression from above directly from a one-dimensional K-matrix. First, we construct the symbolic $\boldsymbol{T}$-matrix in terms of the $\boldsymbol{K}$-matrix, using symbolic placeholders that represent the full matrices rather than individual entries. This way, the algebraic relation $\boldsymbol{T} = \boldsymbol{K}\,(1 - i\,\boldsymbol{K})^{-1}$ is preserved at the matrix level and would also work for larger matrices [180]. In the final line, we expand the result explicitly and extract the element `T[0, 0]` (denoted mathematically as T_{11}).

```
dimension = 1
I = sp.Identity(n=dimension)
K = sp.MatrixSymbol("K", m=dimension, n=dimension)
T = K / (I - sp.I * K)
T.as_explicit()[0, 0]
```

$$\frac{K_{0,0}}{-iK_{0,0} + 1}$$

Next, the $\boldsymbol{K}$-matrix elements are parametrised in terms of a single pole position m_0 and a coupling g, and then substituted into the $\boldsymbol{T}$-matrix expression. Since m_0 and s were defined earlier, the only new symbol here is the coupling g, which must be declared first.

```
g = sp.Symbol("g")
K_parametrisation = {
    K[0, 0]: g**2 / (mass**2 - s),
}
T_expr = T.as_explicit().subs(K_parametrisation)[0, 0]
T_expr
```

$$\frac{g^2}{(m_0^2 - s)\left(-\frac{ig^2}{m_0^2 - s} + 1\right)}$$

Finally, we can derive that the this single-pole, single-channel expression for the $\boldsymbol{T}$-matrix is the same as the Breit–Wigner expression constructed earlier, by setting the coupling to $g^2 = m_0\Gamma_0$. In the expression rendering, the minus sign has been pulled before the fraction, but this is mathematically equivalent.

```
BW_expr = T_expr.subs(g**2, mass * width)
BW_expr.simplify()
```

$$-\frac{\Gamma_0 m_0}{i\Gamma_0 m_0 - m_0^2 + s}$$

The CAS simplifies the expression to a more compact form, which is not only more recognisable, but also more computationally numerically, as its expression tree contains fewer mathematical operations.

Code generation, lambdification, and numerical computations

While a CAS excels at symbolic manipulation and algebraic exactness, it is not designed for high-performance numerical computations. To bridge this gap, CAS-generated expressions can be converted into efficient numerical code. In SymPy, this is known as **code generation**. The CAS goes over the expression tree and serialises the mathematical nodes to numerical code. Subtrees are naturally grouped with the required brackets and parentheses and common subexpressions are identified and reused.

SymPy provides code generation for a variety of programming languages, including Python, C++, Fortran, Julia, and Rust, but the printing mechanism can be extended for any custom language or format [181]. The following demonstrates how the Breit–Wigner expression translates to some of the popular languages.

```
print(sp.pycode(expression))
```

```
Gamma0*m0/(-1j*Gamma0*m0 + m0**2 - s)
```

```
print(sp.cxxcode(expression, standard="c++17"))
```

```
Gamma0*m0/(-I*Gamma0*m0 + std::pow(m0, 2) - s)
```

```
print(sp.fcode(expression).strip())
```

```
Gamma0*m0/(-cmplx(0,1)*Gamma0*m0 + m0**2 - s)
```

```
print(sp.julia_code(expression))
```

```
Gamma0 .* m0 ./ (-im * Gamma0 .* m0 + m0 .^ 2 - s)
```

```
print(sp.rust_code(expression))
```

```
Gamma0*m0/(-I*Gamma0*m0 + m0.powi(2) - s)
```

These example code snippets are not necessarily evaluatable, because the variables that appear, are undefined. SymPy therefore offers a more extensive code generation mechanism, that embeds the generated code in numerical functions. SymPy refers to this process as **lambdification**, because the result is not just source code, but a Python function object. This object is anonymous (commonly called a "lambda" function, after λ calculus) and can be assigned to a variable and called like any regular function. The following code snippet demonstrates how the earlier Breit–Wigner expression is lambdified to a JAX function, with the symbols s, m_0, and Γ_0 marked as arguments to the generated function. The names of the function arguments come from the symbol names (e.g. `Gamma0`), not the names of the corresponding symbol objects (e.g. `width`).

```
func = sp.lambdify(args=(s, mass, width), expr=expression, modules="jax")
```

```
def _lambdifygenerated(s, m0, Gamma0):
    return Gamma0*m0/(-1j*Gamma0*m0 + m0**2 - s)
```

The generated function has the same array-oriented form as the Python function defined in the first JAX example and can be used to evaluate the Breit–Wigner expression over large data arrays. As a simple example, the snippet below defines an array of 10 000 real values between 0 and 10 as input for the variable s, while m_0 and Γ_0 are fixed to scalar constants. The function then returns an array of 10 000 complex values, corresponding to the Breit–Wigner evaluated at each s.

```
s_array = jnp.linspace(0, 10, num=10_000)
func(s_array, m0=0.98, Gamma0=0.06)
```

```
Array([ 0.061+3.734e-03j, ..., -0.007+4.231e-05j], dtype=complex128)
```

The lambdified function is also compatible with JAX's JIT compilation and automatic differentiation, enabling both efficient numerical evaluation and gradient computation. In the example below, we use reverse-mode AD to compute derivatives of the Breit–Wigner function with respect to m_0 and Γ_0 (arguments 1 and 2). Because the Breit–Wigner is complex-valued, the function is marked as holomorphic so that JAX applies the correct rules for complex differentiation. The resulting gradient function is then JIT-compiled and evaluated at the same point in parameter space.

```
gradient = jax.jacrev(func, argnums=(1, 2), holomorphic=True)
jit_gradient = jax.jit(gradient)
jit_gradient(s_array, 0.98+0j, 0.06+0j)
```

```
(Array([-0.062-0.008j, ..., -0.008+0.j   ], dtype=complex128, weak_type=True),
 Array([ 1.009+0.124j, ..., -0.108+0.001j], dtype=complex128, weak_type=True))
```

Since the numerical function is generated from a symbolic expression, its gradient can also be obtained analytically. This provides a straightforward way to check the correctness of the numerical gradient, and may in some cases even yield higher numerical precision thanks to algebraic simplifications. The example below shows this for the Breit–Wigner expression, where the gradients with respect to m_0 and Γ_0 are derived and simplified symbolically.

```
derivative_m0 = sp.diff(expression, mass).simplify()
```

$$\frac{\Gamma_0\left(-i\Gamma_0 m_0 + m_0^2 + m_0\left(i\Gamma_0 - 2m_0\right) - s\right)}{\left(i\Gamma_0 m_0 - m_0^2 + s\right)^2}$$

```
derivative_w0 = sp.diff(expression, width).simplify()
```

$$\frac{m_0\left(m_0^2 - s\right)}{\left(i\Gamma_0 m_0 - m_0^2 + s\right)^2}$$

These symbolic derivatives can then again be lambdified into numerical functions, making them directly usable in the same JAX-based workflow as the original expression. When evaluated over the 10 000 values of s, the analytically derived gradients indeed agree with those obtained through automatic differentiation.

```
df_dm = sp.lambdify(args=(s, mass, width), expr=derivative_m0, modules="jax")
df_dw = sp.lambdify(args=(s, mass, width), expr=derivative_w0, modules="jax")
df_dm(s_array, 0.98, 0.06), df_dw(s_array, 0.98, 0.06)
```

```
(Array([-0.062-0.008j, ..., -0.008+0.j   ], dtype=complex128),
 Array([ 1.009+0.124j, ..., -0.108+0.001j], dtype=complex128))
```

As we have seen in Chapter 2, amplitude models involve complex analytical functions composed of polynomials, exponentials, and trigonometric terms, particularly those arising from Wigner D-functions in relativistic spin formalisms. Writing these models directly in numerical code is cumbersome and can obscure their structure. The examples from the previous sections demonstrate how a CAS can be used to simplify the formulation of these models, inspect the structure of the expressions, and generate efficient numerical code for large-scale computations. In Chapter 5, we will see that this also applies for the more complex models that are used in amplitude analysis and that the remarkable performance of array libraries is still available through code generation. In addition, SymPy can simplify trigonometric identities that emerge in angular momentum recoupling and verify that model expressions are correctly formulated before they are transformed into efficient computational implementations. Similar "symbolic-numeric approaches" have been applied using SymForce [182] and Julia [183], but we are not aware of similar approaches in high-energy physics or hadron spectroscopy.

For large expression trees, CAS-assisted model effectively serves as an additional layer of compiler optimisations. This is comparable to XLA in JAX, which is a hardware-aware optimisation. A CAS is specialised in algebraic simplification, which may result in more efficient numerical function trees and higher numerical precision. An example of this is Linnea [184], where a CAS is used to even further speed up linear algebra kernels like BLAS. The combination of CAS and array-oriented programming therefore provides a powerful toolset for high-performance computing in amplitude analysis.

4.3. Self-documenting workflow

The developments in hadron spectroscopy of the passed few decades have reached a point where transparency, adaptability, and verifiability are not just desirable, but necessary. The increasing complexity of amplitude models, combined with the availability of larger datasets from high-luminosity experiments, requires workflows that ensure that the results are reliable and reproducible, while remaining flexible enough to accommodate new theoretical insights, as well as new computational techniques. The combination of CAS-assisted model building and array programming can make it easier to construct workflows that not only speed up computations, but also document their assumptions, derivations, and numerical implementations in a self-explanatory manner.

Reproducibility

The "replicability crisis", which was initially identified in medical and social sciences [185], has increasingly affected the natural sciences, including physics and computational research. Studies have shown that a significant portion of published findings cannot be reproduced due to inadequate

documentation, inconsistencies in data handling, or reliance on proprietary software [186]. Scientific disciplines that rely on computational methods, may seem less susceptible to these issues due to the deterministic nature of numerical calculations. High-performance computing, however, introduces additional complexities, such as parallelism, hardware dependencies, and software versioning, that can lead to subtle errors and discrepancies [187]. In the case of amplitude analysis, there is the additional challenge of quickly integrating theoretical insights with new experimental data, which often leads to ad-hoc workflows that are difficult to reproduce.

In industry, practices such as automated testing, continuous integration, and environment management have long been the standard for software development, ensuring that code remains stable and deployable across different platforms. The scientific community has started to adopt many of these software development practices [188; 189], such as version control and automated testing through continuous integration (CI). It has also led to initiatives for archiving code in a way that makes it permanently accessible, either simply through Git repository hosting services like GitHub, or through dedicated platforms like Zenodo [190]. More initiatives that specifically tailor to particle physics are being developed [191; 192].

In addition, containerisation technologies such as Docker, Singularity, and Podman have become popular for creating reproducible computational environments. These tools allow researchers to package their code, dependencies, and environment settings into a single, shareable image that can be run on any platform that supports the container runtime [193]. However, containers have the limitation that they are not always compatible with high-performance computing environments, because their virtualisation overhead can slow down computations. In addition, containers have a tendency to become black boxes that cannot be extended later.

An alternative solution in the form of **lock files** may be more suitable for high-performance computing environments, as they capture the specific versions of all declared dependencies used in an analysis. Examples are `Cargo.lock` in Rust, `Manifest.toml` in Julia, `conan.lock` for C++, and `pylock.lock` for Python. Lock files make it possible to reconstruct the same software environment across machines and over time, though not necessarily with identical binaries [194]. They therefore support reproducible software outputs, even if compilers or system libraries differ. At the level of the analysis results, numerical outcomes may vary slightly due to differences in hardware and build configurations. Such variations are usually well within statistical or systematic uncertainties and can themselves be studied across different devices. In this way, lock files provide a reproducible and extensible basis for software, whereas containers go further by freezing entire toolchains. The goal is not strict bitwise equivalence of the software stack, but reproducibility of analyses up to the usual numerical variations.

In Chapter 5 and Chapter 7, we see how lock files are used to offer reproducibility and extensibility when using array-oriented programming libraries with CAS-assisted model building in the Python ecosystem.

Methodological clarity

Ultimately, scientists need to share their computational work in publishable form. To a large degree, such publications still come out in their traditional, static form that can be printed, even if they are distributed electronically. Even though there are initiatives to provide a more interactive, interlinked, and dynamic form of publication [195; 196], there remains a need to clearly document the computational methods that have led to the published analysis results.

One of the first notable attempts to strengthen the link between code implementation and the eventual publication is the concept of “literate programming” [197]. Literate programming

encourages the programmer to write code in a way that is readable and understandable to humans, rather than to the computer. The code is interspersed with explanatory text, which can be formatted in a way that is suitable for publication. The code and text are then "tangled" into a machine-readable form that can be executed by the computer.

While "literate programming" has had its influence on analyses that were written in Pascal and C, its application remained limited to the translation of code to LaTeX-only documents that are not necessarily publication-ready. In addition, these tools have the limitation that they are not interactive and do not allow for the exploration of the code in a dynamic way. In parallel, Org-mode for Emacs [198] extended the paradigm to a multi-language environment, in which source code, text, and publication-quality output could be woven into an executable document. The increasing popularity of dynamic languages like Python and R led to more widely adopted interactive tools such as R Markdown, IPython [199; 200], and Jupyter notebooks [201], which allow the interleaving of code, text, and visualisations in a single document that can be executed and inspected interactively.

These tools starkly remind of CAS systems like Mathematica, Sage, Maxima, and Maple, but offer the additional benefit that they are open-source and can be extended with custom code in popular programming languages. This more interactive form of computational science has been termed "literate computing" [202], because they open the door to more extensive software development behind the interactive environment. As an example, a typical workflow in our experience starts with trying out new ideas in a Jupyter notebook using SymPy, NumPy, and JAX, and then extracting common functionality into an underlying Python package as the implementations become more advanced. The eventual analysis still naturally evolves along in the notebooks as the underlying codebase matures and can be tested using the continuous integration techniques as mentioned earlier.

Motivated by a surge in data science and Machine Learning, Jupyter notebooks are also gaining traction in data-rich, high-performance environments [203]. Jupyter notebooks separate the user interface from the computation: the notebook interface runs in a local client (such as a browser), while a dedicated kernel can run on a remote server or high-performance cluster. This makes it possible to carry out heavy computations on powerful hardware, while the user still interacts with them through the same notebook interface. For instance, CERN has developed the SWAN service that allows users to run Jupyter notebooks on the CERN computing grid. And with tools like Dask, computations over array-based data can be scaled and parallelised behind the scenes, while the user still interacts with the data in a familiar way. The web-based server approach also makes it easier to collaborate on the same analysis, whether through shared kernels or through platforms like Binder, which encourages transparency and reproducibility. More broadly, this separation of interface and computation aligns with a larger shift in industry toward cloud computing, where machine learning and other high-performance workloads are executed on specialised remote infrastructure rather than on local machines.

Finally, the original spirit of "literate programming" is still alive in these more interactive and scaled environments. Tools and platforms like Quarto, the Executable Book Project, Curvenote, and Stencila can render collections of analysis notebooks to a variety of formats, including LaTeX, Portable Document Format (PDF), and static HTML pages [204; 205]. The continuous deployment of these notebooks to a publishable form encourages to continuously consider reproducibility and methodological clarity from the start of the analysis, rather than as an afterthought [187].

The combination of CAS-assisted model building and array-oriented programming in an interactive Jupyter environment therefore provides a powerful toolset for high-performance

computing in amplitude analysis. Symbolic amplitude models can directly be inspected in both the interactive environment as well as the static rendering. High-performance computations can be scaled and parallelised behind the scenes through array computations, and common implementations can be extracted and published in the form of a Python package. The resulting workflow is not only efficient, but also transparent, reproducible, and methodologically clear.

Knowledge transfer and academic continuity

The hadron spectroscopy community covers a wide range of expertise across theory and experiment, which often makes it difficult to communicate results. When amplitude analyses are performed with CAS-assisted model building and published using the tools described above, they not only present the implemented mathematics in a readable mathematical form, but also enable the reader to inspect the implementation if formulations are discipline-specific. Offering ways to launch the code repository and the Jupyter notebooks in a cloud environment, like Binder, can further offer the possibility to try out and modify the analysis to gain a better understanding of the results. This is especially important when communicating results between theorists and experimentalists and between different experimental collaborations.

In addition, code generation using CAS models can be used to serialise models to human-readable, hierarchical formats like JSON that can be parsed into other programming languages or analysis frameworks. This not only allows for cross-verifiability, but also makes it possible to pick up analyses at a much later stage, when implementations are no longer supported or when the original authors are no longer active in the field. The JSON format can also be used to store the model in a database, which can be queried and analysed in a more structured way. This can be particularly useful for large-scale analyses that involve many different models, or for analyses that are performed over long periods of time and need to be revisited and updated.

The latter point touches on a more societal aspect of the field. The hadron spectroscopy community is relatively small and, like many academic disciplines, has a high turnover rate, with many researchers transitioning to other fields after a few years. This makes it essential to document and preserve knowledge systematically to prevent the loss of expertise. The combination of CAS-assisted model building and ability to continuously document analyses can help to bridge the gap between theoretical understanding and practical implementation, and allow future generations of researchers to pick up and extend existing analyses.

This shift is also more generally relevant in the continuous cycle of education and knowledge transfer, where students, graduates, and postdocs must continually familiarise themselves with the computational and theoretical intricacies of the field. The more transparent and self-explanatory the computational workflows are, the easier it is to bring new researchers on board, so that they can more quickly make the step to correctly apply amplitude analysis techniques to the rapidly growing experimental datasets.

5 The ComPWA project

In this chapter, we discuss how the Common Partial-Wave Analysis (ComPWA) project has evolved from a single C++ framework for amplitude analysis to a collection of more dynamic Python tools and documentation workflows that use the techniques described in Chapter 4. The end of the chapter describes how we exposed and extended different functionalities of the original ComPWA framework as easily installable Python packages that can be used independently of each other.

5.1. C++ origins

Most analysis code in high-energy physics and amplitude analysis is written in C++ to facilitate efficient computation and manipulation of large datasets. However, the complexity of C++ can be a barrier to implement and extend large amplitude models. To address this, many projects organise their analysis code as a framework that (1) can build up amplitude models dynamically at runtime and (2) offers a high-level interface for users to define these models through a configuration file that is loaded at runtime without requiring recompilation.

Originally, the Common Partial-Wave Analysis (ComPWA) project also started as a single C++ framework for amplitude analysis. The initial requirement specifications were drafted in 2012 with the intention of designing a collaboration-independent framework that could formulate arbitrary amplitude models, can be flexibly configured, and documents the model in a human-readable format [206]. Inspiration was taken from frameworks like Tara for the Crystal Barrel experiment [207], PWA2000 [208], ROOTPWA for the COMPASS experiment, GPUPWA for BESIII [209], and Laura++ for BaBar and LHCb [210]. The idea was to design a framework with a more modular design, with interfaces that are abstract enough to allow for different implementations for different spin formalisms, estimator methods, minimiser strategies and algorithms, and data sample formats for different collaborations [211].

Figure 5.1 shows the core modules of the original ComPWA framework. Further details can be found in the doctoral theses of M. Michel, P. Weidenkaff, and S. Pflüger [212–214]. The modules provide interfaces that could essentially operate independently of each other and served as protocols for streaming data through the framework. Four-momenta **data** could be either loaded from different file formats or generated using different Monte Carlo generators. **Physics**, such as coherent sum of amplitudes, would be implemented through a function tree (model) that can operate on that data. Different **estimator** implementations could quantise the discrepancy between the model and the data sample. And finally, different **optimiser** implementations could tweak the parameters in the function in order to minimise the estimator value. The key part is that interfaces were general and simple enough that they encapsulated the expected data flow, but still allowed for extending the functionality through specific implementations of the interfaces.

ComPWA followed the classic definition of a framework as an object-oriented *abstract* design, with the preference that classes inherit from an (abstract) base class at most [215; 212, §3.5].

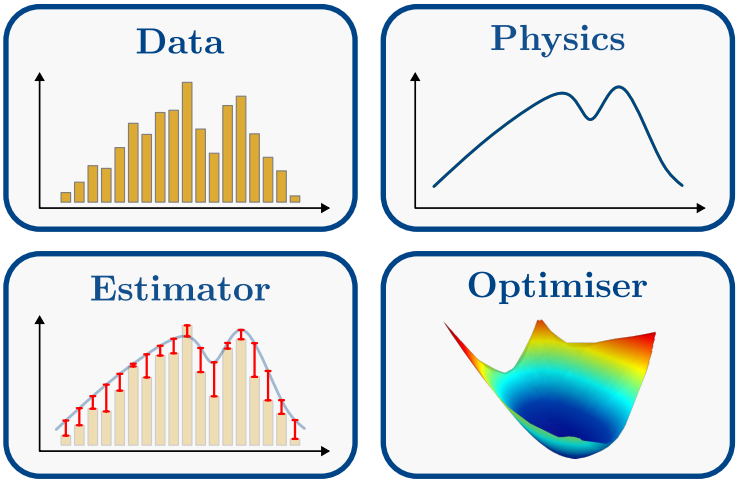


Figure 5.1. Main modules of the ComPWA framework. Inspired by [212].

The concept of a **function tree** deserves special attention. As we will see later, the concept is widely used in other computational frameworks. ComPWA implemented models in the physics module as tree-like representations, with the nodes representing mathematical operations on branches (data points or numbers) or other nodes. This not only provided an interface that can be called agnostically by other modules, but also to facilitate techniques to optimise computations. A case in point is that one can track which nodes depend on certain parameters. If parameters are kept fixed during a minimisation process, the corresponding nodes can be precomputed and cached. In addition, specific nodes can be optimised directly based on their structure. For example, a node with one child can be directly 'collapsed' (restructured) into a single node. Initial tests showed that the speed-ups for certain parameter combinations could be up to 50 times faster [212, p. 82].

Function trees could also be easily serialised, which was intended as a way to document an analysis as well as to export and modify models for reevaluation, as intended by the original requirement specifications [206]. The fact that function trees could be built up from a human-readable format led to the idea of an "expert system" that could automatically generate models based on a set of rules. In the case of amplitude analysis, this means generating decay topologies based on quantum number conservation rules and using that as a template to build up a function tree for the amplitude model.

Finally, a Python interface, `pycompwa`, was developed to provide a more dynamic workflow with the ComPWA framework [216]. First, it provided Python bindings to the C++ implementations of each of the core modules, so that the user could interact with the framework through Python scripts and Jupyter notebooks without having to compile the C++ code. Second, it was complemented by the `expertsystem`, which was implemented purely in Python [217]. The combination of the bindings and the `expertsystem` allowed for a more dynamic workflow with higher-level analysis code.

5.2. Transition to Python

Array programming allows for concise and expressive code that is easier to read and write. While array-oriented code has been popular in the Python community since the 2000s with libraries like NumPy, it gained more traction with the emergence of Machine Learning (ML) libraries like Numba since 2012 [218], TensorFlow in 2015 [219], PyTorch in 2016 [220], and JAX in 2018 [221]. Importantly, these libraries represent their computations (such as neural networks) in the form of a computational graph, which is similar to the function tree concept in ComPWA. Computational graphs, however, usually provide the more advanced features discussed in Section 4.1, such as optimisations through backtracing on the JIT-compiled functions that those graphs represent, or automatic differentiation.

The popularity of these ML frameworks and the fact that they are developed by companies and large groups of professional software developers, makes the effort to maintain an own implementation of a function tree futile. In 2020, it was therefore decided to reimplement the ComPWA framework purely using the Python APIs of those libraries. The decision was further encouraged by initiatives like TensorFlowAnalysis and its follow-up AmpliTF for the amplitude analysis community [222; 223], as well as `zfit`, which serves as an alternative to RooFit with TensorFlow as computational backend [224].

As a first step, we reimplemented the ComPWA framework using only TensorFlow as computational backend, following the design of Figure 5.1. The first release candidate was able to reproduce the standard test cases of the original ComPWA framework and demonstrated that similar fit performance could be achieved with TensorFlow as computational backend. A first release candidate was released on July 9th, 2020, see v0.0.0-alpha0. Unit tests and tutorial examples of the repository at that point in time showed the feasibility of the reimplementation.

The TensorFlow computational graphs were still built up from the same XML files generated from the `expertsystem` that were used as input for the ComPWA C++ framework, which we used as a cross-check for the new implementation. This led us to the realisation that we could use the same input file for constructing computational graphs with different computational libraries. Similarly to AmpliTF, we therefore implemented the amplitude model builder with JAX, next to the existing TensorFlow implementation.

As a final step that led towards the adoption of a CAS (Section 4.2), we explored alternative representations of the XML format for the generated amplitude model. The fact that amplitude models are inherently large mathematical expressions suggested that we adopt a standardised format, such as MathML, for serialising such expressions. Constructing a tree of mathematical nodes to serve as a template for such a serialisation format is essentially a reimplementation of the expression trees used in Computer Algebra Systems, or for instance Python's built-in `operator` module [225]. We therefore decided formulate the amplitude model in the `expertsystem` using SymPy.

SymPy has the additional benefit that it already provides code generation capabilities for expression serialisation as well as for different programming languages. The rest of the software packages therefore became more of a wrapper around SymPy code generation that streamlines the process of performing amplitude analyses with different computational backends, following the original design of the ComPWA framework of Figure 5.1. All in all, the transition to Python and the adoption of a CAS removed the workload of maintaining a computational backend, while opening the door to more flexible workflows, a much easier developer experience, and the more advanced computational features described in Section 4.1. This lead us to transition the project completely to Python.

5.3. Main Python packages

Python makes it easier to split up frameworks into smaller, specialised packages and allows for a fast release flow. Naturally, during the transition to Python, the ComPWA framework evolved into a collection of dedicated Python packages that can mostly be used independently of each other. Figure 5.2 shows the components of the original C++ framework and the new core packages that the ComPWA project has evolved into. The ComPWA framework provided Python bindings to its C++ backend through the `pycompwa` package. This package was replaced by the Python-only TensorWaves package, which streamlines the conversion of SymPy expressions to numerical functions and provides additional functionality for performing amplitude analyses. The `expertsystem` was split into the QRules and Ampform packages, which together provide the necessary tools for formulating symbolic amplitude models. All packages are all available on PyPI as well as conda-forge, so that they can be easily installed or used as libraries in downstream projects.

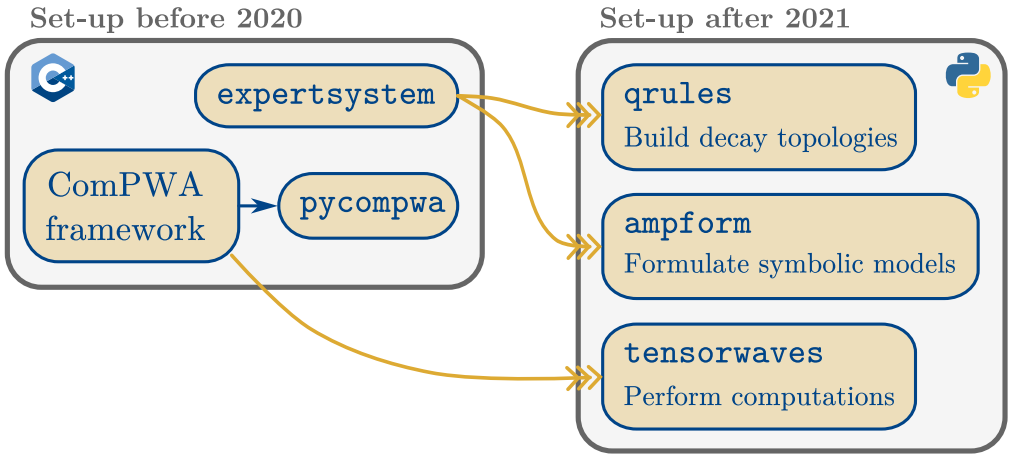


Figure 5.2. ComPWA C++ framework and related Python packages before the transition for Python (left) and the recent main Python packages (right). The yellow arrows indicate from which original framework components the new packages evolved. While the original framework mainly revolved around the C++ framework, the new packages are implemented fully in Python.

QRules

Many steps in the construction of an amplitude model can be standardised, especially once a decay is modelled with the helicity formalism (Chapter 3). Deciding which decays are allowed, and what resonances one can expect, is something one can work out on paper using quantum conservation laws. This is, however, error-prone and something that can be automated. Already at the inception of the ComPWA project [206], this led to the idea of developing a rule-based "expert system" that can automatically generate decay topologies based on quantum number conservation rules.

An expert system emulates the decision-making ability of a human expert. It consists of two main components: a knowledge base and an inference engine. In the case of formulating particle decays, the knowledge base consists of conservation laws, such as charge conservation, and input

information about the decay, such as the initial state and final state. The inference engine can be any algorithm that deduces what kind of decay topologies can connect the initial and final state, based on the facts and rules in the knowledge base. The "expert" implements knowledge through the knowledge base, while a user can use the inference engine to query ("ask advice") the knowledge base.

The inference engine can be implemented in many ways, such as a decision tree, a set of rules, or a neural network. The latter has become more popular in recent years. At the time of writing, the main inference engine used in the ComPWA project is a Constraint Satisfaction Problem (CSP) solver [226]. In most cases, the user specifies outside constraints, such as initial and final state, expected decay topology types, and the conservation rules that apply (such as parity conservation in the case of a strong-force decay). The CSP then tries to satisfy these constraints by building decay topologies and filling them with quantum numbers that comply with the conservation rules.

The ComPWA framework originally included a package named `expertsystem`. However, since this also had the responsibility of formulating amplitude models and exporting them to XML, we extracted a specialised package called QRules with (PyPI name `qrules`) that only contains the particle reaction solver [227].

As an example, the following code (Listing 5.1) shows how a user can request QRules to find which decays can occur in a J/ψ meson decaying to K_S^0, Σ^+, and $\bar{p}$ if one assumes sequential two-body decays (isobar) and applies conservation rules for the strong force. QRules works with flavour eigenstates, because it can only handle the quantum numbers of the particles. In this case, the user has to select K^0 instead of the short-lived mass eigenstate K_S^0. Additional information, such as the maximum angular momentum and a factor that specifies how far resonance may lay outside the kinematically allowed phase space, can be optionally be specified.

Listing 5.1 Problem definition for QRules to find allowed decays for $J/\psi \to K_S^0 \Sigma^+ \bar{p}$.

```
import qrules

reaction = qrules.generate_transitions(
    initial_state=[("J/psi(1S)", (-1, +1))],
    final_state=["K0", "Sigma+", "p~"],
    allowed_interaction_types="strong",
    mass_conservation_factor=0.0,
    max_angular_momentum=3,
    topology_building="isobar",
)
```

In this example, the user has specified that the initial state does not have longitudinal polarisation, which is indicated by the `(-1, +1)` in the initial state. This is the case in the BESIII experiment, where the J/ψ meson is produced in e^+e^- collisions. The `allowed_interaction_types` argument specifies which conservation rules (Table 1.2) are enforced at each node of the sequential decay chains built in the "isobar" topology (two-body decay chain). The CSP solver is then instructed to restrict intermediate states to those with an orbital angular momentum of at most $L = 3$ and a mass conservation factor of $f = 0$, which means that any intermediate state r must lie within $m_r \pm f\Gamma_r$ of the available kinematic phase space.

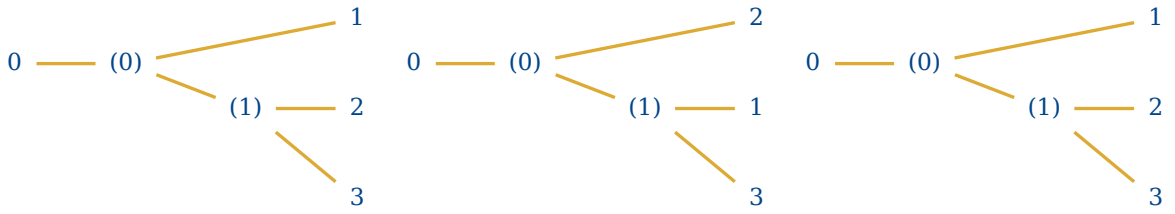


Figure 5.3. Allowed decay topologies that QRules finds for the decay $J/\psi \to K^0_S(1)\,\Sigma^+(2)\,\bar{p}(3)$. QRules renders the topologies with Graphviz in the form of Feynman-like directed graphs, with the initial state on the left and the final state on the right. The edges represent states (particles, or propagators) and the nodes (labelled `(0)` and `(1)`) represent decay vertices with information like angular momentum.

Internally, QRules first builds a list of all possible decay topologies for three-body decay (Figure 5.3). It then fetches the quantum numbers of the initial- and final-state particles (outer edges) that the user has specified in the definition of the problem. By default, this information is obtained from the Particle Data Group [66] through the `particle` package [228], but the user can also provide a database of custom particle definitions when searching for unknown or exotic resonances. Next, QRules fills the inner edges of the topologies with quantum numbers, conservation rules, and expected quantum number domains.

Figure 5.4 shows a QRules rendering of one such filled topology. It represents only one of many possible graphs, since the CSP inference engine explores the full set of topologies from Figure 5.3 as well as different combinations of quantum numbers. The outer edges correspond to the initial- and final-state particles, with one particular set of allowed spin projections shown in brackets. Each node and intermediate edge is annotated with conservation rules, together with priorities that determine the order in which they are applied. They also specify domains for each quantum number type considered at that node or edge during the CSP solving process. For instance, baryon number conservation has the highest priority at both nodes, with possible values of –1, 0, or +1 on the intermediate edge. Strangeness has slightly lower priority and can take values from –3 to +3.

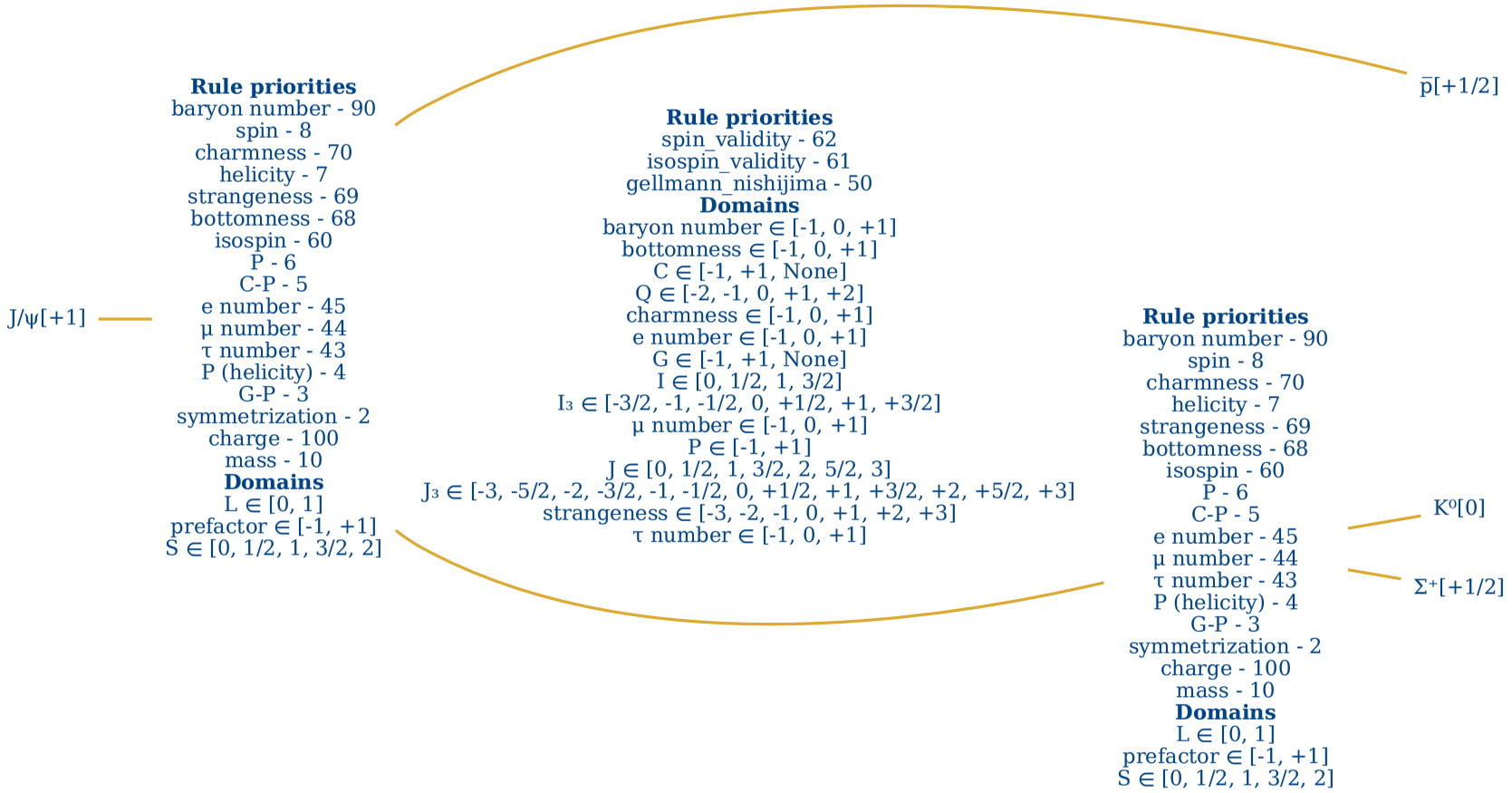


Figure 5.4. Example of a decay topology with quantum numbers and conservation rules filled in. Nodes and intermediate edges show conservation rules along with an assigned priority as well as domains that the CSP inference engine is to consider when solving the problem set later on.

The CSP solver now determines which sets of quantum numbers can be assigned to the graph edges and nodes, while satisfying the conservation rules at each two-body decay node and the fixed quantum numbers on the outer edges. Figure 5.5 shows one of the solutions for the problem defined in Figure 5.4. Here, edges and nodes are assigned explicit quantum numbers such as spin magnitude J, spin projection J_3, parity P, strangeness, and others, along with Particle Data Group (PDG) identifiers of the matching particle definitions. Quantum numbers with value 0 are omitted from the listing. For example, the antiproton $\bar{p}$ (PDG ID `-2212`) carries no strangeness, while the K^0 meson (PDG ID `311`) carries no baryon number.

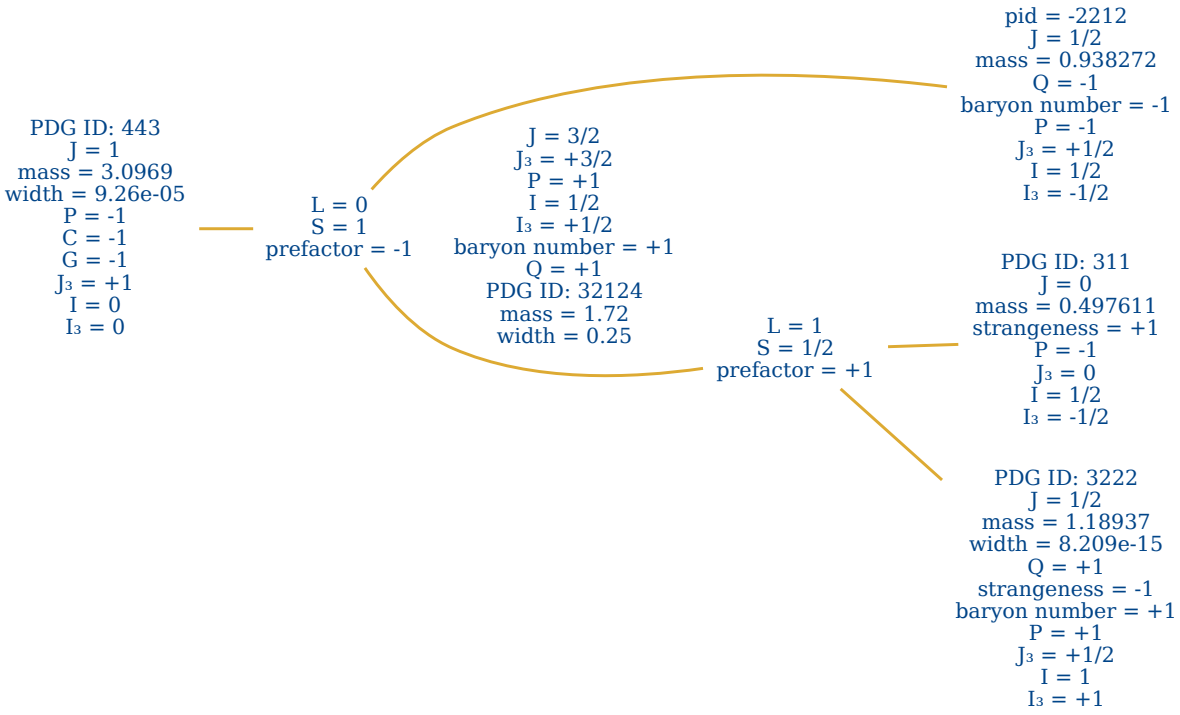


Figure 5.5. Example of a solution with the quantum number sets for the edges and nodes that the CSP has found for the problem set shown in Figure 5.4.

Finally, QRules searches through the particle database for candidates that match the quantum number sets assigned to the intermediate edges of the graph. Figure 5.6 shows the decay topologies that QRules identified for the problem defined in Listing 5.1, collapsed into two graphs. The full reaction object also records the allowed combinations of spin projections. In this case, only two decay-chain topologies were found: one with a subsystem $\Sigma^* \to K^0_S \bar{p}$ and another with $N^* \to K^0_S \Sigma^+$. QRules did not find transitions for a third subsystem, since no K^* mesons were found with masses that satisfy conservation in the topology $J/\psi \to K^0_S(K^* \to \bar{p}\Sigma^+)$. This is a consequence the `mass_conservation_factor` chosen in Listing 5.1, together with the particle definitions available in the default database provided by the `particle` package.

While the figures shown in this section illustrate the strategy that QRules uses to generate allowed decay topologies, the system allows for other solver strategies. For example, instead of using sequential-decay topologies, the system can also check whether a particle reaction is allowed at all by checking whether the quantum numbers of the initial and final state simply add up and comply with conservation rules (such as net charge conservation). The system is

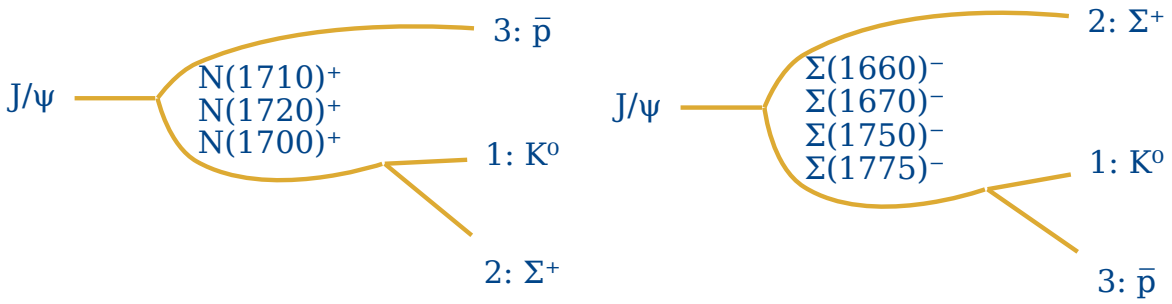


Figure 5.6. Matching particle names for all the solutions that the CSP has found for the problem definition in Listing 5.1. To reduce the number of graphs, we collapse the graphs that have the same topology, remove the spin projections, and show all found resonance names as above each other next to the one intermediate edge in each graph.

also designed to be extendible to more general reactions, for instance involving an initial state of multiple particles. For the purposes of this thesis, this functionality has not been further developed.

AmpForm

As can already be seen from Figure 5.5, the decay graphs that QRules generates, contain much more information than just which resonances can be expected. For example, the graphs contain LS-coupling values, parities, spin projections, and masses and widths of the particles. This information can be used to formulate an amplitude model with the helicity formalism (Chapter 3).

For this purpose, we extracted a second package, AmpForm (PyPI name `ampform`), from the original `expertsystem` package [229]. The package formulates amplitude models symbolically using SymPy rather than serialising them directly to XML, as happened in the original C++ framework. As such, AmpForm serves both as an extension of SymPy – a library of expression classes that are useful for amplitude analysis – and as a tool for formulating amplitude models using different spin formalisms. In the usual workflow, particle reaction graphs generated by QRules serve as input to AmpForm's amplitude model builder, but the user can also use the expression classes directly to build up their own amplitude models or any other parametrisations that are of interest to the analysis.

Amplitude model building

Listing 5.2 shows how the amplitude model is built up for the decay $J/\psi \to K_S^0 \Sigma^+ \bar{p}$ that was generated with QRules in Listing 5.1. It first creates an amplitude model builder, which is a factory class that can be configured dynamically before formulating the model. Next, a dynamics builder class is instantiated that can formulate a relativistic Breit-Wigner functions with an energy-dependent width and two vertex form factors for a given intermediate edge. This dynamics builder is then assigned to each of the intermediate edges in the decay graph. Finally, the `formulate()` method returns a class structure that contains all the relevant definitions for an amplitude model.

Listing 5.2 Example of building an amplitude model with the helicity formalism (no spin alignment) for the decay $J/\psi \to K^0_S \Sigma^+ \bar{p}$ (Listing 5.1) using AmpForm with relativistic Breit–Wigner functions.

```
import ampform
from ampform.dynamics.builder import RelativisticBreitWignerBuilder

model_builder = ampform.get_builder(reaction)
dynamics_builder = RelativisticBreitWignerBuilder(
    energy_dependent_width=True,
    form_factor=True,
)
for resonance in reaction.get_intermediate_particles():
    model_builder.dynamics.assign(resonance.name, dynamics_builder)
model = model_builder.formulate()
model.intensity
```

$$\sum_{m_0\in\{1,-1\}} \sum_{m_1=0} \sum_{m_2=-1/2}^{1/2} \sum_{m_3=-1/2}^{1/2} \left| A^{12}_{m_0,m_1,m_2,m_3} + A^{13}_{m_0,m_1,m_2,m_3} \right|^2$$

The model is formulated in terms of SymPy expressions, which can be combined into a single expression for generating numerical code of the full amplitude model. At the top level, this is represented by `model.intensity`. As its rendering in Listing 5.2 shows, it is a coherent sum over amplitudes (excluding the longitudinal polarisation of the J/ψ meson). The amplitudes $A^{12}_{m_0,m_1,m_2,m_3}$ and $A^{13}_{m_0,m_1,m_2,m_3}$ describes the transition of a J/ψ with spin projection m_0 into the final state K^0_S, Σ^+, and $\bar{p}$ with spin projections m_1, m_2, and m_3, respectively. They correspond to intermediate states in chain (12)3 (subsystem $K^0_S\Sigma^+$) and chain (13)3 (subsystem $K^0_S\bar{p}$), respectively. Note that this amplitude model does not take spin alignment of the baryons in the final state into account and naively implements polarisation by excluding the longitudinal polarisation of the J/ψ meson ($m_0 \neq 0$).

Listing 5.3 Definition of the amplitudes in the intensity expression Listing 5.2 formulated by AmpForm.

```
model.amplitudes
```

$$
\begin{aligned}
A^{12}_{-1,0,-\frac{1}{2},-\frac{1}{2}} &= \frac{C \underset{S=1}{\overset{L=0}{\longrightarrow}} N_1^*\overline{p}; N_1^* \underset{S=1/2}{\overset{L=1}{\longrightarrow}} K^0_S\Sigma^+ \Gamma_{N_1^*} m_{N_1^*} C^{\frac{1}{2},\frac{1}{2}}_{0,0,\frac{1}{2},\frac{1}{2}} C^{1,0}_{0,0,1,0} C^{1,0}_{\frac{1}{2},-\frac{1}{2},\frac{1}{2},\frac{1}{2}} C^{\frac{1}{2},\frac{1}{2}}_{1,0,\frac{1}{2},\frac{1}{2}} \mathcal{F}_1(m^2_{12},m_1,m_2) D^{\frac{1}{2}}_{-\frac{1}{2},\frac{1}{2}}(-\phi_1^{12},\theta_1^{12},0) D^1_{-1,0}(-\phi_{12},\theta_{12},0)}{-m^2_{12}+\left(m_{N_1^*}\right)^2 - i m_{N_1^*}\Gamma_1(m^2_{12})} \\
&+ \cdots \\
&+ \frac{C \underset{S=2}{\overset{L=2}{\longrightarrow}} N_3^*\overline{p}; N_3^* \underset{S=1/2}{\overset{L=1}{\longrightarrow}} K^0_S\Sigma^+ \Gamma_{N_3^*} m_{N_3^*} C^{\frac{1}{2},\frac{1}{2}}_{0,0,\frac{1}{2},\frac{1}{2}} C^{\frac{3}{2},\frac{1}{2}}_{1,0,\frac{1}{2},\frac{1}{2}} C^{2,-1}_{\frac{3}{2},-\frac{3}{2},\frac{1}{2},\frac{1}{2}} C^{1,-1}_{2,0,2,-1} \mathcal{F}_1(m^2_{12},m_1,m_2) D^1_{-1,-1}(-\phi_{12},\theta_{12},0) D^{\frac{3}{2}}_{-\frac{3}{2},\frac{1}{2}}(-\phi_1^{12},\theta_1^{12},0)}{-m^2_{12}+\left(m_{N_3^*}\right)^2 - i m_{N_3^*}\Gamma_3(m^2_{12})} \\
A^{12}_{-1,0,-\frac{1}{2},\frac{1}{2}} &= \frac{C \underset{S=1}{\overset{L=0}{\longrightarrow}} N_1^*\overline{p}; N_1^* \underset{S=1/2}{\overset{L=1}{\longrightarrow}} K^0_S\Sigma^+ \Gamma_{N_1^*} m_{N_1^*} C^{\frac{1}{2},\frac{1}{2}}_{0,0,\frac{1}{2},\frac{1}{2}} C^{1,0}_{0,0,1,0} C^{1,0}_{\frac{1}{2},\frac{1}{2},\frac{1}{2},-\frac{1}{2}} C^{\frac{1}{2},\frac{1}{2}}_{1,0,\frac{1}{2},\frac{1}{2}} \mathcal{F}_1(m^2_{12},m_1,m_2) D^{\frac{1}{2}}_{\frac{1}{2},\frac{1}{2}}(-\phi_1^{12},\theta_1^{12},0) D^1_{-1,0}(-\phi_{12},\theta_{12},0)}{-m^2_{12}+\left(m_{N_1^*}\right)^2 - i m_{N_1^*}\Gamma_1(m^2_{12})} \\
&+ \cdots \\
&+ \frac{C \underset{S=2}{\overset{L=2}{\longrightarrow}} N_3^*\overline{p}; N_3^* \underset{S=1/2}{\overset{L=1}{\longrightarrow}} K^0_S\Sigma^+ \Gamma_{N_3^*} m_{N_3^*} C^{\frac{1}{2},\frac{1}{2}}_{0,0,\frac{1}{2},\frac{1}{2}} C^{\frac{3}{2},\frac{1}{2}}_{1,0,\frac{1}{2},\frac{1}{2}} C^{2,-1}_{\frac{3}{2},-\frac{1}{2},\frac{1}{2},-\frac{1}{2}} C^{1,-1}_{2,0,2,-1} \mathcal{F}_1(m^2_{12},m_1,m_2) D^1_{-1,-1}(-\phi_{12},\theta_{12},0) D^{\frac{3}{2}}_{-\frac{1}{2},\frac{1}{2}}(-\phi_1^{12},\theta_1^{12},0)}{-m^2_{12}+\left(m_{N_3^*}\right)^2 - i m_{N_3^*}\Gamma_3(m^2_{12})} \\
&\vdots \\
A^{13}_{1,0,\frac{1}{2},\frac{1}{2}} &= \frac{C \underset{S=0}{\overset{L=1}{\longrightarrow}} \Sigma_1^*\Sigma^+; \Sigma_1^* \underset{S=1/2}{\overset{L=0}{\longrightarrow}} K^0_S\overline{p} \Gamma_{\Sigma_1^*} m_{\Sigma_1^*} \left(C^{\frac{1}{2},-\frac{1}{2}}_{0,0,\frac{1}{2},-\frac{1}{2}}\right)^2 C^{0,0}_{\frac{1}{2},\frac{1}{2},\frac{1}{2},-\frac{1}{2}} C^{1,0}_{1,0,0,0} \mathcal{F}_0(m^2_{13},m_1,m_3) D^{\frac{1}{2}}_{\frac{1}{2},-\frac{1}{2}}(-\phi_1^{13},\theta_1^{13},0) D^1_{1,0}(-\phi_{13},\theta_{13},0)}{-m^2_{13}+\left(m_{\Sigma_1^*}\right)^2 - i m_{\Sigma_1^*}\Gamma_1(m^2_{13})} \\
&+ \cdots \\
&+ \frac{C \underset{S=2}{\overset{L=1}{\longrightarrow}} \Sigma_2^*\Sigma^+; \Sigma_2^* \underset{S=1/2}{\overset{L=2}{\longrightarrow}} K^0_S\overline{p} \Gamma_{\Sigma_2^*} m_{\Sigma_2^*} C^{\frac{1}{2},-\frac{1}{2}}_{0,0,\frac{1}{2},-\frac{1}{2}} C^{1,-1}_{1,0,2,-1} C^{\frac{5}{2},-\frac{1}{2}}_{2,0,\frac{1}{2},-\frac{1}{2}} C^{2,-1}_{\frac{5}{2},-\frac{1}{2},\frac{1}{2},-\frac{1}{2}} \mathcal{F}_2(m^2_{13},m_1,m_3) D^1_{1,-1}(-\phi_{13},\theta_{13},0) D^{\frac{5}{2}}_{-\frac{1}{2},-\frac{1}{2}}(-\phi_1^{13},\theta_1^{13},0)}{-m^2_{13}+\left(m_{\Sigma_2^*}\right)^2 - i m_{\Sigma_2^*}\Gamma_2(m^2_{13})}
\end{aligned}
$$

Listing 5.3 shows some of amplitudes contained in the `model.amplitudes` mapping, with the LaTeX rendering generated directly by the codebase. For example, the first amplitude, $A^{12}_{-1,0,-\frac{1}{2},-\frac{1}{2}}$ is a coherent sum of all amplitudes in the subsystem, $J/\psi[-1] \to \bar{p}[-\frac{1}{2}]\, N^*$ for all resonances $N^* \to K^0_S[0]\Sigma^+[-\frac{1}{2}]$. To save space, N^* and Σ^* resonances are marked with an index rather than their full name. The model is formulated in the canonical basis using the transformation Equation (3.17). AmpForm takes the LS couplings generated by QRules, but it can also formulate in the helicity basis. The structure of each amplitude is always the same: a complex-valued coefficient (scaling factor with phase), two form factors $\mathcal{F}_L, \mathcal{F}_\ell$ for angular momentum L in the production vertex and ℓ in the decay vertex, two Clebsch–Gordan coefficients and one Wigner D-function per vertex, and a Breit–Wigner function with energy-dependent width for the intermediate edge (resonance) in each graph.

The amplitudes contain symbols that can be interpreted as either a **parameter** (such as resonance mass $m_{N^*_1}$ and width $\Gamma_{N^*_1}$) and **kinematic variables** (such as the invariant mass m_{12} and helicity angle θ^{12}_1 of particle 1 in the rest frame of particle 1 and 2). AmpForm provides suggested default values for the parameters by extracting them from the particle database that the user provides in Listing 5.1. It also defines symbolic expressions for computing the kinematic variables from the three four-momenta of the final state. For example, ϕ^{12}_1 is defined as the azimuthal angle of the four-momentum of particle 1 boosted into the rest frame of the sum of the four-momenta of particle 1 and 2 and with the z axis rotated into the direction of movement (see Equation (3.9)).

```
phi = sp.Symbol("phi_1^12", real=True)
model.kinematic_variables[phi]
```

$$\phi\left(B_z\left(\frac{|\vec{p_{12}}|}{E\left(p_{12}\right)}\right) R_y\left(-\theta\left(p_{12}\right)\right) R_z\left(-\phi\left(p_{12}\right)\right) p_1\right)$$

Together, all attributes of an amplitude model object generated by AmpForm contain the information to evaluate the intensity for a random point in phase space. For this, the amplitudes of Listing 5.3 are symbolically substituted into the main intensity definition of Listing 5.2. The resulting expression, as well as the definitions in `model.kinematic_variables`, can be used as template for generating numerical code for high-performance computations.

Nested expression definitions

In principle, amplitude models can be fully implemented with SymPy directly. However, amplitude models contain so many mathematical operations, that it is hard to render them concisely. For this reason, AmpForm provides a few tools that help to define amplitude model expressions in a nested form that is understandable, making it more suitable for a self-documenting workflow (Section 4.3). The expression $\mathcal{F}_2\left(m^2_{12}, m_1, m_2\right)$ in Listing 5.3 is an example of this and is built up of other expression classes, like the Blatt–Weisskopf function and a squared breakup momentum function.

```
from ampform.dynamics import FormFactor

m_12, m_1, m_2 = sp.symbols("m_12 m_1 m_2", nonnegative=True)
expr = FormFactor(m_12**2, m_1, m_2, angular_momentum=2)
```

$$\begin{aligned}
\mathcal{F}_2\left(m_{12}^2, m_1, m_2\right) &= \sqrt{B_2^2\left(q_{12}^2\left(m_{12}^2\right)\right)} \\
B_2^2\left(q_{12}^2\left(m_{12}^2\right)\right) &= \frac{13 q_{12}^2\left(m_{12}^2\right)^2}{q_{12}^2\left(m_{12}^2\right)^2 + 3 q_{12}^2\left(m_{12}^2\right) + 9} \\
q_{12}^2\left(m_{12}^2\right) &= \frac{\left(m_{12}^2 - (m_1 - m_2)^2\right)\left(m_{12}^2 - (m_1 + m_2)^2\right)}{4 m_{12}^2}
\end{aligned}$$

Nested expression classes like these can be easily extended using the decorator function, `@unevaluated`, provided by AmpForm. This allows for extending amplitude builders with large mathematical definitions. A simplified example is shown in Listing 5.4. The expression class is 'unfolded' to its full form using SymPy's `doit()` method.

Listing 5.4 Example of a custom expression class that uses the `@unevaluated` decorator to postpone evaluation of the expression until it is explicitly requested. The class is used to define a custom expression for $f(x, y) = x^2 + y^2$. The second snippet uses AmpForm's `aslatex()` helper function to render f and its `doit()` definition.

```
from ampform.sympy import unevaluated

@unevaluated
class MyExpr(sp.Expr):
    x: sp.Symbol
    y: sp.Symbol
    _latex_repr_ = "f({x}, {y})"

    def evaluate(self) -> sp.Expr:
        x, y = self.args
        return x**2 + y**2

a, b = sp.symbols("a b")
expr = MyExpr(a, b**2)
```

```
from ampform.io import aslatex
Math(aslatex({expr: expr.doit()}))
```

$$f(a, b^2) = a^2 + b^4$$

Cached expression unfolding

Combining all the definitions into one large expression template can be slow when using SymPy. AmpForm therefore provides ways to automatically cache the result of this "expression unfolding" to disk. All of these functions are available through the `ampform.sympy.cached` module, which caches expensive CAS operations to disk. Listing 5.5 shows two equivalent ways of fully unfolding the amplitude model expression that was generated with Listing 5.2.

These are two examples of additional functionality that AmpForm provides to facilitate working with large symbolic expressions for amplitude analysis. It contains more functionality, such as

Listing 5.5 Examples of AmpForm's `cached` that caches expensive CAS operations to disk. The first example shows how the full intensity expression is constructed from an amplitude model object and the second example shows how the `cached.unfold()` convenience function simplifies that procedure.

```
from ampform.sympy import cached

intensity_expr = cached.doit(model.intensity)
expr_with_amplitudes = cached.xreplace(intensity_expr, model.amplitudes)
full_expression = cached.doit(expr_with_amplitudes)
```

```
full_expression = cached.unfold(model)
```

an `UnevaluatableIntegral` class that postpones evaluation of integrals that cannot be solved analytically, which is essential when parametrising decay dynamics with lineshapes that are analytically continuous.

AmpForm-DPD

The central physics contribution of this thesis is the implementation of spin-aligned amplitude models for three-body decays using Dalitz-plot decomposition. AmpForm can handle arbitrary multi-body decays, and works well as long as the decay does not involve chains with differing topologies that include final-state particles with spin. In such cases it becomes difficult to formulate spin-aligned models symbolically, since the Wigner rotation angles must be computed numerically (Equation (3.20)). To address this, a specialised package was developed alongside AmpForm: **AmpForm-DPD** (PyPI: `ampform-dpd`). This package builds amplitude models with SymPy using the DPD method [157], in particular the unpolarised intensity of Equation (3.24) with the Dalitz-plot function of Equation (3.26). In the long term, the goal is to merge AmpForm-DPD back into AmpForm and extend spin alignment to arbitrary multi-body decays [156].

Listing 5.6 shows how an amplitude model is formulated with AmpForm-DPD. The procedure is similar as in Listing 5.2, but the main difference is that the amplitude model is built up from a custom class structure that represents decay chains for three-body decays. The rendered intensity expression is the implementation of the unpolarised differential cross section of Equation (3.24). As opposed to Listing 5.2, AmpForm-DPD correctly sums over all J/ψ helicities to get the unpolarised intensity. The trivial helicity $\lambda_1 = 0$ for K_S^0 has been omitted from the amplitude indices.

Note that the Wigner d-functions for the Wigner rotations appearing in the Dalitz-plot function of Equation (3.26) appear in this top-level intensity expression rather than in the amplitudes $A^k_{\lambda_0\lambda_2\lambda_3}$. These individual amplitudes are shown in Listing 5.7. The amplitude rendering is clearer than that of Listing 5.3, because it uses AmpForm's `@unevaluated` expression classes of Listing 5.4 to denote the vertex functions with $\mathcal{F}$ and Breit–Wigner functions with $\mathcal{R}$. AmpForm-DPD also formulates the model with one complex-valued *coupling* per decay node rather than with one coefficient per *chain*. The angles are formulated using Equation (3.27) and are shown in Listing 5.8.

Listing 5.6 Example of building an amplitude model for the decay $J/\psi \to K_S^0 \Sigma^+ \bar{p}$ using Dalitz-plot decomposition with AmpForm-DPD. As opposed to Listing 5.2, the resulting intensity expression correctly aligns the two subsystems.

```
from ampform_dpd import DalitzPlotDecompositionBuilder
from ampform_dpd.adapter.qrules import to_three_body_decay
from ampform_dpd.dynamics.builder import (
    formulate_breit_wigner_with_form_factor
)

decay = to_three_body_decay(reaction.transitions)
model_builder = DalitzPlotDecompositionBuilder(decay)
for chain in model_builder.decay.chains:
    model_builder.dynamics_choices.register_builder(
        chain, formulate_breit_wigner_with_form_factor
    )
dpd_model = model_builder.formulate(reference_subsystem=2)
dpd_model.intensity
```

$$\sum_{\lambda_0=-1}^{1}\sum_{\lambda_2=-1/2}^{1/2}\sum_{\lambda_3=-1/2}^{1/2}\left|\sum_{\lambda_0'=-1}^{1}\sum_{\lambda_2'=-1/2}^{1/2}\sum_{\lambda_3'=-1/2}^{1/2} A^2_{\lambda_0'\lambda_2'\lambda_3'} d^{\frac{1}{2}}_{\lambda_2'\lambda_2}\left(\zeta^2_{2(2)}\right) d^{\frac{1}{2}}_{\lambda_3'\lambda_3}\left(\zeta^3_{2(2)}\right) d^{1}_{\lambda_0\lambda_0'}\left(\zeta^0_{2(2)}\right) + A^3_{\lambda_0'\lambda_2'\lambda_3'} d^{\frac{1}{2}}_{\lambda_2'\lambda_2}\left(\zeta^2_{3(2)}\right) d^{\frac{1}{2}}_{\lambda_3'\lambda_3}\left(\zeta^3_{3(2)}\right) d^{1}_{\lambda_0\lambda_0'}\left(\zeta^0_{3(2)}\right)\right|^2$$

Listing 5.7 Definition of the DPD amplitudes in the intensity expression of Listing 5.6. The listing also uses AmpForm's `aslatex()` helper function, which renders mappings of expressions as a single LaTeX block.

```
from IPython.display import Math
from ampform.io import aslatex
Math(aslatex(dpd_model.amplitudes))
```

$$
\begin{aligned}
A^{2}_{-1,0,-\frac{1}{2},-\frac{1}{2}} &= 2\sum\nolimits_{\lambda_R=-3/2}^{3/2} -\delta_{-1,\lambda_R+\frac{1}{2}} \mathcal{F}_1\left(m_0^2, m_{\Sigma_0^*}, m_2\right) \mathcal{F}_2\left(\sigma_2^2, m_1, m_3\right) \mathcal{H}^{\text{decay}}_{\Sigma_0^*,-\frac{1}{2},0} \mathcal{H}^{\text{production}}_{\Sigma_0^*,\lambda_R,-\frac{1}{2}} \mathcal{R}_2\left(\sigma_2, m_{\Sigma_0^*}, \Gamma_{\Sigma_0^*}\right) d^{\frac{3}{2}}_{\lambda_R,-\frac{1}{2}}\left(\theta_{31}\right) \\
&+ \cdots \\
&+ \sum\nolimits_{\lambda_R=-5/2}^{5/2} -\delta_{-1,\lambda_R+\frac{1}{2}} \mathcal{F}_1\left(m_0^2, m_{\Sigma_2^*}, m_2\right) \mathcal{F}_2\left(\sigma_2^2, m_1, m_3\right) \mathcal{H}^{\text{decay}}_{\Sigma_2^*,-\frac{1}{2},0} \mathcal{H}^{\text{production}}_{\Sigma_2^*,\lambda_R,-\frac{1}{2}} \mathcal{R}_2\left(\sigma_2, m_{\Sigma_2^*}, \Gamma_{\Sigma_2^*}\right) d^{\frac{5}{2}}_{\lambda_R,-\frac{1}{2}}\left(\theta_{31}\right) \\
A^{3}_{-1,0,-\frac{1}{2},-\frac{1}{2}} &= 2\sum\nolimits_{\lambda_R=-3/2}^{3/2} \delta_{-1,\lambda_R+\frac{1}{2}} \mathcal{F}_1\left(m_0^2, m_{N_2^*}, m_3\right) \mathcal{F}_2\left(\sigma_3^2, m_1, m_2\right) \mathcal{H}^{\text{decay}}_{N_2^*,0,-\frac{1}{2}} \mathcal{H}^{\text{production}}_{N_2^*,\lambda_R,-\frac{1}{2}} \mathcal{R}_2\left(\sigma_3, m_{N_2^*}, \Gamma_{N_2^*}\right) d^{\frac{3}{2}}_{\lambda_R,\frac{1}{2}}\left(\theta_{12}\right) \\
&+ \cdots \\
&+ \sum\nolimits_{\lambda_R=-3/2}^{3/2} \delta_{-1,\lambda_R+\frac{1}{2}} \mathcal{F}_0\left(m_0^2, m_{N_3^*}, m_3\right) \mathcal{F}_1\left(\sigma_3^2, m_1, m_2\right) \mathcal{H}^{\text{decay}}_{N_3^*,0,-\frac{1}{2}} \mathcal{H}^{\text{production}}_{N_3^*,\lambda_R,-\frac{1}{2}} \mathcal{R}_1\left(\sigma_3, m_{N_3^*}, \Gamma_{N_3^*}\right) d^{\frac{3}{2}}_{\lambda_R,\frac{1}{2}}\left(\theta_{12}\right) \\
&\vdots \\
A^{3}_{1,0,\frac{1}{2},\frac{1}{2}} &= 2\sum\nolimits_{\lambda_R=-3/2}^{3/2} \delta_{1,\lambda_R-\frac{1}{2}} \mathcal{F}_1\left(m_0^2, m_{N_2^*}, m_3\right) \mathcal{F}_2\left(\sigma_3^2, m_1, m_2\right) \mathcal{H}^{\text{decay}}_{N_2^*,0,\frac{1}{2}} \mathcal{H}^{\text{production}}_{N_2^*,\lambda_R,\frac{1}{2}} \mathcal{R}_2\left(\sigma_3, m_{N_2^*}, \Gamma_{N_2^*}\right) d^{\frac{3}{2}}_{\lambda_R,-\frac{1}{2}}\left(\theta_{12}\right) \\
&+ \cdots \\
&+ \sum\nolimits_{\lambda_R=-3/2}^{3/2} \delta_{1,\lambda_R-\frac{1}{2}} \mathcal{F}_0\left(m_0^2, m_{N_3^*}, m_3\right) \mathcal{F}_1\left(\sigma_3^2, m_1, m_2\right) \mathcal{H}^{\text{decay}}_{N_3^*,0,\frac{1}{2}} \mathcal{H}^{\text{production}}_{N_3^*,\lambda_R,\frac{1}{2}} \mathcal{R}_1\left(\sigma_3, m_{N_3^*}, \Gamma_{N_3^*}\right) d^{\frac{3}{2}}_{\lambda_R,-\frac{1}{2}}\left(\theta_{12}\right)
\end{aligned}
$$

Listing 5.8 Definitions of the DPD angles (kinematic variables) appearing in Listing 5.6 and Listing 5.7.

```
Math(aslatex(dpd_model.variables))
```

$$
\begin{aligned}
\theta_{31} &= \operatorname{acos}\left(\frac{2\sigma_2\left(-m_2^2-m_3^2+\sigma_1\right)-\left(m_0^2-m_2^2-\sigma_2\right)\left(-m_1^2+m_3^2+\sigma_2\right)}{\sqrt{\lambda\left(m_0^2,m_2^2,\sigma_2\right)}\sqrt{\lambda\left(\sigma_2,m_3^2,m_1^2\right)}}\right)\\
\theta_{12} &= \operatorname{acos}\left(\frac{2\sigma_3\left(-m_1^2-m_3^2+\sigma_2\right)-\left(m_0^2-m_3^2-\sigma_3\right)\left(m_1^2-m_2^2+\sigma_3\right)}{\sqrt{\lambda\left(m_0^2,m_3^2,\sigma_3\right)}\sqrt{\lambda\left(\sigma_3,m_1^2,m_2^2\right)}}\right)\\
\zeta_{2(2)}^{0} &= 0\\
\zeta_{2(2)}^{2} &= 0\\
\zeta_{2(2)}^{3} &= 0\\
\zeta_{3(2)}^{0} &= -\operatorname{acos}\left(\frac{-2m_0^2\left(-m_2^2-m_3^2+\sigma_1\right)+\left(m_0^2+m_2^2-\sigma_2\right)\left(m_0^2+m_3^2-\sigma_3\right)}{\sqrt{\lambda\left(m_0^2,m_3^2,\sigma_3\right)}\sqrt{\lambda\left(m_0^2,\sigma_2,m_2^2\right)}}\right)\\
\zeta_{3(2)}^{2} &= \operatorname{acos}\left(\frac{2m_2^2\left(-m_0^2-m_1^2+\sigma_1\right)+\left(m_0^2+m_2^2-\sigma_2\right)\left(-m_1^2-m_2^2+\sigma_3\right)}{\sqrt{\lambda\left(m_0^2,m_2^2,\sigma_2\right)}\sqrt{\lambda\left(\sigma_3,m_2^2,m_1^2\right)}}\right)\\
\zeta_{3(2)}^{3} &= \operatorname{acos}\left(\frac{2m_3^2\left(-m_0^2-m_1^2+\sigma_1\right)+\left(m_0^2+m_3^2-\sigma_3\right)\left(-m_1^2-m_3^2+\sigma_2\right)}{\sqrt{\lambda\left(m_0^2,m_3^2,\sigma_3\right)}\sqrt{\lambda\left(\sigma_2,m_3^2,m_1^2\right)}}\right)
\end{aligned}
$$

TensorWaves

The last package in the toolchain is TensorWaves (PyPI name `tensorwaves`) [230]. At core, it is a wrapper around SymPy's code generation capabilities that makes it easier to work with large symbolic expressions and fit them to large data samples. As discussed, TensorWaves started as a reimplementation of the ComPWA framework in TensorFlow and inherits many of the interfaces from the original design. Importantly, the interfaces are kept as simple as possible, so that it is easy to insert custom implementations into the workflow (open-closed principle).

The central interface of TensorWaves is the `Function` class. It is the most generic class for defining mathematical functions that take input and return output. More specific interfaces are derived from this generic class. Two examples relevant for amplitude analysis are a `ParametrizedFunction` and an `Estimator`. A `ParametrizedFunction` can be used to separate the parameters from the kinematic variables in an amplitude model. It takes a `DataSample`, which is a mapping of string keys to data arrays (like pre-computed helicity angles), and carries around scalar parameter values that can be tweaked separately. The `Estimator` represents a function that takes a set of scalar parameters and returns a single estimator value, such as a likelihood or a chi-square value. Both functions take mappings of kinematic variables (data columns) and parameters as input rather than a list of positional arguments. This allows for a large number of arguments to be handled without being constrained to valid Python variable names.

For performing a fit, TensorWaves provides an `Optimizer` interface. It takes an `Estimator` along with a mapping with parameter values that serve as an initial starting point in parameter space. At the time of writing, TensorWaves comes with two implementations for the `Optimizer` interface: `Minuit2` [231; 232] and `ScipyMinimizer` [233]. Other optimisers such as NLopt could easily be implemented without redesigning the interfaces. In the analyses presented in this thesis, we found that the `Minuit2` implementation was the most robust and fastest and we therefore did not invest in implementing other optimisers. In addition, benchmarks using TensorWaves indicate that array-oriented libraries are much more performant when the computation is parallelised in parameter space as well. This suggests that it may be more efficient to perform fits using optimisers that are parallelisable in parameter space, rather than using optimisers that use gradient-descent methods, which is inherently sequential in parameter space.

Behind these main interfaces, TensorWaves uses SymPy's code generation capabilities to generate code for the mathematical functions. This makes it possible to switch between different computational backends, such as TensorFlow, JAX, and NumPy. The generated code is optionally JIT-compiled and wrapped in classes that follow interfaces such as `ParametrizedFunction` and `Estimator`. These interface classes are designed so that a large number of implementations can be used without having to rewrite the workflow. Listing 5.9 shows how the unfolded intensity expression for the DPD model in Listing 5.6 is used to generate a `ParametrizedFunction` that can be used for fitting.

Internally, the created function is a JAX function of a lambdified form that is shown in Section 4.2, but with dozens of arguments. The `ParametrizedFunction` wraps this function, so that the input data columns and current parameter values are mapped to the correct argument.

Ultimately, the goal of these packages is to translate the description of scattering problems into computational code that can efficiently evaluate amplitude models on large experimental data samples. Each package addresses a specific task within the workflow, and their combination

Listing 5.9 Example of using TensorWaves to generate a parametrised function with JAX as backend for the intensity expression of the amplitude model in Listing 5.6. Symbols in the expression that are listed under `parameter_defaults` are marked as parameters in the generated `ParametrizedFunction`. Any remaining free symbols become arguments to the function call through an input `DataSample`.

```
from tensorwaves.function.sympy import create_parametrized_function

intensity_func = create_parametrized_function(
    expression=cached.unfold(dpd_model),
    parameters=dpd_model.parameter_defaults,
    backend="jax",
)
```

provides a coherent toolchain for translating theoretical models into practical, high-performance analyses.

5.4. Developer Experience

As discussed in Section 5.2, the move to Python eliminated much of the burden of maintaining a high-performance computational backend in C++. This section highlights several ways in which Python also improves the developer experience within ComPWA and how this strengthens the prospects for academic continuity. Beyond enabling amplitude analyses, the workflow and tools are designed to lower the entry barrier for new contributors, while ensuring that the code remains thoroughly tested and well documented. This is especially important in a small scientific community like hadron physics, where the roles of user and developer often overlap.

Source control

All projects of the ComPWA project are hosted as Git repositories on GitHub under github.com/ComPWA. The repositories enforce a linear commit history through pull requests (PRs) that are squash-merged into single commits. This ensures that changes to the code base remain comprehensive and that code and documentation quality is guaranteed before a PR is merged. All repositories that host general-use packages are public and licensed under an Apache-2.0 license.

Repositories that contain source code for Python packages deploy the package through the Python Package Index (PyPI), as well as conda-forge, and are versioned according to Semantic Versioning. These repositories are also automatically archived on Zenodo with a DOI, so that users can cite the code in their publications.

Pinned developer environment

All ComPWA repositories come with a `pyproject.toml` file that specifies the dependencies of the project, organised in terms of dependency groups. In repositories where stability is preferred, lock files are provided that pin all dependencies to specific versions, ensuring that the environment is consistent both on the operating system where the developer works as well as

the continuous integration (CI) environment where the code is tested. Over time, we employed different strategies for generating lock files, but recently settled for `uv` for Python-only projects and Pixi for projects that require other languages, such as Julia. Both tools are able to generate and update cross-platform lock files with high speed. Cloning the repository and activating the environment with these tools is all the user has to do to start developing.

Code quality assurance

All packages include **unit tests** that are automatically executed whenever a pull request (PR) is submitted. Test coverage is tracked with the Codecov service to ensure the code remains well-tested. In addition, the repositories provide tutorials in the form of Jupyter notebooks. These notebooks also act as **integration tests**, since they are executed during the documentation build triggered by each PR. The GitHub repositories are configured so that code can only be merged into the production branch via a PR that has passed these checks. AmpForm and TensorWaves further include continuous benchmarks that track the runtime of selected tests and issue warnings if changes to the codebase result in performance regressions.

ComPWA enforces its style guidelines automatically rather than describing them in a document, so that the style can be continuously updated and developers can stay in the workflow without having to consult a guide. This is done through autoformatters such as Prettier and linters such as Ruff. These tools are enforced through Pre-commit, which performs a set of checks locally on every commit as well as on PRs, applying autofixes where possible. In projects with much documentation, spelling is checked through Pre-commit as well with a CSpell hook. Other developer tools that the ComPWA project uses, can be found under compwa.github.io/develop [234].

Consistent and interlinked documentation

All ComPWA packages come with documentation of the API (function and class descriptions) as well as tutorials in the form of Jupyter notebooks. The text is kept minimal by linking to relevant resources as well as to other parts of the documentation. An example are automatic links in code examples to the function description in the API, so that the main text does not have to explain the code example in detail. The interlinking of the documentation is continuously tested as well to ensure that links are not broken. On each release of a package, a separate, versioned website is hosted through the Read the Docs documentation hosting service. In addition, links to the APIs of external dependencies point to the specific version of the dependency at the time the documentation was built, so that links in older versions of the documentation remain valid. All of these techniques ensure that the documentation is consistent and up-to-date, even when the code changes, and reduce the maintenance load of the documentation.

Rapid prototyping

Python is a high-level language that allows for rapid prototyping of new features. This is especially important in a research environment, where new ideas and concepts need to be tested quickly. The ability to write code in a more human-readable way allows for faster iteration and experimentation. We test and document ideas in the form of Jupyter notebooks named "Technical Reports" hosted on compwa.github.io/report. The notebooks are automatically run, tested, and deployed as webpages for preservation.

6 Experimental set-ups

For this work, we carried out two studies using data from two collider experiments: the **LHCb experiment** at the Large Hadron Collider at CERN, and the **BESIII experiment** at the Beijing Electron–Positron Collider II (BEPCII) at IHEP in Beijing. The following sections describe both experimental setups.

6.1. LHCb experiment

The Large Hadron Collider (LHC) at CERN is the world's largest and most powerful particle accelerator. It collides beams of protons or heavy ions at center-of-mass energies of up to 13.6 TeV at several interaction points around its 27-km ring (see Figure 6.1). Dedicated experiments at these points pursue different physics programs: ATLAS and CMS conduct general-purpose searches, ALICE focuses on heavy-ion physics, and LHCb is optimised for precision studies of heavy-flavour hadrons and spectroscopy.

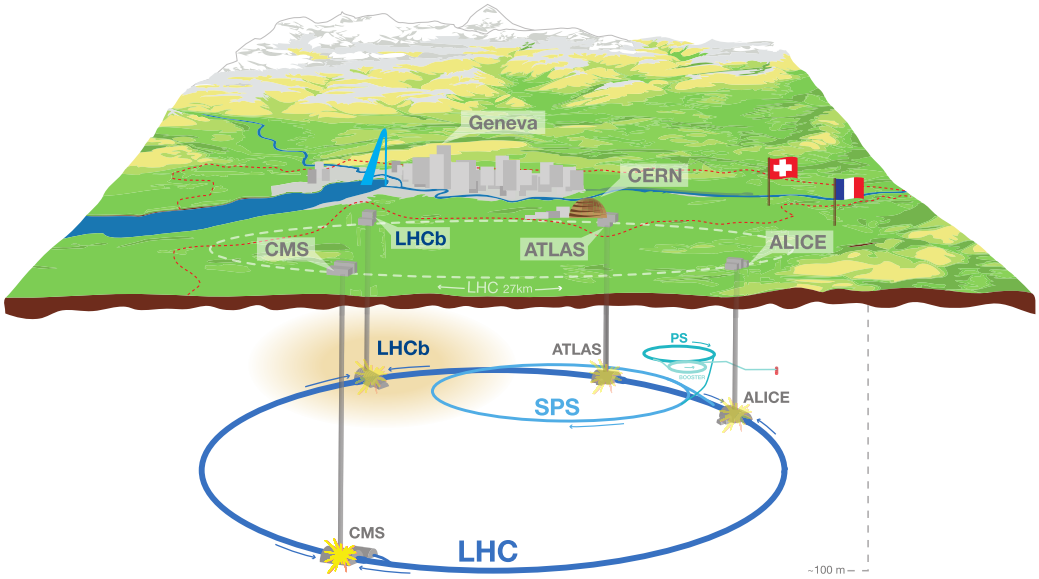


Figure 6.1. Schematic view of the Large Hadron Collider and its four main experiments located underground near Geneva [235]. The position of the LHCb experiment is highlighted in yellow.

The LHCb experiment is dedicated to the study of heavy-flavour hadrons. Its primary aim is to test the Standard Model through precision measurements of CP violation, rare decays, and other processes that are sensitive to potential new physics. An important part of its program is

devoted to the spectroscopy of hadrons containing charm and bottom quarks, where LHCb has provided some of the most precise results worldwide, including the discovery and characterisation of exotic states such as tetraquarks and pentaquarks. In addition, the experiment contributes to electroweak measurements, QCD studies, and heavy-ion collisions.

The LHCb detector is a single-arm forward spectrometer covering the pseudorapidity range $2 < \eta < 5$, which corresponds to particles produced at small angles with respect to the beam. Its geometry is optimised for the study of heavy-flavour hadrons, which are predominantly produced in this forward region. The detector combines precise tracking and vertex reconstruction with efficient photon and particle identification, complemented by a fast trigger system. The following sections provide a more detailed description of its subsystems, grouped by their main functions: tracking, photon reconstruction, particle identification, and event triggering. Figure 6.2 shows a side view of the detector and its subsystems at the time of the LHC Run 2 period (2015–2018), after its trigger system was upgraded to accommodate the increased luminosity of the LHC [236; 237].

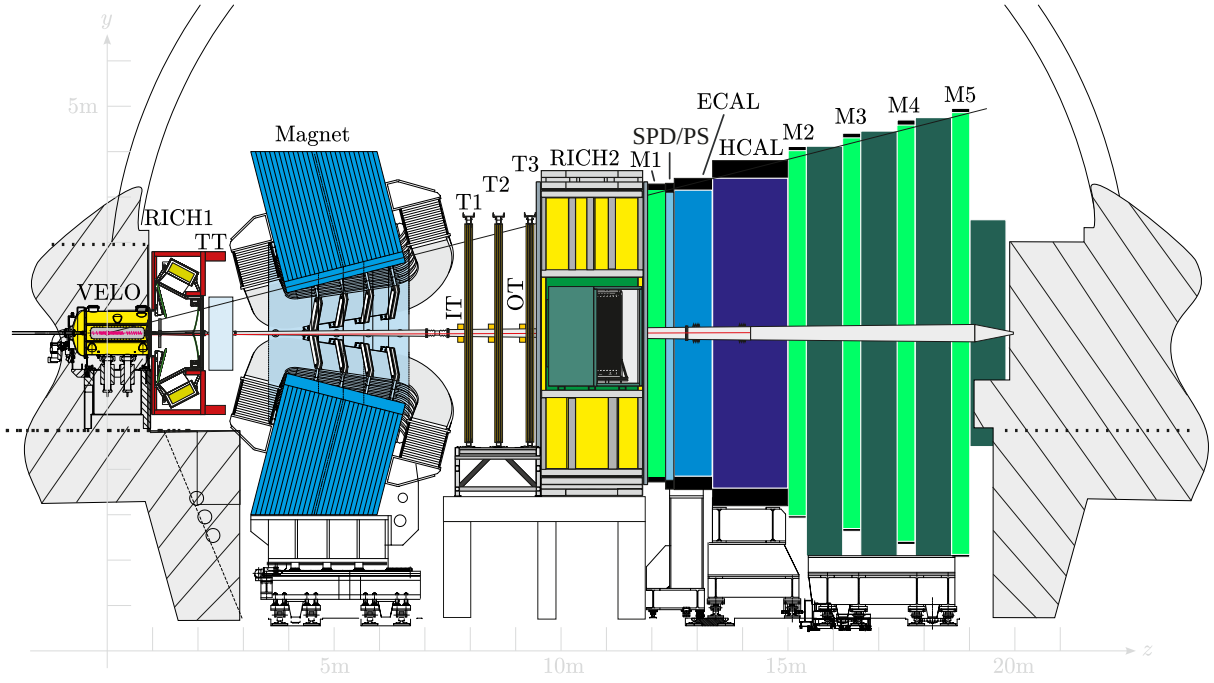


Figure 6.2. Side view of the LHCb detector. Adapted from [238; 239].

Tracking

Among all sub-detectors of the LHCb experiment, the **Vertex Locator (VELO)** is located most closely to the interaction point. It is designed to measure particle trajectories extremely close to the interaction region (in the order of a few millimetres), which makes it possible to accurately determine the primary (collision) and secondary (decay) vertices. The module consists of several disks of radial and azimuthal silicon sensor strips that are arranged in parallel along the beam pipe. The beam goes through the center of the disks. As such, each disk is split into two halves that are moved towards each other around the beam once it is in operation and stable [238, Fig. 5.1 and 5.6].

Prior to Upgrade 1 (LHC Run 2 and before) of the LHCb experiment (2018), two **Silicon Tracker (ST)** systems were located right after the VELO detector to further improve tracking efficiency. The **Tracker Turicensis (TT)** is located upstream of the dipole magnet and covers LHCb's full acceptance range. The TT is therefore used to measure the trajectory and momentum before they enter the magnetic field. Three planar tracking stations (T1, T2, T3) are located downstream of the magnet. They consist of silicon microstrips near the beam pipe (Inner Tracker, IT) and straw-tubes further away from the beam pipe (Outer Tracker, OT). While the IT is well-adapted at detecting high-energy, high-density particles, the OT has a better angular resolution for lower-energy particles that are further away from the beam pipe.

Photon reconstruction

Photons are reconstructed primarily with the **electromagnetic calorimeter (ECAL)**, which measures their energy deposition and impact position with fine granularity. Since photons leave no track in the upstream tracking detectors, the absence of a matching track is used to confirm their neutral nature. To further suppress background signals, the **Scintillating Pad Detector (SPD)** and **Pre-shower Detector (PS)**, placed directly in front of the ECAL, distinguish photons from electrons and reject background signals from charged and neutral pions. This system enables efficient photon reconstruction and the identification of neutral mesons such as $\pi^0 \to \gamma\gamma$.

Particle identification

Charged-particle identification is achieved with a combination of Cherenkov, calorimeter, and muon detectors. Two **Ring Imaging Cherenkov (RICH)** detectors measure the velocity of charged particles over a wide momentum range. **RICH 1**, located between the VELO and TT, covers low-momentum tracks (1–60 GeV/c) over the full angular acceptance using aerogel and C_4F_{10} radiators. **RICH 2**, located downstream after T3, identifies higher-momentum particles (15–100 GeV/c and beyond) with a CF_4 radiator and a reduced angular acceptance (15–120 mrad).

For electrons, the ECAL response is used in combination with the SPD and PS to provide a clean separation from hadrons, while hadrons deposit most of their energy in the downstream **hadronic calorimeter (HCAL)**, which is segmented into two zones with larger cell sizes to match their broader showers.

Finally, muons are identified in five dedicated stations (**M1**–**M5**) consisting of 1,380 muon chambers that are placed downstream of the calorimeters and interleaved with iron absorbers. Only penetrating particles with momenta above about 6 GeV/c reach these chambers. Station M1, located upstream of the calorimeters, improves the transverse-momentum resolution, while the downstream stations provide both identification and track-direction measurements.

Event triggering

The sub-detectors and trigger system of the LHCb experiment cannot handle the maximum design luminosity of the LHC. To reduce the luminosity, LHCb therefore crosses the beams at a larger angle. Prior to Upgrade 1, this resulted in an average luminosity of 2×10^{32} cm^{-2}s^{-1}. At the time, the LHCb experiment had a two-level trigger system. A fast, hardware-only **Level 0 (L0)** trigger system was used to reduce the event rate from 40 MHz to 1 MHz, which is the rate at which all sub-detectors could read out events. Next, the **High-Level Trigger (HLT)**

asynchronously processed the data from the sub-detectors to reduce the event rate to 2 kHz and store the selected events to disk. At L0, a Decision Unit processed data from the calorimeter system and muon system to identify clusters of high-p_T photons and charged particles. In addition, pile-up in the VELO was used to estimate the number of primary proton–proton interactions. The VELO and other tracking detectors were too slow for L0 and were processed at HLT. Since Run 3, the hardware L0 trigger has been removed and the experiment operates at a higher luminosity (2×10^{33} cm^{-2}s^{-1}) with a fully software-based trigger.

Data sets and software stack

The analyses presented in this work use data collected by the LHCb experiment during **Run 2** of the LHC, corresponding to proton–proton collisions at a center-of-mass energy of 13 TeV. The full Run 2 data set amounts to about 6 fb^{-1}, recorded between 2015 and 2018, with subsets selected according to the trigger requirements and detector conditions relevant for the studied channel. Over the years, LHCb has collected several other data sets. During **Run 1** (2010–2012), proton–proton collisions were recorded at center-of-mass energies of 7 and 8 TeV, corresponding to an integrated luminosity of about 3 fb^{-1}. More recently, **Run 3** started in 2022 at a nominal energy of 13.6 TeV, with the upgraded detector and trigger system designed to record datasets that will eventually exceed 20 fb^{-1}. Heavy-ion data from proton–lead and lead–lead collisions are also available, with smaller integrated luminosities.

Event simulation and reconstruction are performed with the LHCb software stack, built on the Gaudi framework [240]. The detector simulation application (Gauss) integrates Pythia for pp collisions, EvtGen for hadron decays, and Geant4 for the detector simulation. In a typical analysis, reconstructed tracks are combined into particle candidates, particle-identification information is used to assign their species, and a kinematic fit with vertex and mass constraints improves the resolution of the reconstructed variables. Based on these candidates, selection criteria are applied to suppress combinatorial and physics background channels, while aiming to retain high signal efficiency. To support this workflow, large inclusive Monte Carlo (MC) productions of heavy-flavour events, such as $b\bar{b}$ and $c\bar{c}$ production, are routinely generated; these contain all hadronisation and decay modes and allow any final state to be filtered. In addition, smaller exclusive signal MC samples are produced for specific decay topologies to optimise the selection criteria.

6.2. BESIII experiment

The Beijing Electron–Positron Collider II (BEPCII) is a double-ring electron–positron collider that operates at center-of-mass (CM) energies between 2 and 4.95 GeV. Originally, the BEPCII collider was designed with a maximum energy of 4.6 GeV. To investigate more XYZ states, such as $Z_{cs}(3985)^+$, $Y(4630)$, and $Y(4660)$, BEPCII was upgraded to have a maximum energy of 4.95 GeV [241].

The **Beijing Electron Spectrometer (BES)** experiment is located at the Beijing campus of the Institute of High Energy Physics (IHEP) in China and is part of the BEPC facility (Figure 6.3). BES has gone through three generations since 1988, with the current detector, **BESIII**, beginning operation in 2008 alongside the BEPCII upgrade.

The BESIII experiment explores the physics of the charm–tau sector in the energy range accessible at BEPCII [244]. A large part of its program is devoted to hadron spectroscopy, as it offers high-statistics samples of charmonium, open-charm states, and light hadrons. BESIII

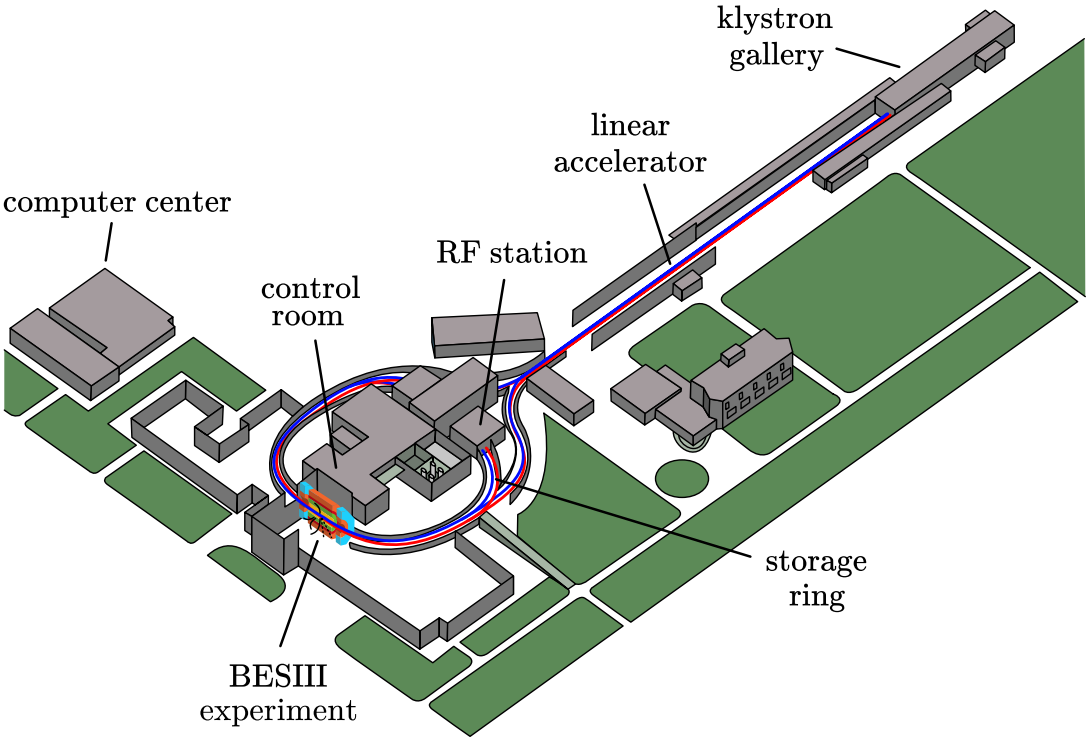


Figure 6.3. The Beijing Electron–Positron Collider II with the BESIII experiment located at the storage ring. Adapted from [242; 243, Fig. 4.1].

has also become a key source of precise cross-section data for processes such as $e^+e^- \to$ hadrons. These serve as inputs to Standard Model calculations, including the hadronic contribution to the muon anomalous magnetic moment. Alongside spectroscopy and cross-section measurements, the experiment investigates charm decays and mixing, τ lepton properties, rare and forbidden processes, and a variety of tests of discrete symmetries.

To carry out this physics program, the BESIII detector is built as a general-purpose magnetic spectrometer arranged cylindrically around the BEPCII interaction point (Figure 6.4), so that it has wide angular coverage. From the inside outward, the detector consists of a Multilayer Drift Chamber (MDC), a Time-of-Flight (TOF) system, an Electromagnetic Calorimeter (EMC), a superconducting solenoid magnet that provides a 1 T axial field, and outer Muon Chambers (MUC). Most subsystems are divided into a cylindrical "barrel" around the beam pipe and two "endcaps" covering the forward and backward regions. The following sections describe these detector subsystems in more detail, grouped by their main functions: tracking, photon reconstruction, particle identification, and event triggering.

The storage ring of BEPCII consists of two separate beam pipes, with positrons stored in one and electrons in the other. At the interaction point, the two beams cross at an angle of 11 mrad, which results in a small transverse momentum to the CM system of the colliding electron–positron pair. The beam axis defines the z axis of the lab frame coordinate system, while the crossing plane sets the x direction along the net CM momentum; the y axis is taken perpendicular to the xz plane. In particular, for a beam energy tuned to the J/ψ mass, so that $E_{\text{beam}} = m_{J/\psi}$, the four-momentum of the CM system in the laboratory frame is

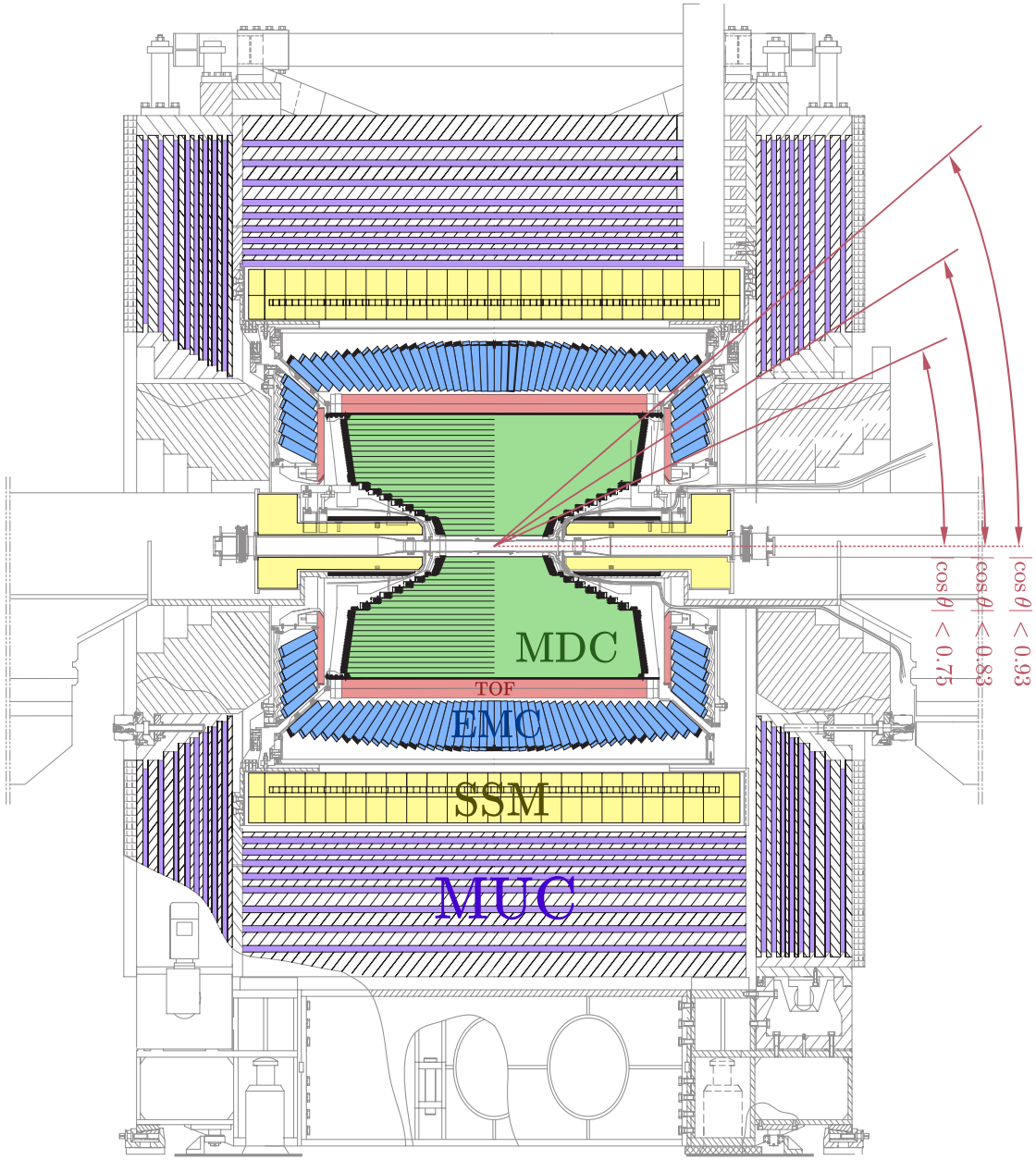


Figure 6.4. BESIII detector geometry, adapted from [245].

$$p_{\text{beam}}^{\text{lab}} = \left(\sqrt{m_{J/\psi}^2 + p_x^2},\, p_x,\, 0,\, 0\right), \quad \text{with } p_x \approx 34 \text{ MeV}/c\,. \tag{6.1}$$

This small asymmetry is important for amplitude analyses, since reconstructed four-momenta must be boosted from the laboratory to the CM frame, where the beam system is at rest, such that $p_{\text{beam}} = \left(m_{J/\psi}, 0, 0, 0\right)$.

Tracking

Charged-particle tracking at BESIII is provided by the **Multilayer Drift Chamber (MDC)**, which forms the inner core of the detector. With an inner radius of 59 mm and an outer radius of 810 mm, it is closest to the interaction point and therefore well suited for reconstructing low-momentum tracks and determining their ionisation energy loss (dE/dx) for particle identification. The MDC consists of 43 layers of small drift cells filled with a $\text{He}:\text{C}_3\text{H}_8$ 60:40 gas mixture, chosen to minimise multiple scattering and optimise momentum and dE/dx resolution. Each cell contains a 25 μm gold-plated tungsten sense wire surrounded by eight 110 μm aluminium field wires. As charged particles traverse the chamber, they ionise the gas, and the resulting signals on the sense wires provide position measurements with a resolution of about 130 μm in the radial direction and 2 mm along the wires. The conical shape at the endcaps accommodates the final focusing quadrupoles and results in a solid-angle coverage of about 93% of 4π.

Momentum measurement is enabled by the **Superconducting Solenoid Magnet (SSM)**, which surrounds the MDC, TOF, and EMC [246]. It provides a uniform axial magnetic field of 1.0 T, bending the trajectories of charged particles in the MDC to determine their momenta. The magnet is enclosed by an iron yoke, which both enhances the magnetic field and acts as an absorber for separating muons from hadrons in the outer muon system.

Photon reconstruction

Photons at BESIII are reconstructed in the **Electromagnetic Calorimeter (EMC)**, which surrounds the TOF system at a radius of about 94 cm. The EMC consists of 6,240 cesium iodide crystals doped with thallium (CsI(Tl)) crystals arranged in 56 disks pointing toward the interaction point, with a total mass of about 25.6 tons. When photons or electrons enter the crystals, they initiate electromagnetic showers through bremsstrahlung and pair production. The secondary particles deposit their energy in the crystals, from which the position and energy of the incident particle are reconstructed. The large numbers of dense crystal disks allows for a spatial resolution of $\sigma = 0.6 \text{ cm}/\sqrt{E}$.

Given the relatively low CM energies at BEPCII (at most 4.95 GeV), photons and electrons can carry only a few MeV of energy. In addition, common processes like $\pi^0 \to \gamma\gamma$ result in a large opening angle between the two photons, which requires a good position resolution to make it possible to perform cuts on the opening angle. The EMC therefore also plays an important role in the identification of neutral mesons.

Particle identification

Particle identification at BESIII relies on a combination of energy loss measurement, timing, and muon detection. The dE/dx profile measured in the MDC provides separation of particle species at low momenta. This information is complemented by the **Time-of-Flight (TOF)** system, which surrounds the MDC. The TOF barrel consists of two layers of 88 plastic scintillator bars

located at radii of 81 and 86 cm, while each endcap contains a single layer of 48 bars at 1.4 m from the interaction point. With a time resolution of about 100 ps, the TOF achieves a 3σ separation between pions and kaons up to momenta of roughly 770 MeV/c, and, in combination with dE/dx, enables identification of electrons, muons, pions, kaons, and protons across a wide momentum range.

The **Muon Chambers (MUC)** form the outermost subsystem of BESIII. They consist of resistive plate counters (RPCs) embedded in the iron return yoke of the superconducting magnet, at radii between 170 and 262 cm. The RPCs detect ionisation signals left by penetrating charged particles, which are matched to tracks reconstructed in the MDC. Since muons interact only weakly in the calorimeter and magnet yoke, while charged pions are largely absorbed, the MUC provides reliable muon identification above a momentum threshold of about 0.4 GeV/c.

Event triggering

The BESIII trigger operates in two stages: a **Level-1 (L1) hardware trigger** and a **Level-3 (L3) software trigger**. At L1, fast signals from the MDC, TOF, and EMC are processed by a global trigger logic implemented in FPGA-based electronics, running synchronously with the 41.65 MHz bunch crossing frequency of BEPCII. This reduces the event rate to about 4 kHz at the J/ψ resonance. The surviving events are then passed to the L3 software trigger, which applies more refined selections before writing events to permanent storage.

Data sets and software stack

The BESIII experiment has accumulated large data sets at many center-of-mass energies between 2 and 4.95 GeV, corresponding to an integrated luminosity of more than $10\,\text{fb}^{-1}$ since 2009. Dedicated runs have been taken at narrow resonances such as the J/ψ, $\psi(2S)$, and $\psi(3770)$, as well as at higher energies in the charmonium-like XYZ region. These samples provide the basis for amplitude analyses and spectroscopy studies of charmonium, open-charm, and light hadron final states.

Event reconstruction and simulation are performed with the BOSS (BESIII Offline Software System) framework, which provides event generation, detector simulation, and reconstruction. The initial e^+e^- collision is generated with KKMC to include effects from initial-state radiation. Subsequent hadron decays are modelled with EvtGen, while the interaction of particles with the detector material is simulated using Geant4. Reconstructed tracks are combined into particle candidates, whose identities are assigned using particle-identification information from the MDC, TOF, and EMC. A kinematic fit imposing vertex and energy–momentum constraints is then applied to improve resolution and ensure consistency with the known collision energy. Event selection is performed using a set of kinematic and particle-identification criteria designed to suppress background channels while retaining high efficiency for the targeted decay mode.

Large inclusive MC samples are regularly generated for specific beam energies in which the e^+e^- annihilation and subsequent decays of the relevant resonances (J/ψ, $\psi(2S)$, open-charm states, etc.) are simulated according to the world-average branching fractions. These serve as the main tool for estimating background channels and efficiencies. In addition, exclusive MC samples are produced for specific decay topologies to optimise selections and validate the analysis chain. Finally, phase-space samples are generated for multi-body decays without detailed dynamical input, which are needed as efficiency-corrected domain sample for amplitude analysis models.

7 Application to data

This chapter demonstrates the application of the formalisms and computational techniques developed in the preceding chapters. The aim is to show that the tools and workflows can be applied in different contexts and are accurate and performant. First, by comparing against an existing amplitude analysis by LHCb, we validated the implementation of the Dalitz-plot decomposition (DPD) method of Section 3.3. The new symbolic implementation enabled the computation of a polarimeter vector field for the decay $\Lambda_c^+ \to pK^-\pi^+$, the results of which were published in [247]. We then illustrate how CAS-assisted model building the DPD method is employed to fit an amplitude model to $J/\psi \to \bar{p}K_S^0\Sigma^+$ data from the BESIII experiment. Finally, we assess the performance of symbolic amplitude models on different hardware configurations when using Minuit2 for fitting.

7.1. Polarimeter vector field of Λ_c

So far, we have considered spin projections to be a kinematic degree of freedom that we factor out in amplitude models through the helicity formalism, because it is not relativistically invariant. However, particles can exhibit a preferred spin orientation that is *characteristic* for how it is produced. This phenomenon is known as **polarisation**.

Particle polarisation can appear as a forward–backward asymmetry in the angular distributions of decay products. As such, it is an observable that is widely used in hadron spectroscopy studies. For hadrons, polarisation provides insights into the hadronisation process by which quarks and gluons combine to form the hadron. Although hadronisation is a non-perturbative phenomenon, the polarisation of the produced hadron preserves information about the quarks' spin orientation during this process. This retention of spin information therefore offers a window into how spin is transferred or altered during hadron formation [248–250].

In semileptonic decays, the polarisation of the produced hadrons can be used to study the handedness of weak interactions. Since the weak force couples only to left-handed particles (and right-handed antiparticles), the angular asymmetry of the produced hadrons can reveal details about the chiral structure of the interaction. This sensitivity makes hadron polarisation a probe for testing the Standard Model and searching for possible signs of new physics. For example, deviations from the expected polarisation patterns could indicate contributions from right-handed currents or non-standard couplings, which are predicted in various beyond-the-Standard Model scenarios [251–259].

Asymmetry parameter

In two-body decays, the forward–backward asymmetry appears in the (normalised) differential decay rate $\mathrm{d}\Gamma$ when expressed as a function of the scattering angle θ,

Figure 7.1. Definition of the forward–backward asymmetry for a $\Lambda \to p\pi$ decay, starting from the lab frame (left) and boosting into the center-of-mass frame (right) to get the angle θ between the decay product and the parent particle.

$$\frac{2}{\Gamma}\frac{\mathrm{d}\Gamma}{\mathrm{d}\cos\theta} = 1 + P\alpha\cos\theta\,, \tag{7.1}$$

where P denotes the polarisation of the decaying particle and α is an **asymmetry parameter** characteristic of the decay. The definition of the scattering angle θ is shown in Figure 7.1, for the decay $\Lambda \to p\pi$ as an example.

For a spin-half or spin-1 parent decaying into a spin-half baryon and a pseudoscalar meson, the decay amplitude contains only two independent partial waves: an S-wave ($L = 0$) and a P-wave ($L = 1$). These correspond to the parity-violating and parity-conserving contributions, usually denoted H_S and H_P. The asymmetry parameter is determined by their interference:

$$\alpha = \frac{2\,\mathrm{Re}(H_S^* H_P)}{|H_S|^2 + |H_P|^2}\,. \tag{7.2}$$

The decay therefore only has an angular asymmetry if both parity-conserving and parity-violating amplitudes are present.

Polarimetry

Since the asymmetry parameter α is an intrinsic decay property that does not depend on the production mechanism, it can be used to combine polarisation information between different analyses. Once an asymmetry parameter is determined, it can be used to determine the polarisation of the decaying particle in other analyses. For example, in $\Xi_b^0 \to J/\psi p\Lambda$, the asymmetry parameter of $\Lambda \to p\pi^-$ provides sensitivity to the spin of the pentaquark candidate $P_{cs} \to J/\psi\,\Lambda$ [260].

However, heavy baryons often do not predominantly decay to two-body final states, which makes it difficult to measure the asymmetry parameter with high precision. For example, in $\Lambda_b^0 \to \Lambda_c^+ \bar{D}^0 K^-$ decays, pentaquarks are expected in the $\Lambda_c^+ \bar{D}^0$ final state, but the Λ_c^+ is reconstructed through the predominant three-body decay $\Lambda_c^+ \to pK^-\pi^+$. The lack of polarisation information makes it harder to identify pentaquarks in the angular distributions.

Lepton studies in the 1990s introduced the concept of a **polarimeter vector** $\vec{h}$ in order to characterise the asymmetry in τ lepton decays [261]. Just like the asymmetry parameter in Equation (7.1), the polarimeter vector allows formulating the angular asymmetry as a function of the **polarisation vector** $\vec{P}$ of the decaying τ lepton as

$$\frac{1}{\Gamma}\frac{\mathrm{d}\Gamma}{\mathrm{d}\Phi} \propto 1 + \vec{P}\cdot\vec{h}\,, \tag{7.3}$$

with $\mathrm{d}\Phi$ describing all remaining kinematic degrees of freedom and Γ the total decay width of the τ lepton. This simple relation only works in the case of a decaying τ lepton, because its spin is transferred to the neutrino in the final state, leaving the other decay products unpolarised. In the case of decaying hadrons, the decay products are polarised, which complicates the interpretation of the angular asymmetry. The polarimeter vector is then not a constant, but a function of the kinematic variables of the decay.

It turns out that the concept of the polarimeter vector can be extended to three-body decays of spin-half hadrons by introducing an **aligned polarimeter vector** $\vec{\alpha}$ [247]. By aligning the polarimeter to the decay plane by a rotation $\boldsymbol{R}(\phi,\theta,\chi)$ (see Figure 7.2), we get a vector that, like the asymmetry parameter, is intrinsic to the decay. The polarimeter vector of equation Equation (7.3) is then given by

$$\vec{h} \;=\; \boldsymbol{R}(\phi,\theta,\chi)\,\vec{\alpha}\,, \tag{7.4}$$

which results in a more general expression for Equation (7.1),

$$\frac{1}{\Gamma}\frac{\mathrm{d}\Gamma}{\mathrm{d}\Phi} \propto 1 + \vec{P}\,\boldsymbol{R}(\phi,\theta,\chi)\,\vec{\alpha}(\tau)\,. \tag{7.5}$$

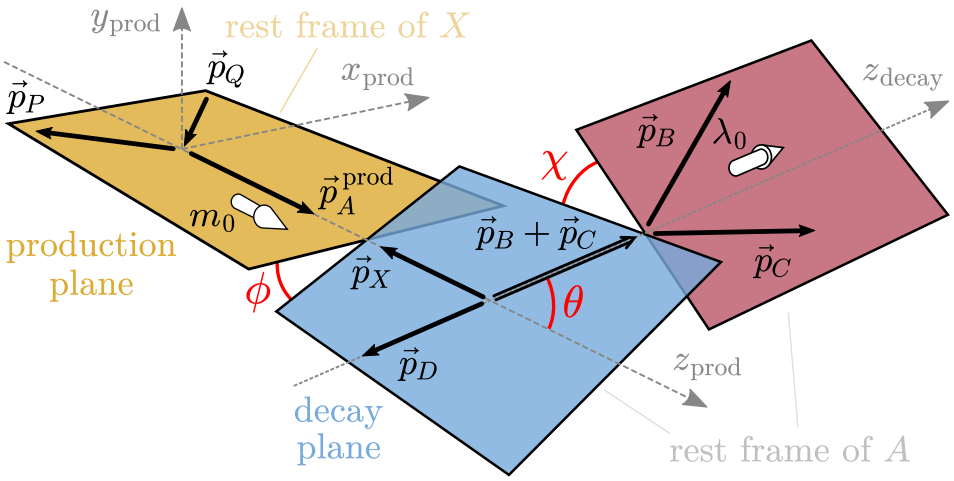


Figure 7.2. Definition of the Euler angles ϕ, θ, χ that align the polarimeter vector $\vec{\alpha}$ to the decay plane. Adapted from [247].

Just like in Equation (7.2), the aligned polarimeter vector can be determined from the helicity amplitudes, but these amplitudes need to be aligned consistently with regard to the final state. For this, we employ the methods described in Section 3.3, which enable us to relate the rotation of the aligned polarimeter vector in Equation (7.4) to that of the rotation of the aligned transition amplitudes.

As discussed in Section 3.3, the polarised intensity function for any n-body decay can be expressed in terms of amplitudes $\widetilde{A}_{m_0,\{\lambda\}}$ (Equation (3.22)) that are aligned with regard to the production quantisation axis over Euler angles $\Omega = (\phi,\theta,\chi)$, giving

$$I(\Omega,\tau) \;=\; \sum_{\{\lambda\}} \sum_{m_0,m_0'} \widetilde{A}^{*}_{m_0',\{\lambda\}}(\Omega,\tau)\, \rho_{m_0' m_0}\, \widetilde{A}_{m_0,\{\lambda\}}(\Omega,\tau)\,, \tag{7.6}$$

with $m_0^{(\prime)}$ the allowed spin projections of the decaying hadron along that quantisation axis determined by the production process and $\{\lambda\}$ the set of allowed helicities $\lambda_1, \lambda_2, \ldots, \lambda_n$ of the decay products. If decaying particle 0 is a spin-half fermion, the spin-density matrix $\boldsymbol{\rho}$ depends on its polarisation vector $\vec{P}$ via

$$\rho_{m_0' m_0} \;=\; 1 + \vec{P}\cdot\vec{\sigma}_{m_0' m_0}\,, \tag{7.7}$$

where $\vec{\boldsymbol{\sigma}}$ denotes the three Pauli matrices $\boldsymbol{\sigma_x}, \boldsymbol{\sigma_y}, \boldsymbol{\sigma_z}$. By inserting Equation (3.22) into Equation (7.6), we get

$$\begin{aligned} I(\Omega,\tau) \;&=\; \sum_{m_0,m_0'} \sum_{\lambda_0,\lambda_0'} \rho_{m_0' m_0}\; D^{1/2}_{m_0'\lambda_0'}(\phi,\theta,\chi)\; D^{1/2\,*}_{m_0\lambda_0}(\phi,\theta,\chi)\; \mathfrak{X}_{\lambda_0'\lambda_0}(\tau)\,, \\ \mathfrak{X}_{\lambda_0'\lambda_0}(\tau) \;&\equiv\; \sum_{\{\lambda\}} A^{*}_{\lambda_0',\{\lambda\}}(\tau)\, A_{\lambda_0,\{\lambda\}}(\tau)\,. \end{aligned} \tag{7.8}$$

By defining the vector field

$$\begin{aligned} \vec{\alpha}(\tau) \;&=\; \sum_{\lambda_0',\lambda_0,\{\lambda\}} A^{*}_{\lambda_0',\{\lambda\}}(\tau)\, \vec{\sigma}_{\lambda_0',\lambda_0}\, A_{\lambda_0,\{\lambda\}}(\tau) \Big/ I_0(\tau)\;, \\ I_0(\tau) \;&\equiv\; \sum_{\lambda_0,\{\lambda\}} \left| A_{\lambda_0,\{\lambda\}}(\tau) \right|^2\,, \end{aligned} \tag{7.9}$$

we can rewrite Equation (7.8) as

$$I(\Omega,\tau) \;=\; I_0(\tau)\, \left(1 + \sum_{i,j} P_i\, R_{ij}(\phi,\theta,\chi)\, \alpha_j(\tau) \right)\,. \tag{7.10}$$

This transformation is enabled by a homomorphism [262],

$$\frac{1}{2}\mathrm{Tr}\left[\boldsymbol{D}^{1/2\,*}(\phi,\theta,\chi)\, \vec{\boldsymbol{\sigma}}_i\, \boldsymbol{D}^{1/2}(\phi,\theta,\chi) \right] \;=\; R_{ij}(\phi,\theta,\chi)\,,$$

which can be used to relate the SU(2) spin rotation matrices in Equation (7.8) to the SO(3) rotation matrix in Equation (7.4) using the Pauli matrices that come from Equation (7.7). We have verified this relation algebraically with a CAS [263].

We can derive a similar relation as found in Equation (7.5) by integrating Equation (7.10) over the remaining kinematic variables τ. This gives us

$$\frac{8\pi^2}{\Gamma} \frac{\mathrm{d}^3\Gamma}{\mathrm{d}\phi\, \mathrm{d}\cos\theta\, \mathrm{d}\chi} \;=\; 1 + \sum_{i,j} P_i\, R_{ij}(\phi,\theta,\chi)\, \overline{\alpha}_j\,, \tag{7.11}$$

where we have defined an **averaged polarimeter vector** $\overline{\alpha}$ as

$$\overline{\alpha}_j \;=\; \int I_0(\tau)\, \alpha_j(\tau)\, \mathrm{d}^n\tau \Big/ \int I_0(\tau)\, \mathrm{d}^n\tau\;. \tag{7.12}$$

Naturally, the use of the averaged polarimeter vector rather than a field simplifies reuse in other analyses. However, using this averaged polarimeter vector results in an increase of statistical uncertainty.

Both the polarimeter vector field $\vec{\alpha}(\tau)$ of decaying fermion 0, as well as its averaged polarimeter vector $\overline{\alpha}$, can be used to extend amplitude models that involve this fermion in their decay chains. Rewriting $\mathfrak{X}_{\lambda_0'\lambda_0}(\tau)$ from Equation (7.8) in terms of $\vec{\alpha}$ and the unpolarised intensity I_0 as

$$\mathfrak{X}_{\lambda_0'\lambda_0}(\tau) \;=\; \frac{I_0(\tau)}{2}\,(1 + \vec{\alpha}(\tau)\cdot\vec{\boldsymbol{\sigma}})_{\lambda_0'\lambda_0}\;, \tag{7.13}$$

it can be directly inserted into another amplitude model. In our case, with $\vec{\alpha}$ computed for Λ_c^+, the field can for instance be used to construct an amplitude for the decay $B^+ \to \Lambda_c^+ \bar{\Lambda}_c^- K^+$. This gives

$$\begin{aligned} I(\sigma_{\Lambda_c K^+}, \sigma_{\bar{\Lambda}_c K^+}; \phi, \theta, \chi, \sigma_{pK^-}, \sigma_{K^-\pi^+}; \bar{\phi}, \bar{\theta}, \bar{\chi}, \sigma_{\bar{p}K^+}, \sigma_{K^+\pi^-}) \;=\; \\ \sum_{m_0,\bar{m}_0} \sum_{m_0',\bar{m}_0'} A^{B^+*}_{m_0',\bar{m}_0'}(\sigma_{\Lambda_c K^+}, \sigma_{\bar{\Lambda}_c K^+})\, A^{B^+}_{m_0,\bar{m}_0}(\sigma_{\Lambda_c K^+}, \sigma_{\bar{\Lambda}_c K^+}) \\ \times \sum_{\lambda_0,\lambda_0'} D^{1/2\,*}_{m_0\lambda_0}(\phi,\theta,\chi)\, D^{1/2}_{m_0'\lambda_0'}(\phi,\theta,\chi)\, \mathfrak{X}^{\Lambda_c}_{\lambda_0'\lambda_0}(\sigma_{pK^-}, \sigma_{K^-\pi^+}) \\ \times \sum_{\bar{\lambda}_0,\bar{\lambda}_0'} D^{1/2\,*}_{\bar{m}_0\bar{\lambda}_0}(\bar{\phi},\bar{\theta},\bar{\chi})\, D^{1/2}_{\bar{m}_0'\bar{\lambda}_0'}(\bar{\phi},\bar{\theta},\bar{\chi})\, \mathfrak{X}^{\bar{\Lambda}_c}_{\bar{\lambda}_0'\bar{\lambda}_0}(\sigma_{\bar{p}K^+}, \sigma_{K^+\pi^-})\,. \end{aligned} \tag{7.14}$$

The insertion of the $\mathfrak{X}$ matrices adds 10 more degrees of freedom to the $B^+ \to \Lambda_c^+ \bar{\Lambda}_c^- K^+$ amplitude model, indicated in blue. However, by evaluating $\mathfrak{X}$ using Equation (7.13) with the $\vec{\alpha}$ field that we provide in this study (with interpolation from a sampled grid), the degrees of freedom are fixed, which removes a large number of fit parameters, but does propagate spin information from the Λ_c^+ decay to the B^+ decay and increases the sensitivity of the model to the polarisation of the Λ_c^+.

Application to $\Lambda_c^+ \to pK^-\pi^+$

The key point is that we now have an way to compute the polarimeter vector field $\vec{\alpha}$ from transition amplitudes $A_{\lambda^0,\{\lambda\}}$ that we can determine in a partial-wave analysis. We applied this technique to compute the aligned polarimeter vector field for the decay $\Lambda_c^+ \to pK^-\pi^+$. Using the fit parameters of 18 amplitude models [264], we computed the polarimeter vector field α by reformulating those amplitude models with the DPD method. The amplitude model with the best fit to the data is named the "default" model and was used to compute the reported mean values. Systematic and statistical uncertainties were propagated to the polarimeter vector field by evaluating the polarimeter vector field over the different models and over the uncertainties of the fit parameters of the nominal model, respectively.

The amplitude analysis from which the fit parameters were extracted, was performed using data from Run 2 of the Large Hadron Collider (LHC). The sample consists of proton–proton collisions at a center-of-mass energy of 13 TeV, recorded in 2016 with an integrated luminosity of 1.7 fb^{-1}.

Intensity distributions and decay-rate matrices

Figure 7.3 shows a comparison between the data distribution over the Dalitz plane and our reimplementation of the original default amplitude model in terms of DPD. We verified that the

intensity distribution of all 18 amplitude models formulated symbolically with the DPD method is the same, up to floating-point precision [263]. A second cross-check was performed with a DPD implementation in Julia for the default model [265].

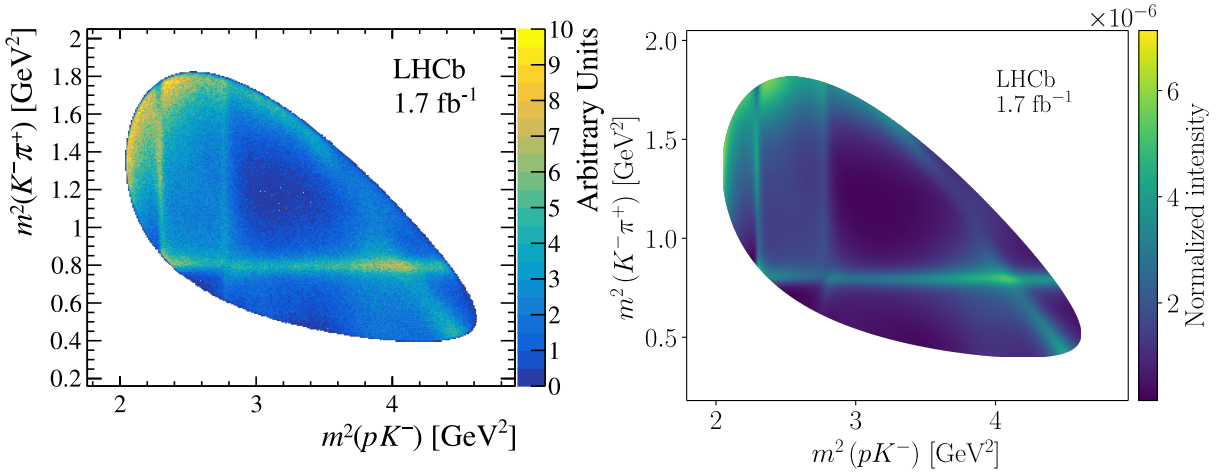


Figure 7.3. Comparison between a rendering of the data distribution of $\Lambda_c \to pK^-\pi^+$ in the Dalitz plane measured by LHCb (left) and the intensity distribution of the "default" DPD amplitude model, evaluated over a 1000×1000 grid (right). Adapted from [264; 247].

The $\Lambda_c^+ \to pK^-\pi^+$ decay has three subsystems. We indicate the set of resonances in each subsystem with **, giving us the subsystems $K^{**} \to K^-\pi^+$, $\Lambda^{**} \to pK^-$, and $\Delta^{**} \to p\pi^-$. The resonance contributions of each of these subsystems are clearly visible (see Figure 7.4).

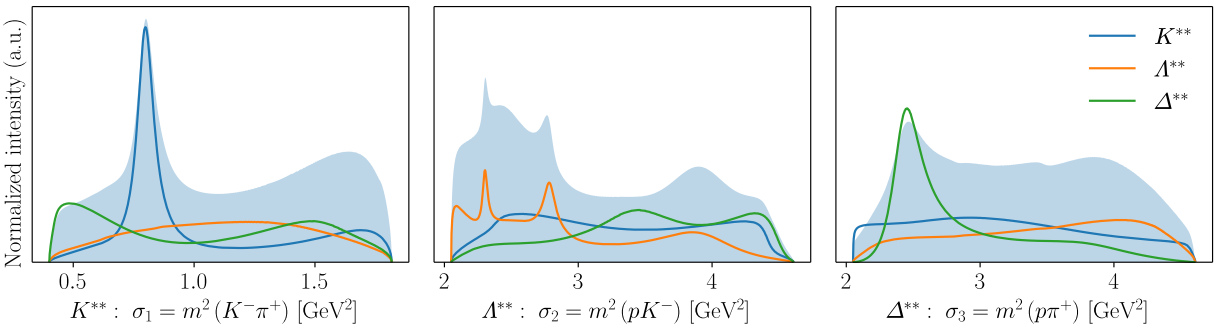


Figure 7.4. Projections of the intensity distribution for the "default" DPD amplitude model of $\Lambda_c^+ \to pK^-\pi^+$ to the invariant mass of the subsystems $K^-\pi^+$ (left), pK^- (middle), and $p\pi^+$ (right).

As a validation step, we computed the fit fractions of the default model with full uncertainty propagation and compared them with the results reported in [264, Table X]. The values in Table 7.1 show that the fit fractions obtained in this polarimetry study are consistent within uncertainties with those from the original LHCb amplitude analysis. Further details of the uncertainty propagation are discussed later.

Resonance	Polarimetry study	Original analysis
$\Lambda(1405)$	$7.78 \pm 0.43^{+3.01}_{-2.53}$	$7.7 \pm 0.2 \pm 3.0$
$\Lambda(1520)$	$1.91 \pm 0.10^{+0.04}_{-0.24}$	$1.86 \pm 0.09 \pm 0.23$
$\Lambda(1600)$	$5.16 \pm 0.28^{+0.50}_{-1.93}$	$5.2 \pm 0.2 \pm 1.9$
$\Lambda(1670)$	$1.15 \pm 0.04^{+0.06}_{-0.29}$	$1.18 \pm 0.06 \pm 0.32$
$\Lambda(1690)$	$1.16 \pm 0.01^{+0.06}_{-0.33}$	$1.19 \pm 0.09 \pm 0.34$
$\Lambda(2000)$	$9.55 \pm 0.67^{+0.83}_{-2.26}$	$9.58 \pm 0.27 \pm 0.93$
$\Delta(1232)^{++}$	$28.73 \pm 1.34^{+1.76}_{-0.79}$	$28.6 \pm 0.29 \pm 0.76$
$\Delta(1600)^{++}$	$4.50 \pm 0.51^{+0.93}_{-1.40}$	$4.5 \pm 0.3 \pm 1.5$
$\Delta(1700)^{++}$	$3.89 \pm 0.07^{+0.94}_{-0.48}$	$3.9 \pm 0.2 \pm 0.94$
$K_0^*(700)$	$2.99 \pm 0.20^{+0.91}_{-0.59}$	$3.02 \pm 0.16 \pm 0.92$
$K^*(892)$	$21.95 \pm 1.24^{+0.59}_{-0.70}$	$22.14 \pm 0.23 \pm 0.64$
$K_0^*(1430)$	$14.70 \pm 0.80^{+2.78}_{-2.67}$	$14.7 \pm 0.6 \pm 2.7$

Table 7.1. Comparison of the fit fractions of the resonances in the decay $\Lambda_c^+ \to pK^-\pi^+$ between this polarimetry study and the original LHCb amplitude analysis [264]. The first quoted uncertainty combines the statistical and systematic components in quadrature, the second reflects the model dependence. In this work, the model uncertainty is defined by the extrema of the deviations, rather than by their standard deviation.

The decay rates of the amplitude models were also investigated. In particular, the model using that uses LS couplings (canonical basis) rather than helicity couplings allows one to distinguish partial waves as either parity-violating or parity-conserving, which can be related to Equation (7.2). Figure 7.5 shows the decay rates of all partial waves in the canonical basis. Each matrix element i, j is obtained by setting all couplings to zero except those of the ith and jth partial waves, and then normalising to the total intensity I_{tot} of the full model. Diagonal elements represent the decay rate I_i/I_{tot} (the "fit fraction") of the ith partial wave, rather than the resonance fit fractions listed in Table 7.1. Off-diagonal elements quantify interference between the ith and jth partial waves, given by $(I_{ij} - I_i - I_j)/I_{\text{tot}}$. Interference should vanish between different partial waves (different LS combinations) and where it does not, this indicates correlations between amplitudes induced by the production mechanism. By construction, the sum of all elements in the lower triangle of the matrix including the diagonal equals 100%.

Field visualisations

Having reformulated and validated the amplitude models with DPD, we can compute the polarimeter vector field from the aligned transition amplitudes using Equation (7.9). The alignment can be performed with respect to any of the three subsystems. In Figure 7.2, this corresponds to choosing a different pair of decay products to define the red plane. As a result, one obtains three distinct polarimeter vector fields $\vec{\alpha}$, associated with the K^{**}, Λ^{**}, and Δ^{**} subsystems. However, the polarised intensity as given in Equation (7.10), remains invariant under this choice, owing to the rotation $\boldsymbol{R}$ into the decay plane.

Figure 7.6 shows these three aligned polarimeter vector fields over the Dalitz plane as computed from the default amplitude model. The diagram in the upper right shows the decay plane in the center-of-mass frame of Λ_c and the red arrow highlights subsystem to which the vector field has

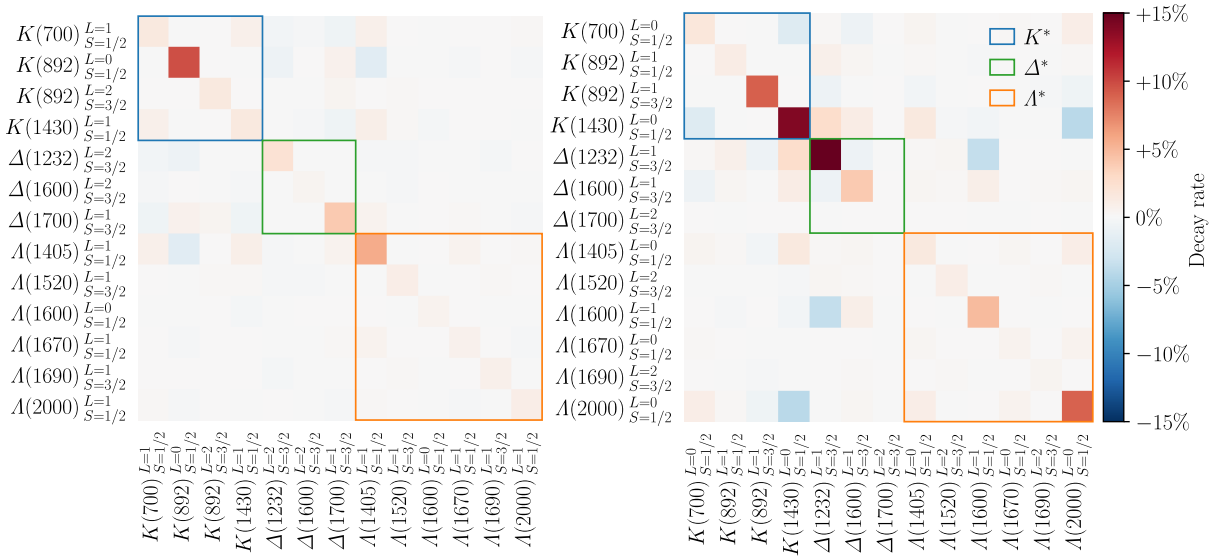


Figure 7.5. Decay rates of all partial waves in the model formulated in the canonical basis. A distinction is made between parity-violating (left) and parity-conserving (right) currents.

been aligned. The α_x component of the field is projected onto the vertical axis of the plot and the α_z component is projected onto the horizontal axis. The y component of the vector field is of less interest, as the y axis points out of the decay plane (see Figure 7.2). The colours of the vectors indicate their length.

The shaded areas indicate where each of the subsystems contribute for more than 70% to the intensity (some regions overlap due to interference). One would expect that the vector field points in the z direction in regions where the subsystem to which the vector field has been aligned, dominates. However, it is difficult to determine whether the vector field consistently points in a single direction, even in those regions.

The polarimeter vector field does behave consistently when we evaluate it for each of the resonances individually (by setting the couplings of other amplitudes to zero). Figure 7.7 shows the polarimeter vector field for each resonance, using the corresponding subsystem as reference for alignment. In almost each case, the vector field aligns perfectly to the z axis of the decay plane (that is, in the direction of the resonance). For the Λ^{**} and Δ^{**} subsystems, this is because their recoil particle is a spin-zero meson, so that all spin information is transferred to the resonance. For the K^{**} subsystem, the vector field only aligns with the z axis in the case of the scalar mesons $K_0^*(700)$ and $K_0^*(1430)$. For the $K^*(892)$ resonance, the polarimeter field does not align, because it is a vector meson.

When the polarimeter vector field aligns consistently for a specific resonance and its recoil, we essentially have the situation sketched in Figure 7.1. The modulus of the averaged polarimeter vector from Equation (7.12) is indicated in the top right of Figure 7.7. The value is characteristic to that resonance and its recoil and indicates the strength of the forward–backward asymmetry in the decay via Equation (7.1).

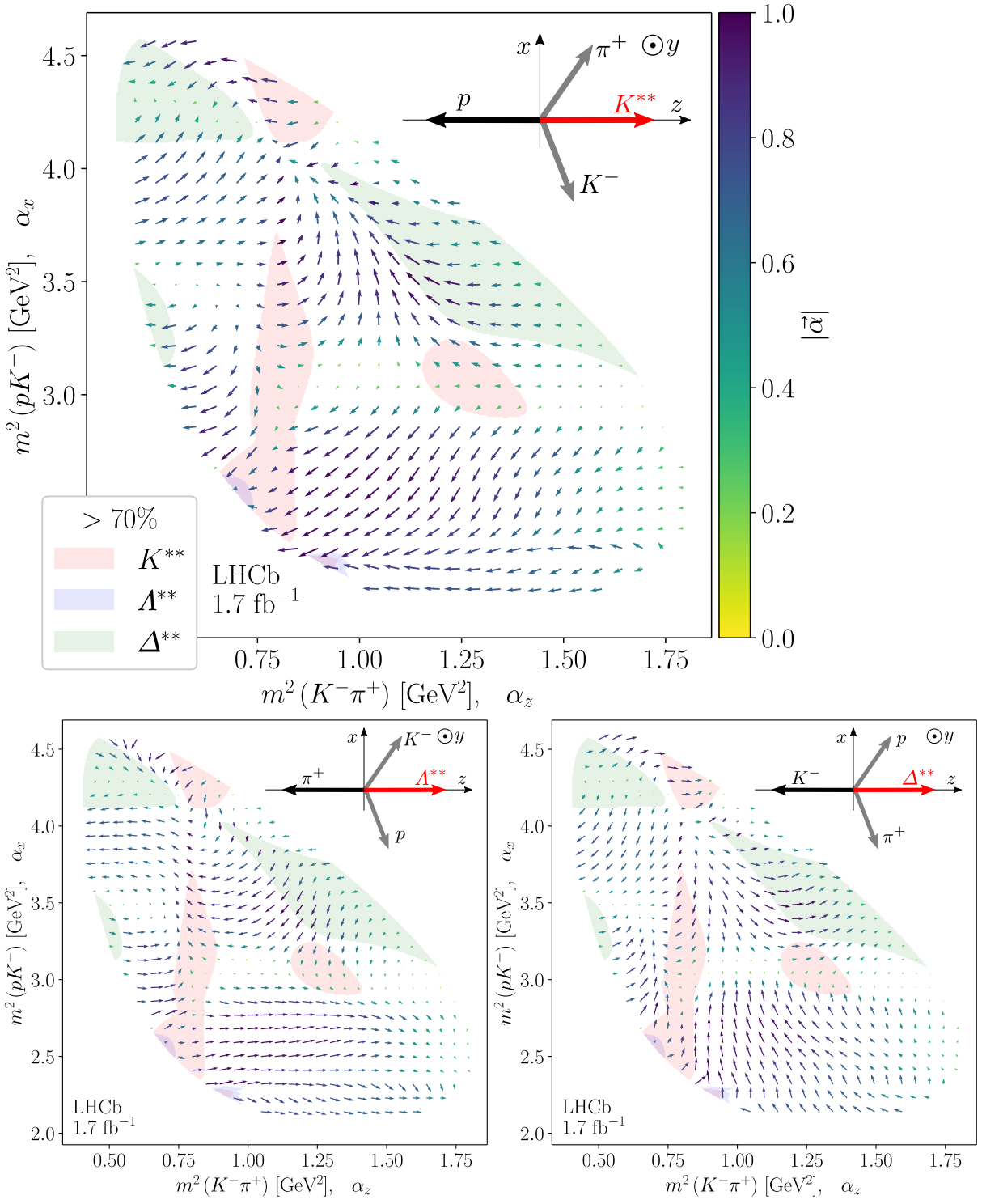


Figure 7.6. Visualisation of the aligned polarimeter vector field $\vec{\alpha}$ over the Dalitz plane for three reference subsystems. The upper right corner of each plot indicates the subsystem to which $\vec{\alpha}$ has been aligned, with $\vec{\alpha}$ aligned to the K^{**} subsystem (top), to the Λ^{**} subsystem (lower left), and to the Δ^{**} subsystem (lower right). The direction of the field vectors indicates the projection of the α_x and α_z component of $\vec{\alpha}$, with the colour indicating its magnitude. The shaded regions indicate where the subsystem dominates, for instance, where $I_{\Lambda^{**}}(\tau)/I_0(\tau) > 70\%$.

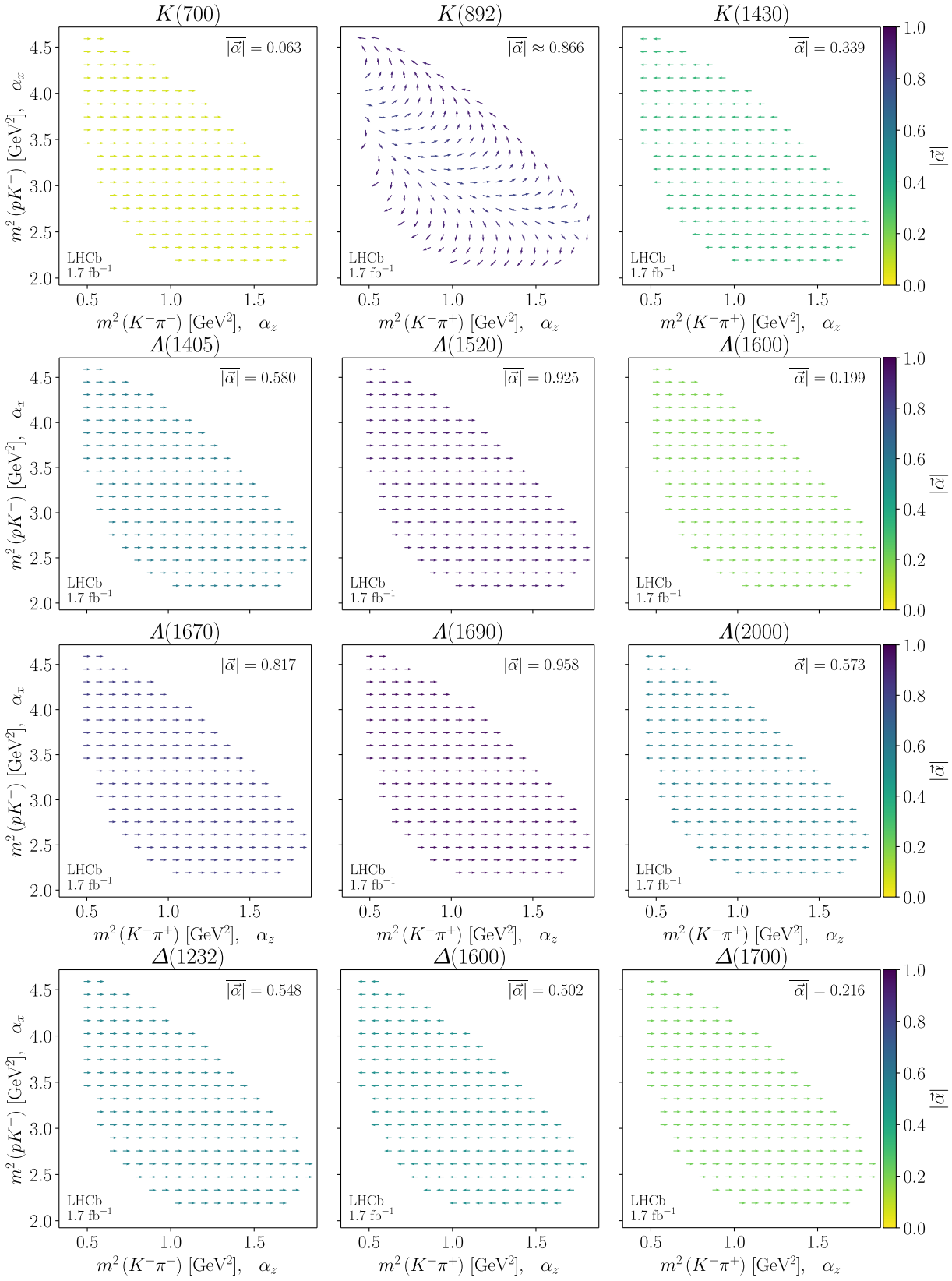


Figure 7.7. Polarimeter vector field $\vec{\alpha}$ for each resonance, aligned to the corresponding subsystem. The modulus of the averaged polarimeter vector is indicated on the top right.

Uncertainty propagation

Since the computed polarimeter vector field is intended to be used in other analysis, we also propagated uncertainties over the Dalitz plane. We investigated three types of uncertainties: statistical uncertainties that are related to the size of the data sample of the original study, systematic uncertainties that are related to detector effects, and model uncertainties that come from the alternative model hypotheses in the performed amplitude analysis.

We grouped the systematic and statistical uncertainties together by evaluating the default model of over a Gaussian-distributed sample of **100 parameter sets** generated from the published fit parameters with their uncertainties taken as the standard deviation. The vector field is computed over a 100×100 **grid** over the Dalitz plane that encapsulates the kinematically allowed region. In each point on the grid, the standard deviation over those 100 model evaluations is then taken as the combined statistical and systematic uncertainty of the polarimeter field. Similarly, to propagate the model uncertainties, we evaluated the polarimeter vector field all the **18 amplitude models** from [264] but take the extrema of the deviations rather than their standard deviation.

It is difficult to visualise the uncertainties of the polarimeter vector field over the Dalitz plane. We found that the best representation is given by the norm $\left|\vec{\alpha}^{(i)} - \vec{\alpha}^{\text{default}}\right|$ of the discrepancy between the default model $\vec{\alpha}^{\text{default}}$ and the alternative models $\vec{\alpha}^{(i)}$, as well as the angle $\theta^{(i)}$ and solid angle $\delta\Omega^{(i)}$ between the two vectors,

$$\begin{aligned} \delta\Omega^{(i)} &= \int_0^{2\pi}\int_0^{\pi} \mathrm{d}\phi \mathrm{d}\cos\theta = 2\pi\left(1 - \cos\theta^{(i)}\right) , \\ \cos\theta^{(i)} &= \frac{\vec{\alpha}^{(i)} \cdot \vec{\alpha}^{\text{default}}}{\left|\vec{\alpha}^{(i)}\right| \left|\vec{\alpha}^{\text{default}}\right|} . \end{aligned}$$

Figure 7.8 illustrates how the discrepancy norm and $\delta\Omega$ are defined. In the left panels of Figure 7.9, the observable of interest (e.g. the solid angle $\delta\Omega$) is evaluated across the Gaussian-distributed sample of 100 parameter sets, and the pointwise standard deviation on the Dalitz plane is taken as the combined uncertainty. The right panels apply the same procedure, but evaluate the observable across all 18 amplitude models. In this case, the maximum deviation at each point is taken as the model uncertainty. The plots demonstrate that the direction of the vector fields is stable under all sources of uncertainty (small deviations in $\delta\Omega$), whereas the magnitude can fluctuate substantially in regions where the intensity is low (see Figure 7.3).

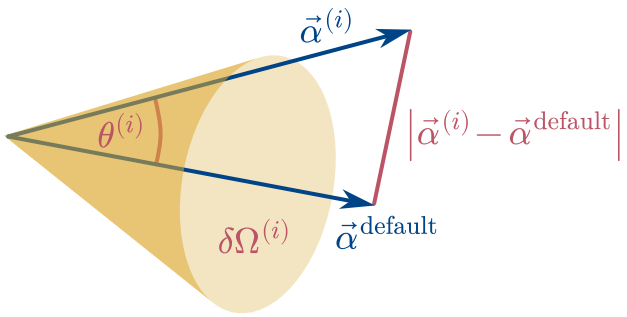


Figure 7.8. Definition of the solid angle $\delta\Omega$, angle $\theta^{(i)}$, and norm of the discrepancy between the polarimeter vector of the default model $\vec{\alpha}^{\text{default}}$ and the alternative models $\vec{\alpha}^{(i)}$.

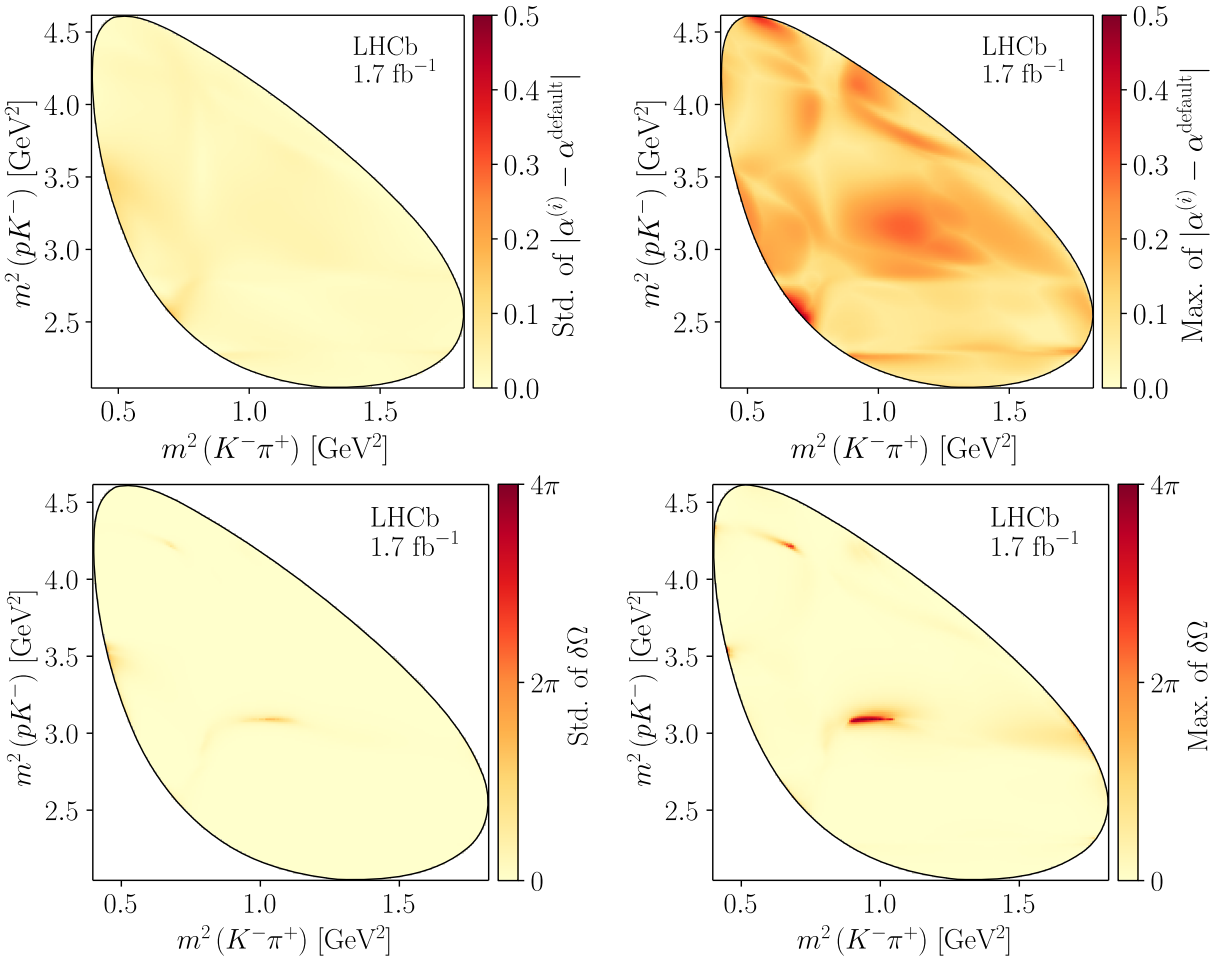


Figure 7.9. Systematic and statistical uncertainties (left) and model uncertainties (right) of the polarimeter vector field $\vec{\alpha}$. Top row: norm of the discrepancy between the default model $\vec{\alpha}^{\text{default}}$ and the alternative models $\vec{\alpha}^{(i)}$. Lower row: the solid angle $\delta\Omega$ between $\vec{\alpha}^{\text{default}}$ and $\vec{\alpha}^{(i)}$.

Averaged polarimeter vector

The uncertainties were also propagated for the averaged polarimeter vector from Equation (7.12). In cartesian coordinates, the polarimeter vector with its uncertainties are

$$\begin{aligned} \overline{\alpha_x} &= \left(-62.6 \pm 4.5^{+8.4}_{-14.8}\right) \times 10^{-3} \\ \overline{\alpha_y} &= \left(+8.9 \pm 8.9^{+9.1}_{-12.7}\right) \times 10^{-3} \\ \overline{\alpha_z} &= \left(-278.0 \pm 23.7^{+12.6}_{-40.4}\right) \times 10^{-3} \\ \overline{|\alpha|} &= \left(669.4 \pm 9.3^{+15.3}_{-10.4}\right) \times 10^{-3}\,. \end{aligned} \tag{7.15}$$

The uncertainties in the averaged polarimeter vector also reflect that the direction of the vector field is quite insensitive to the uncertainties, while the magnitude of the vector field can vary significantly in regions where the intensity is low. This can be clearly seen in polar coordinates, the averaged polarimeter vector and its uncertainties are

$$
\begin{aligned}
|\overline{\alpha}| &= \left(+285.1 \pm 24.0^{+37.9}_{-13.8}\right) \times 10^{-3} \\
\theta\left(\overline{\alpha}\right) &= +2.92 \pm 0.01^{+0.05}_{-0.04}\ \text{rad} \\
&= \left(+0.929 \pm 0.002^{+0.017}_{-0.011}\right) \times \pi \\
\phi\left(\overline{\alpha}\right) &= +3.00 \pm 0.14^{+0.21}_{-0.09}\ \text{rad} \\
&= \left(+0.955 \pm 0.045^{+0.067}_{-0.028}\right) \times \pi\,,
\end{aligned}
$$

where we have used the general transformation for a vector $\vec{v}$ to its polar-coordinate angles,

$$
\begin{aligned}
\theta(\vec{v}) &= \arccos\left(v_z \,/\, |\vec{v}|\right) \\
\phi(\vec{v}) &= \pi - \operatorname{atan2}\left(v_y, -v_x\right)\,.
\end{aligned}
$$

Note that $\overline{|\alpha|}$ is the average of the norm of the polarimeter vector field, while $|\overline{\alpha}|$ is the norm of the averaged polarimeter vector field.

Figure 7.10 shows the distribution of the averaged polarimeter vector field $\overline{\alpha}$ over the 100 parameter samples and the 18 amplitude models. The distribution shows that there is a certain correlation, or at least an upper bound, to the statistical and systematic uncertainties. This is related to the fact that the individual contributions from the resonances to the vector field are aligned, as can be seen in Figure 7.7, which results in a slight correlation.

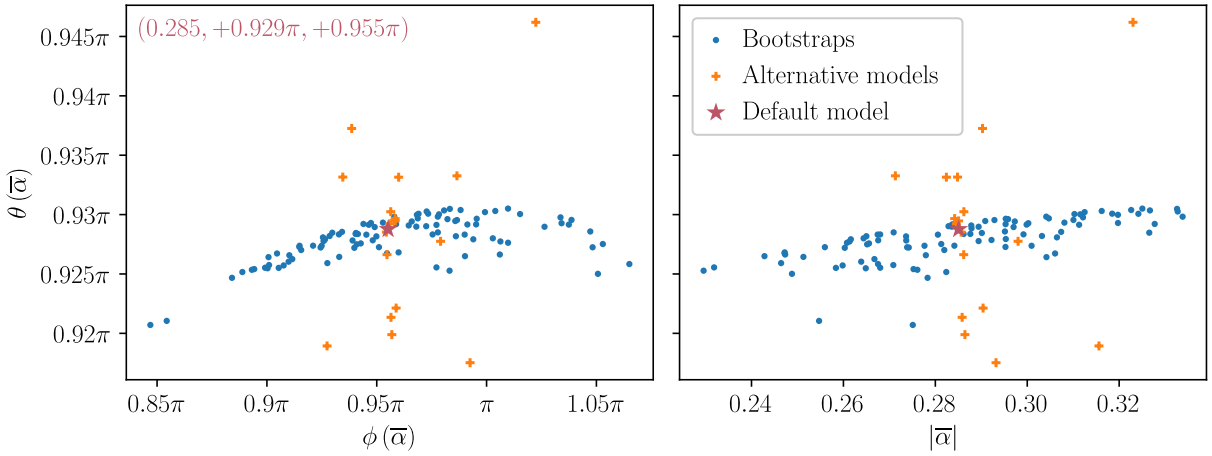


Figure 7.10. Distribution of the computed averaged polarimeter vector $\overline{\alpha}$ in polar coordinates over the bootstrap sample of 100 parameter sets and the 18 amplitude models.

Verification of CAS-assisted model building

The formulation of $\vec{\alpha}$ from the transition amplitudes was powered by the fact that the amplitude models were formulated symbolically for each reference subsystem, which allowed to easily reorder the terms using the CAS. The symbolic amplitude models were directly rendered along with the code using the self-documenting workflow described in Section 4.3. In addition, the symbolic vector fields could directly be expressed as array-oriented code that could efficiently handle the evaluation of the three-dimensional vector fields, showing that the approach can be used for more than intensity evaluation in an amplitude analysis.

We found that JAX was the fastest computational backend, enabling us to propagate both systematic and statistical uncertainties for the entire analysis within an hour. All documentation and results on lc2pkpi-polarimetry.docs.cern.ch were produced through continuous-integration (CI) pipelines, which demonstrates that the self-documenting workflow was both robust and fast enough for high-performance computations, even on low-performance CI machines. The use of an array-oriented computational library further simplified the evaluation of statistical uncertainties, since the models could be applied directly to the sample of 100 parameter sets and computed over the Monte Carlo-generated phase space in a single broadcasted operation (Section 4.1).

Finally, since the amplitude models were formulated using packages from the ComPWA project described in Chapter 5, the cross-check described in Section 7.1 verified that the amplitude models formulated with these packages were consistent with the amplitude models used in [264] and the Julia implementation of the default model [265].

Polarisation determination

For the uncertainty propagation, the polarimeter vector fields were evaluated over a 100×100 grid for each of the 18 models as well as the 100 parameter samples. The results are exported as JSON files, so that they can be reused in other analyses using for instance Equation (7.10) or Equation (7.14). In this section, we use the framework to determine the polarisation of the Λ_c^+ in a toy study using an interpolation over the exported grids.

At first, using Equation (7.10), we generate a toy-data sample for the $\Lambda_c^+ \to pK^-\pi^+$ decay including polarisation of the decaying Λ_c^+, and take that as a measured distribution I. As a test value for $\vec{P}$, we take the polarisation $\vec{P}^0$ that was determined by LHCb [264, p. 11]. These values are, with their reported model uncertainties,

$$\begin{aligned} P_x^0 &= +21.65 \pm 0.36 \\ P_y^0 &= +1.08 \pm 0.09 \\ P_z^0 &= -66.5 \pm 1.1\,, \end{aligned}$$

where the values are given in percentage (%). Next, we reuse the exported 100×100 grids to interpolate the polarimeter vector field $\vec{\alpha}$ as well as the unpolarised intensity I_0 over the generated phase space sample. A gradient-descent algorithm with Minuit2 is then applied to find the most optimal value for $\vec{P}$ such that the computed distribution of I in Equation (7.10) matches the generated distribution best. As a starting guess value, we take $\vec{P} = (+0.3, -0.3, +0.4)$. The fit is repeated for each of the 18 models in order to obtain model uncertainties. The resulting value for the default model, along with the model uncertainties are (in %)

$$\begin{aligned} P_x &= +21.65^{+0.30}_{-0.62} \\ P_y &= +1.08^{+0.02}_{-0.05} \\ P_z &= -66.50^{+1.66}_{-0.85} \end{aligned} \tag{7.16}$$

This is consistent with the model uncertainties reported for $\vec{P}^0$.

The same fit procedure was carried out using the *averaged* polarimeter vector values from Equation (7.15) in Equation (7.11). Again, the uncertainties were determined as extrema over the averaged polarimeter vector field values of each model, resulting in

$$\begin{aligned} P_x &= +20.32^{+1.04}_{-2.44} \\ P_y &= -0.26^{+0.17}_{-0.08} \\ P_z &= -66.14^{+7.91}_{-3.32}\,. \end{aligned} \tag{7.17}$$

This fit is faster, because there is no need to interpolate values for a vector field from the grid samples. However, the uncertainties are a factor 3.5–4.8 larger than when using the polarimeter vector field (Equation (7.16)). This is consistent with the increase in uncertainty when working with scalar polarisation sensitivity [261]. By defining

$$\begin{aligned} S_0^2 &= 3 \int I_0 \, |\vec{\alpha}|^2 \, \mathrm{d}^{(n)}\tau \Big/ \int I_0 \, \mathrm{d}^{(n)}\tau \\ \overline{S}_0^2 &= 3\,(\overline{\alpha}_x^2 + \overline{\alpha}_y^2 + \overline{\alpha}_z^2)\,, \end{aligned}$$

it can be shown that the increase in uncertainty is in the order of at least $S_0/\overline{S}_0 \approx 3$. Here, S_0 quantifies the effective polarisation sensitivity from the full polarimeter vector field averaged over phase space, while $\overline{S}_0$ gives the corresponding sensitivity when this field is replaced by its averaged vector.

Relevance to ComPWA project

The polarimetry study served as a test for the methodologies described in Chapter 4. Formulating the amplitudes symbolically allowed us to rearrange terms with the CAS and derive explicit expressions for $\vec{\alpha}$. From these, we generated JAX code that could evaluate the vector field efficiently, both on dense grids (e.g. 1000×1000 points on the Dalitz plane for visualisations) and across a large number of bootstrap samples. The bootstraps were efficiently handled in parallel

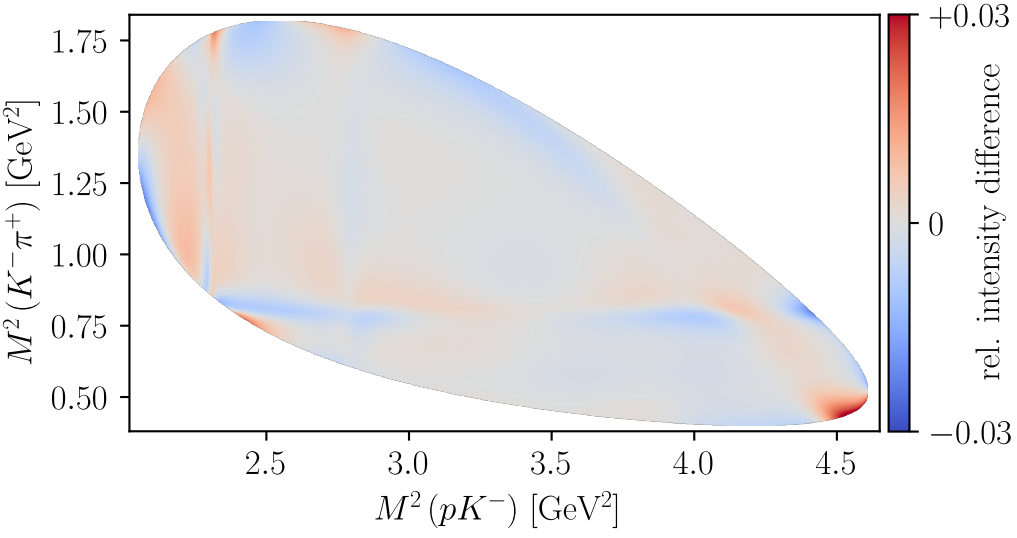


Figure 7.11. Relative intensity difference between the default model for $\Lambda_c^+ \to pK^-\pi^+$ in the helicity basis and the same model in LS basis. The intensities of the LS model are normalised by divided by the sum of the intensities of the default model. For each point of the Dalitz plane, the difference between these normalised intensities is computed and divided over the maximum intensity value of the default model.

by broadcasting parameter arrays with 100 elements against a Monte Carlo-generated phase space sample with 100,000 events. Finally, both numerical results as well as the mathematical formulations were directly published on lc2pkpi-polarimetry.docs.cern.ch using the self-documenting workflow described in Section 4.3 as the analysis progressed. All the results are fully reproducible and are further developed on github.com/ComPWA/polarimetry.

Importantly, the aligned amplitudes and intensities were formulated with the AmpForm-DPD package, so this analysis also served as a validation of the implemented physics. A cross-check against the results from LHCb [264] as well as a comparison with the Julia implementation confirmed that the numerical code is consistent to floating-point precision across all amplitude models of the original LHCb analysis. Moreover, while the default amplitude model was formulated in the helicity basis, one of the alternative models employed the canonical LS basis. As shown in Figure 7.11, the two bases differ by at most 3% relative to the maximum intensity over the Dalitz plane. The polarimetry analysis therefore proves that the symbolic implementation of the DPD method works correctly in both bases and can be commonly used in future analyses.

7.2. Spin alignment tests with $J/\psi \to \bar{p}K_S^0\Sigma^+$

The BESIII collaboration has carried out an investigative amplitude analysis of the decay $J/\psi \to \bar{p}K_S^0\Sigma^+$ using the standard helicity formalism [149–151]. This analysis, documented in the PhD thesis of Wollenberg [73], employed the original AmpForm implementation [229], which did not account for interference between different subsystems. In the present work, we reimplemented the same amplitude model with spin alignment using the DPD method and performed new fits to assess its impact on both fit quality and decay rates.

Data and event selection

The same data set as in the investigative amplitude analysis was used for this study. In total, $(10,087 \pm 44) \times 10^6$ $e^+e^- \to J/\psi$ events (about 10 billion) were collected with the BESIII detector between 2009 and 2019 [266]. From this sample, the decay channel $J/\psi \to \bar{p}K_S^0\Sigma^+$ and its charge-conjugate counterpart, $J/\psi \to pK_S^0\overline{\Sigma}^-$, were selected and reconstructed [267; 268; 73]. Here, we summarise only the main steps of the event selection.

In this decay, the Σ^+ baryon and K_S^0 meson are unstable and have to be reconstructed from their decay products. In the analysis, these particles were reconstructed using their dominant decay channel, as shown in Figure 7.12. The K_S^0 was reconstructed from its decay into two charged pions, while the Σ^+ was reconstructed from its decay into a proton p and neutral pion π^0. The neutral pion cannot be detected directly and was reconstructed from its decay into two photons, $\gamma\gamma$.

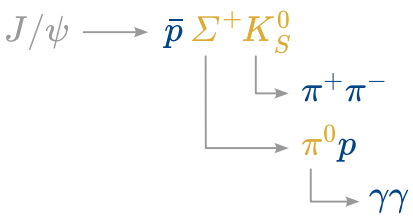


Figure 7.12. Reconstruction of the decay $J/\psi \to \bar{p}\Sigma^+K_S^0$. Detected particles are shown in blue, while the unstable particles that have to be reconstructed are shown in yellow.

All in all, the final state of detected particles is $p\bar{p}\pi^+\pi^-\gamma\gamma$. Apart from the photons, these are all distinct, easily identifiable particles, which makes event selection relatively easy. The first selection criteria are therefore four charged tracks with net charge 0 and at least two photons. Other typical requirements that were applied, are motivated by the BESIII detector geometry. A $|\cos\theta| < 0.93$ cut is applied on the polar angle θ, to select tracks inside the acceptance range of the MDC.

The reconstructed vertex of the prompt proton (or antiproton) is required to be close to the interaction point, within a cylindrical volume around the interaction point with radius $|V_{xy}| < 1$ cm and $|V_z| < 10$ cm along the direction of the beam. The vertex of the secondary proton (antiproton) that originates from the Σ^+ ($\overline{\Sigma}^-$) may not be 20 cm away from the interaction point due to the short lifetime of the Σ baryon, but no cuts are applied in the xy plane. Protons were distinguished from kaons and pions using information from the TOF sub-detector and the energy loss profile from the MDC. Tracks that have a higher probability of a kaon or pion hypothesis are rejected ($\mathcal{P}(p) > \mathcal{P}(K)$ and $\mathcal{P}(p) > \mathcal{P}(\pi)$).

Photons candidates from the π^0 decay are required to have an energy of at least 50 MeV if they are detected in the endcaps of the EMC, at $0.86 < |\cos\theta| < 0.92$, and at least 25 MeV if they are detected in the barrel part of the EMC, at $|\cos\theta| < 0.8$. Photons are only selected if they are recorded 700 ns after the collision to suppress electronic noise. Furthermore, photons that are at an angle of less than 20° with respect to any charged track are rejected to suppress split-offs from those charged tracks.

Since only four charged particles are selected with a net-zero charge, and the proton and antiproton are already identified, no particle identification is required for the remaining two charged tracks. The remaining pions cannot be kaons, because that hardly leaves any phase space and violates strangeness conservation. Both pions should originate from short-lived K^0_S mesons, so the pions are required to be close to the interaction point, at $|V_z| < 20$ cm. A secondary vertex fit is applied to the charged pion pair, which yields a distance $L(K^0_S)$ between the primary and secondary vertex (decay length), with an uncertainty σ_L. Only pion pairs with $\sigma_L < \frac{1}{2}L(K^0_S)$ are selected.

The four-momenta of the short-lived and neutral particles are reconstructed by summing the four-momenta of their decay products, after which additional selection cuts are applied to enhance the signal-to-background ratio. For amplitude analyses, an important step is to fit the measured four-momenta to a kinematic hypothesis. In this decay channel, a **7C kinematic fit** is performed, imposing seven constraints: the total four-momentum of all selected particles must equal that of the initial $e^+e^- \to J/\psi$ system in the lab frame, the invariant mass of the two photons must match the π^0 mass, the invariant mass of the (anti)proton with the reconstructed π^0 must match the Σ^+ mass, and the invariant mass of the charged pion pair must match the K^0_S mass. If multiple combinations are possible, the candidate with the lowest reduced chi-square value, χ^2/n.d.f., is chosen. Only events with a χ^2/n.d.f. value of less than 100 are retained.

In parallel, three Monte Carlo (MC) samples were generated: a **signal** sample containing only events with the decay topology shown in Figure 7.12, a **background** sample including all known J/ψ decays, and a **phase-space** sample in which the three-body final state is distributed uniformly over the available phase space. A total of 4×10^6 events were generated for the signal sample, and 8.9×10^9 events for the background sample, which roughly matches the size of the full data set. The initial e^+e^- collision was simulated with the KKMC event generator to account for initial-state radiation, while the subsequent decays were modelled with `EvtGen` [269] using the world-average branching fractions from the Review of Particle Physics [66].

The signal sample was used to determine the reconstruction efficiency of the decay in the branching fraction measurements, in the case of the background sample. The phase-space sample served to correct the data sample for detector efficiency by reweighting the fit model from the investigative amplitude analysis. The background sample provided estimates of the background distributions. To suppress events with Λ or η in the final states, two additional cuts were applied. Stricter cuts did not significantly reduce the remaining background contributions but did lower the overall event yield. By comparing the number of selected events in the data set with those in the background sample, it was estimated that about 94% of the selected events are genuine signal events and about 6% are irreducible background contributions [73, §6.1].

After applying all selection criteria, a total of 122 664 measured BESIII events remained in the data sample, while the detector-corrected phase-space sample consisted of 609 796 events. Each event consists of three four-momenta that sum to the four-momentum of the J/ψ system in the lab frame, as ensured by the final 7C kinematic fit. The phase-space sample is about five times larger than the data sample, which provides sufficient statistics for normalisation, since no narrow structures are present in the distributions of its kinematic degrees of freedom.

Original amplitude model

As noted in the beginning of this section, no spin alignment was considered in this investigative amplitude analysis, as it was used for the determination of the reconstruction efficiency correction. The intensity function is therefore taken to be a simple incoherent sum over the amplitudes for each helicity in the initial and final state (barring helicity 0 for the J/ψ, as it is produced from e^+e^- collisions, through a virtual photon). Notating the subsystem $N^* \to K^0_S\Sigma^+$ as (2) and $\Sigma^* \to \bar{p}K^0_S$ as (3) as in Figure 7.13, the fitted intensity function is

$$I\left(\sigma_{(2)},\sigma_{(3)};\Omega_{(2)},\Omega^{(2)}_{\Sigma^+},\dots\right) = \sum_{\lambda_{J/\psi}\in\{-1,+1\}}\sum_{\lambda_{\bar{p}}=-1/2}^{+1/2}\sum_{\lambda_{\Sigma^+}=-1/2}^{+1/2}\left|A^{(2)}_{\lambda_{J/\psi}\lambda_{\bar{p}}\lambda_{\Sigma^+}}+A^{(3)}_{\lambda_{J/\psi}\lambda_{\bar{p}}\lambda_{\Sigma^+}}\right|^2, \tag{7.18}$$

with amplitudes of the form (taking subsystem (3) as an example),

$$\begin{aligned}A^{(3)}_{\lambda_{J/\psi}\lambda_{\bar{p}}\lambda_{\Sigma^+}}\left(\sigma_{(3)},\Omega_{(3)},\Omega^{(3)}_{K^0_S}\right) = &\sum_{\rho\in\{\Sigma^*\}}\sum_{\lambda_r=-J_r}^{+J_r} D^{J_{J/\psi}}_{\lambda_{J/\psi},\lambda_r-\lambda_{\Sigma^+}}\left(\Omega_{(3)}\right)\, D^{J_r}_{\lambda_r,\lambda_{K^0_S}-\lambda_{\bar{p}}}\left(\Omega^{(3)}_{K^0_S}\right)\\ &\times\sum_{L,S} C^{J_{J/\psi},\lambda_r-\lambda_{\Sigma^+}}_{L,0,S,\lambda_r-\lambda_{\Sigma^+}}\; C^{S,\lambda_r-\lambda_{\Sigma^+}}_{J_r,\lambda_r,J_{\Sigma^+},-\lambda_{\Sigma^+}}\\ &\times\sum_{\ell,s} C^{J_{\Sigma^*},\lambda_{K^0_S}-\lambda_{\bar{p}}}_{\ell,0,s,\lambda_{K^0_S}-\lambda_{\bar{p}}}\; C^{s,\lambda_{K^0_S}-\lambda_{\bar{p}}}_{J_{K^0_S},\lambda_{K^0_S},J_{\bar{p}},-\lambda_{\bar{p}}}\\ &\times\; c^r_{L,S;\ell,s}\; X^r_{L,S;\ell,s}\left(\sigma_{(3)},m_r,\Gamma_r\right).\end{aligned} \tag{7.19}$$

Here, $\sigma_{(i)}$ denotes the squared invariant mass of subsystem (i), $\Omega^{(i)}$ denotes the helicity angles for that subsystem in the rest frame of J/ψ, $\Omega^{(i)}$ are the helicity angle of final-state particle k in the rest frame of (i), and λ_k is the helicity of particle k. The helicity of the spinless K^S_0 meson is always 0 and is not listed in the indices of the amplitudes. This equation is an implementation of Equation (3.18), using Equation (3.17) to transform it to the canonical basis. Scalar factors

have been absorbed into a single complex-valued scalar factor $c^r_{L,S;\ell,s}$ for each partial wave with angular momentum and coupled spin L, S for the production vertex of the intermediate state and ℓ, s for the decay vertex of the intermediate state (see Figure 7.13). The function $X^r_{L,S;\ell,s}$ parametrises the dynamics of the entire decay chain for resonance r with mass m_r and width Γ_r. Optimisable parameters are indicated in orange, while kinematic variables computed for each event in the data are indicated in blue.

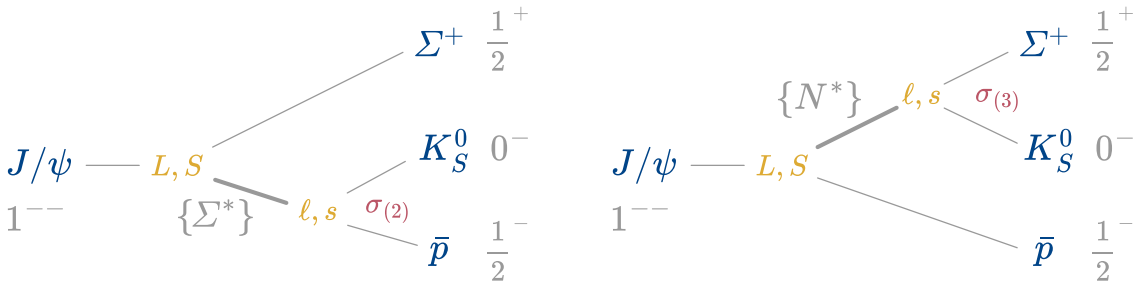


Figure 7.13. The two subsystems that the amplitude model in Equation (7.18) defines. The $J^{P(C)}$ values of the initial state and final state are shown in gray.

Several versions of the model were optimised to the data by minimising an unbinned negative log-likelihood function using gradient descent with Minuit2 and JAX as the computational backend. The models differed in the number of included resonances but shared a common parametrisation for the dynamics function X, formulated in the canonical basis with angular momentum truncated at 4. Each resonance was described by a Breit–Wigner function $\mathcal{R}_\ell(s)$ with an energy-dependent width $\Gamma_\ell(\sigma)$ over Mandelstam variable σ, multiplied by two form factors: $\mathcal{F}_L$ for the production vertex and $\mathcal{F}_\ell$ for the decay vertex. For example, a decay chain $A \xrightarrow{L} (r \xrightarrow{\ell} i\,j)\,k$ would be parametrised as

$$X^r_{L,S;\ell,s}(\sigma_k, m_r, \Gamma_r) \;=\; \mathcal{F}_L(\mu^2_{J/\psi}, \sigma_k, \mu^2_k)\; \mathcal{R}_\ell\left(\sigma_k, m_r, \Gamma_r\right)\; \mathcal{F}_\ell(\sigma_k, \mu^2_i, \mu^2_j)\,,$$

with μ_i the rest mass of particle i. Each of the functions is defined as

$$\begin{aligned}
\mathcal{R}_\ell\left(\sigma, m_r, \Gamma_r\right) &= \frac{\Gamma_r m_r}{m_r^2 - i m_r \Gamma_\ell\left(\sigma\right) - \sigma} \\
\Gamma_\ell\left(\sigma\right) &= \frac{\Gamma_r \mathcal{F}_\ell\left(\sigma, \mu_i^2, \mu_j^2\right)^2 \rho\left(\sigma\right)}{\mathcal{F}_\ell\left(m_r^2, \mu_i^2, \mu_j^2\right)^2 \rho\left(m_r^2\right)} \\
\rho\left(\sigma\right) &= \frac{2\sqrt{q^2\left(\sigma\right)}}{\sqrt{\sigma}} \\
q^2\left(\sigma\right) &= \frac{\left(\sigma - \left(\mu_i - \mu_j\right)^2\right)\left(\sigma - \left(\mu_i + \mu_j\right)^2\right)}{4\sigma}\,.
\end{aligned}$$

Some models contained "non-resonant" contributions that were parametrised without a Breit–Wigner, but with two form factors only.

The final resonance set was determined by iteratively including resonances and selecting the model with the lowest Akaike information criterion (AIC) to penalise overfitting. The optimal

model (referred to as the **non-DPD model** in the following) includes five Σ^* resonances, one N^* resonance, and a non-resonant contribution in the $\bar{p}K^0_S$ subsystem. In the reported fit result, the resonance masses were fixed to the mean values listed in the 2022 Review of Particle Physics of 2022 [66], while the widths were left free. Production and decay couplings were combined into a single complex coefficient per amplitude, also treated as a free parameter, with one coefficient fixed to 1 to remove the overall phase and scaling ambiguity.

The measured data sample is shown in Figure 7.14 (left) and the optimised non-DPD model is shown in Figure 7.15. The right pane in Figure 7.14 shows the MC-generated, acceptance-corrected phase space that is used for plotting the intensity model, which shows that part of the kinematically allowed region is not covered by the detector. Even without a proper treatment of spin alignment, the amplitude models describe the data distributions decently. However, certain structures are not yet well accounted for. In addition, the decay rates of this model [73, p. 76] were remarkably high, which might be caused by interference effects (see Figure 7.18, left).

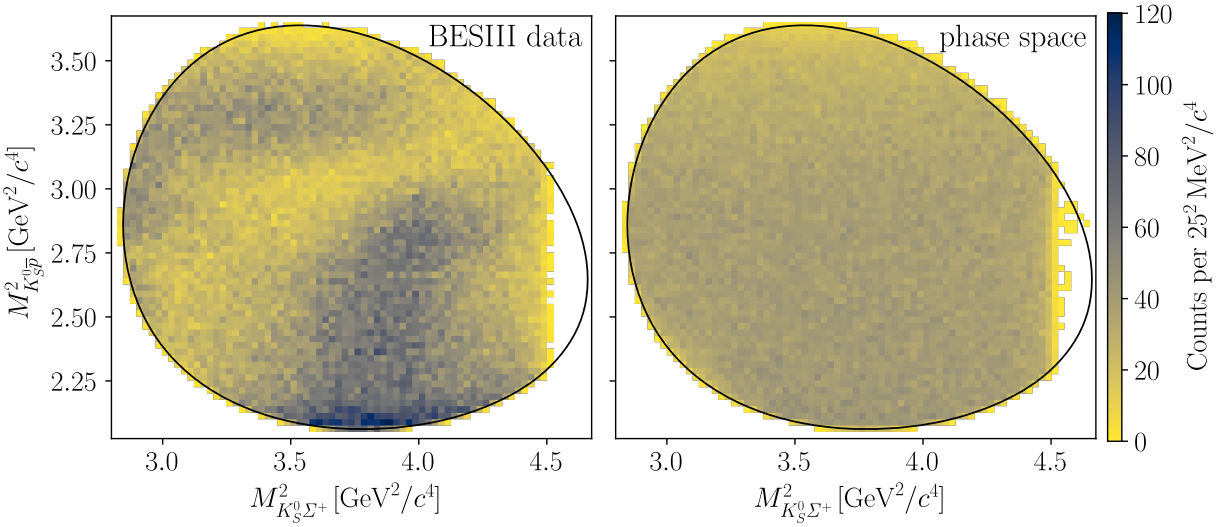


Figure 7.14. Dalitz-plot visualisation of BESIII data (left) and acceptance-corrected phase space (right) for the decay $J/\psi \to \bar{p}K^0_S\Sigma^+$. The bin values of the phase space sample have been normalised with regard to the integral over the data sample. Higher energies in the invariant $K^0_S\Sigma^+$ mass are limited by the detector acceptance.

Fit result with DPD

For this work, we performed another investigative fit with the same resonance content, but reformulated it with the DPD method [157]. The main intensity function is an aligned version of Equation (7.18), which contains a Wigner d-function for each initial and final state that has spin (three in this case: J/ψ, $\bar{p}$, and Σ^+). Using DPD, the unpolarised intensity function of Equation (3.24) for the decay in Figure 7.13 becomes

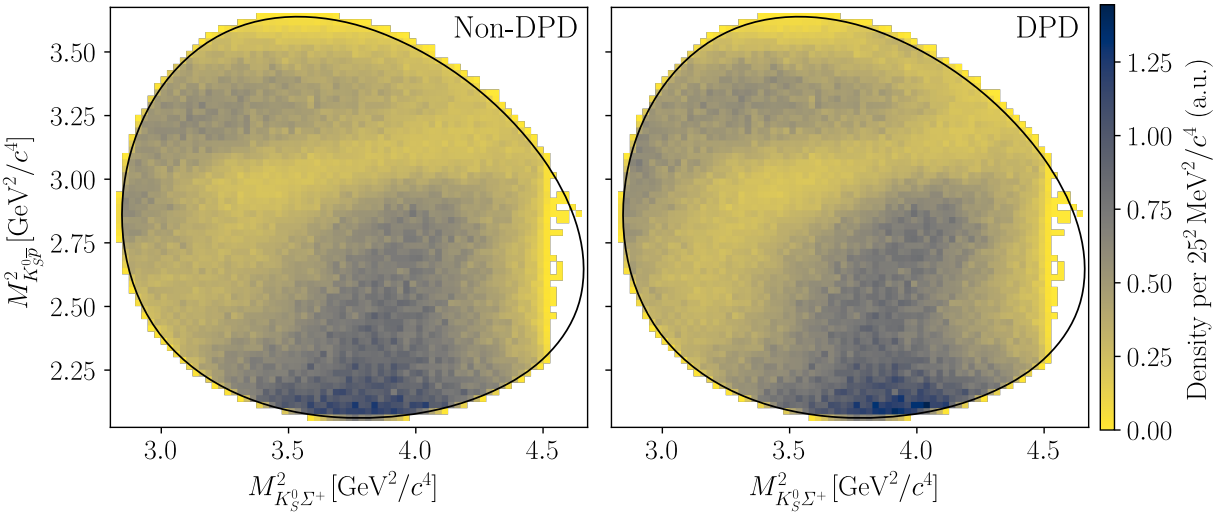


Figure 7.15. Comparison between the non-DPD model that did not use spin alignment (left) and the DPD model (right).

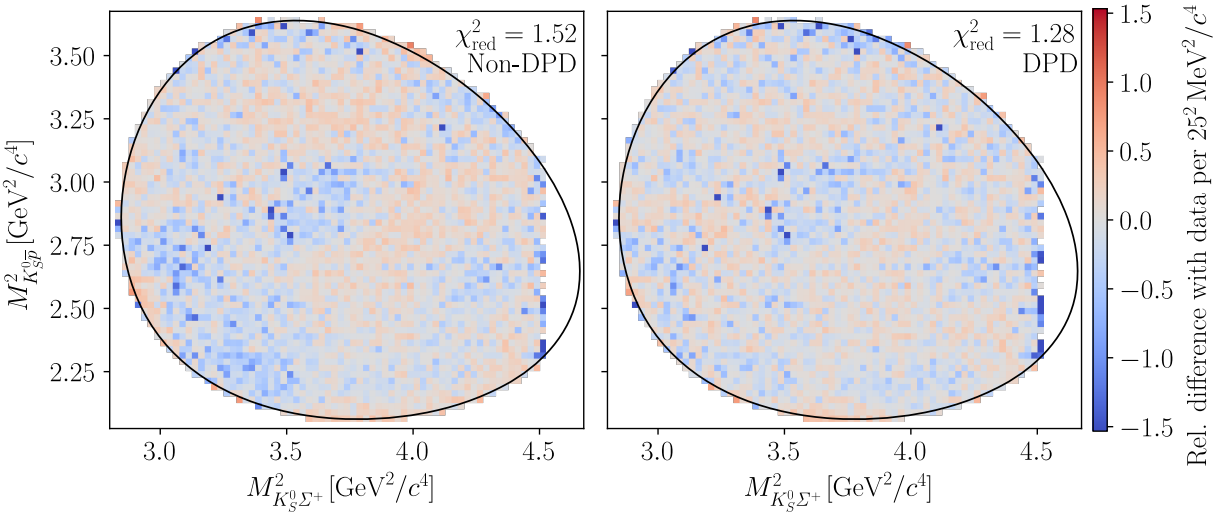


Figure 7.16. Bin-by-bin differences between the BESIII data and the non-DPD model of the previous study (left) and the DPD model (right), relative to each bin uncertainty.

$$
\begin{aligned}
I\left(\sigma_{(2)},\sigma_{(3)};\theta_{(2)},\theta^{(2)}_{\Sigma^+},\dots,\zeta^3_{3(3)}\right) \;=\; & \sum_{\lambda'_{J/\psi}=-1}^{+1}\sum_{\lambda'_{\bar{p}}=-1/2}^{+1/2}\sum_{\lambda'_{\Sigma^+}=-1/2}^{+1/2} \\
\Bigg| & \sum_{\lambda_{J/\psi}=-1}^{+1}\sum_{\lambda_{\bar{p}}=-1/2}^{+1/2}\sum_{\lambda_{\Sigma^+}=-1/2}^{+1/2} \\
& A^{(2)}_{\lambda_{J/\psi}\lambda_{\bar{p}}\lambda_{\Sigma^+}}\; d^{J_{J/\psi}}_{\lambda'_{J/\psi}\lambda_{J/\psi}}\left(\zeta^0_{2(3)}\right)\; d^{J_{\bar{p}}}_{\lambda_{\bar{p}}\lambda'_{\bar{p}}}\left(\zeta^2_{2(3)}\right)\; d^{J_{\Sigma^+}}_{\lambda_{\Sigma^+}\lambda'_{\Sigma^+}}\left(\zeta^3_{2(3)}\right) \\
+\; & A^{(3)}_{\lambda_{J/\psi}\lambda_{\bar{p}}\lambda_{\Sigma^+}}\; d^{J_{J/\psi}}_{\lambda'_{J/\psi}\lambda_{J/\psi}}\left(\zeta^0_{3(3)}\right)\; d^{J_{\bar{p}}}_{\lambda_{\bar{p}}\lambda'_{\bar{p}}}\left(\zeta^2_{3(3)}\right)\; d^{J_{\Sigma^+}}_{\lambda_{\Sigma^+}\lambda'_{\Sigma^+}}\left(\zeta^3_{3(3)}\right)\Bigg|^2 ,
\end{aligned}
\tag{7.20}
$$

where $\zeta^i_{j(k)}$ are additional rotation angles that can be computed from Mandelstam variables using Equation (3.27). The Wigner rotations for subsystem (3) are marked gray, because their corresponding rotation angles $\zeta^i_{k(k)}$ are zero. As opposed to Equation (7.18), the longitudinal polarisation of J/ψ is also summed over, consistently with the derivation in Equation (3.24). As an example of one of the amplitudes, the amplitude for subsystem (3) in Figure 7.13 is

$$
\begin{aligned}
A^{(3)}_{\lambda_{J/\psi}\lambda_{\bar{p}}\lambda_{\Sigma^+}}\left(\sigma_{(3)},\theta^{(3)}_{K^0_S}\right) \;=\; & (-1)^{J_{\Sigma^+}-\lambda_{\Sigma^+}}\,(-1)^{J_{\bar{p}}-\lambda_{\bar{p}}} \\
& \sum_{r\in\{\Sigma^*\}}\sum_{\lambda_r=-J_r}^{+J_r}\delta_{\lambda_{J/\psi},\lambda_r-\lambda_{\Sigma^+}}\; d^{J_r}_{\lambda_r,\lambda_{K^0_S}-\lambda_{\bar{p}}}\left(\theta^{(3)}_{K^0_S}\right) \\
& \times \sum_{L,S} C^{J_{J/\psi},\lambda_r-\lambda_{\Sigma^+}}_{L,0,S,\lambda_r-\lambda_{\Sigma^+}}\; C^{S,\lambda_r-\lambda_{\Sigma^+}}_{J_r,\lambda_r,J_{\Sigma^+},-\lambda_{\Sigma^+}} \\
& \times \sum_{\ell,s} C^{J_{\Sigma^*},\lambda_{K^0_S}-\lambda_{\bar{p}}}_{\ell,0,s,\lambda_{K^0_S}-\lambda_{\bar{p}}}\; C^{s,\lambda_{K^0_S}-\lambda_{\bar{p}}}_{J_{K^0_S},\lambda_{K^0_S},J_{\bar{p}},-\lambda_{\bar{p}}} \\
& \times\; \mathcal{H}^r_{L,S}\;\mathcal{H}^r_{\ell,s}\;X^r_{L,S;\ell,s}\left(\sigma_{(3)};m_r,\Gamma_r\right) .
\end{aligned}
\tag{7.21}
$$

There are small differences with the amplitude in Equation (7.19). There is only one Wigner d-function rather than two Wigner D-functions (the Wigner d-function for the production node simplifies to a Kronecker delta). The single scaling factor $c^r_{L,S;\ell,s}$ is replaced by two couplings $\mathcal{H}^r_{L,S}$ and $\mathcal{H}^r_{\ell,s}$, for the production node and the decay node, respectively.

The same set of resonances and the non-resonant term were retained, but their masses and widths were fixed to the values reported by the Review of Particle Physics [66]. We refer to this as a **coupling-only fit**, which is computationally efficient because the free parameters enter only as linear coefficients. In the non-DPD model, production and decay couplings were combined into a single complex coefficient per partial wave. In the coupling-only fit, however, only the production couplings $\mathcal{H}^r_{L,S}$ were treated as free parameters, with one fixed to 1 to remove phase and magnitude ambiguities. All decay couplings $\mathcal{H}^r_{\ell,s}$ were set to 1. Although this simplification is not strictly physical, since it removes certain degrees of freedom in some partial waves, it reduces the parameter space considerably. To ensure convergence to the global minimum, the fit was repeated 600 times with randomised initial parameter values, and the solution with the best likelihood value was selected.

The fitted non-DPD and DPD models across the Dalitz plane are shown in Figure 7.15, with the invariant mass of the $K^0_S\bar{p}$ (Σ^*) subsystem on the vertical axis and the invariant mass of the $K^0_S\Sigma^+$ (N^*) subsystem on the horizontal axis. The two models exhibit no substantial

differences across these kinematic degrees of freedom. A more detailed comparison is provided in Figure 7.16, which shows the normalised bin-by-bin deviations between each model and the data. These distributions demonstrate that both models reproduce the data well within statistical uncertainties across the Dalitz plane.

More pronounced differences appear in one-dimensional projections onto specific kinematic variables. Figure 7.17 compares the BESIII data with the fitted DPD model and the original non-DPD model. The data distributions are shown as transparent blue histograms, while the acceptance-corrected phase-space distributions are indicated in gray to highlight where observed structures must arise from dynamics rather than kinematics. The green and orange histogram outlines represent the DPD and non-DPD models, respectively, evaluated over the phase-space sample and normalised to the number of data events. For invariant-mass and helicity-angle distributions, the resonance contributions of the DPD model are overlaid as coloured lines in the relevant subsystem. The upper panels display the bin-by-bin residuals (pulls) between the models and the data, normalised to the statistical uncertainty of each bin. The sum of squared pulls, divided by the number of degrees of freedom (bins minus fitted parameters), defines the reduced chi-square χ^2_{red} shown in each plot. Despite having fewer degrees of freedom, the DPD model consistently achieves a lower χ^2_{red} across all projections compared to the non-DPD model. The ϕ distributions are included for completeness, although neither model contains explicit ϕ-dependent terms.

An explanation for the fact that the DPD model performs much better can be found when we investigate the decay-rate matrix of both models. Figure 7.18 displays the decay rates per resonance, obtained by setting all couplings except those of the resonance under consideration to zero. In the non-DPD model, the decay rates show large contributions from individual resonances that interfere destructively to reproduce the data. Despite this cancellation, the total sum of decay rates still far exceeds 100%, leading to large off-diagonal terms in the matrix, $(I_{ij} - I_i - I_j)/I_{\text{tot}}$. By contrast, the DPD model does not rely on strong destructive interference: its decay-rate matrix has a lower triangle and diagonal that add up consistently to 100%.

Figure 7.19 shows the model in the Dalitz plane for each element of the decay-rate matrix in the non-DPD model (Figure 7.18, left). The mass positions of the relevant resonances are indicated to aid the interpretation of the interference patterns. Many of the Σ^* resonances lie above the upper limit of the phase space and contribute only through their low-mass tails. Interestingly, resonance structures appear inverted in some of the off-diagonal elements, with the resonance peak showing a reduced intensity compared to other regions of the Dalitz plane (most notably for $\Sigma(1750)$ and $\Sigma(1900)$). This behaviour arises from fully destructive interference between the corresponding partial waves, which may explain the unusually large fit fractions obtained in the non-DPD model.

The DPD model in Figure 7.20, on the other hand, produces interference patterns where resonances from different subsystems meet, with some intersections showing constructive and others destructive interference. Moreover, the off-diagonal elements no longer display the inverted peaks observed in the non-DPD distributions of Figure 7.19.

For reference, Figure 7.21 presents the decay rates per partial wave for the selected DPD model. Within a given subsystem, partial waves do not interfere (as indicated by the absence of non-zero off-diagonal elements in each sub-matrix), whereas interference is present between subsystems. Strong interference within the same subsystem is observed only between resonances with identical quantum numbers, such as $\Sigma(1750)$, $\Sigma(1900)$, and the non-resonant term. This suggests that a more sophisticated parametrisation of the dynamics is needed to disentangle their contributions and to properly describe threshold openings.

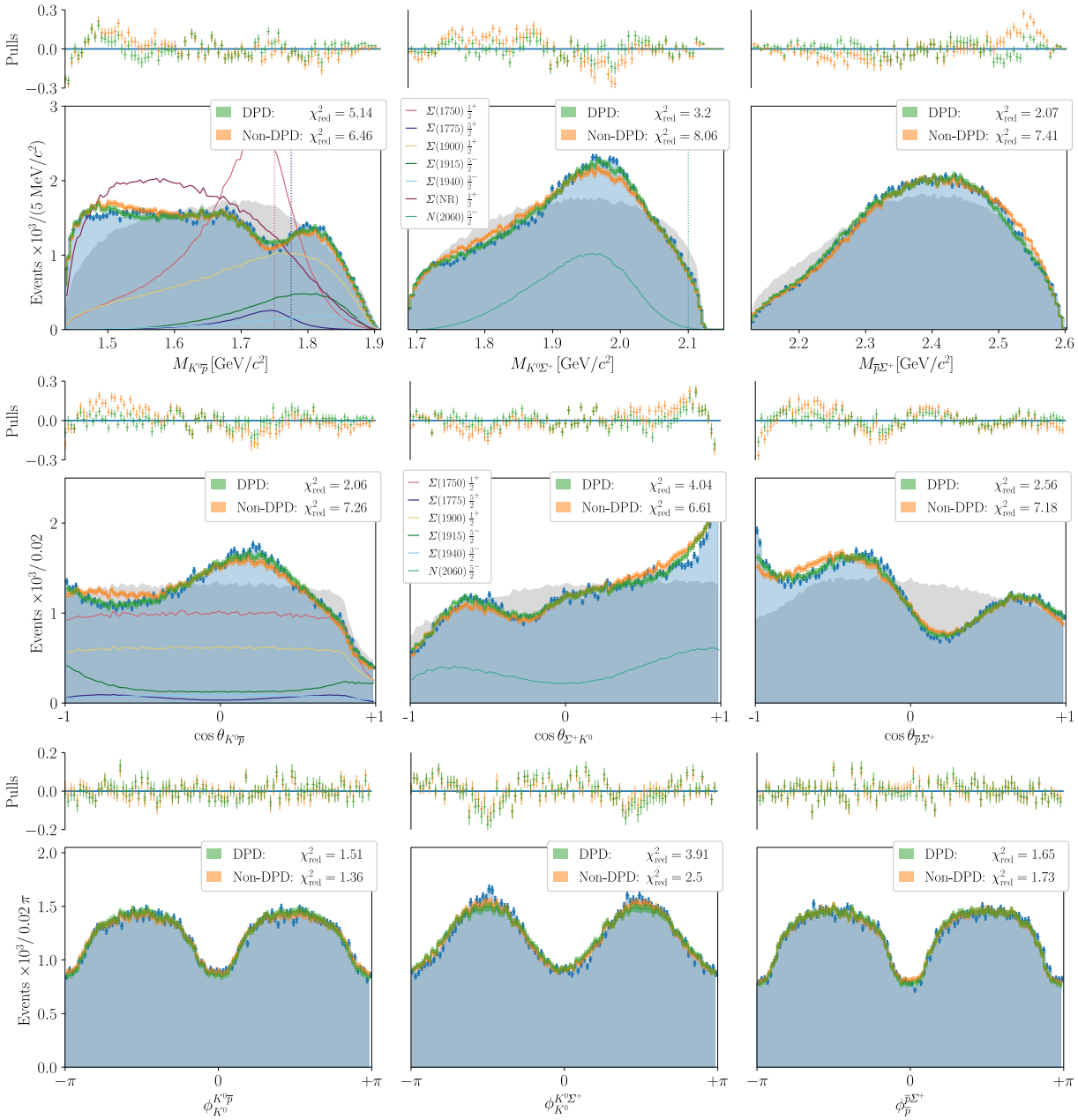


Figure 7.17. Comparison between the BESIII data (blue) for $J/\psi \to \bar{p}K^0_S\Sigma^+$, the fit of the DPD model (green), and the original non-DPD model (orange) across nine kinematic variables. The acceptance-corrected phase-space distribution is shown in gray. Each panel quotes the reduced chi-square χ^2_{red} for both models, computed from the sum of squared differences between data and model in each bin, normalised to the statistical uncertainty of the bin. The corresponding residuals ("pulls") are shown above each histogram. Thin coloured lines indicate the resonance contributions of the DPD model in the relevant subsystems (Σ^* in $K^0_S\bar{p}$ and N^* in $K^0_S\Sigma^+$).

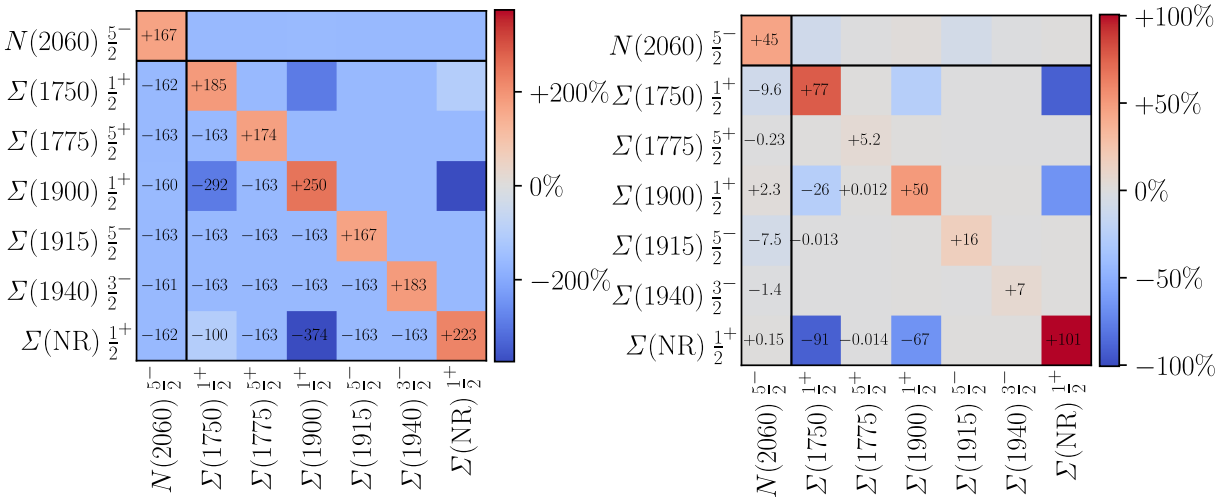


Figure 7.18. Decay-rate matrix for non-DPD model (left) and the correct DPD model (right).

Further analyses are required to determine whether the selected resonances in the DPD model have physical significance. As mentioned, the focus in [73] was the determination of the branching fraction of $J/\psi \to \bar{p}K^0_S\Sigma^+$, along with a test whether the infrastructure can converge amplitude models efficiently to the data sample. The reimplementation with DPD is a first step towards a more complex amplitude model that includes spin alignment, but a systematic analysis of the resonance and individual partial-wave contributions, interference effects, and correlations between the parameters for each of the different models needs to be performed before any conclusions can be drawn about the N^* and Σ^* spectrum.

As a final thought, although the amplitude in Equation (7.21) carries three helicity indices, it can be used to construct three polarimeter vector fields (one for each $\lambda_{J/\psi}$ in the rest frame) using Equation (7.9). These polarimeter vector fields encode the spin correlations of the final-state baryons and thus carry information about the underlying dynamics. This makes it possible to relate the study of these three-body decays to the spin-transfer observables and moment expansions commonly extracted in scattering experiments (Section 1.3), in addition to matching dynamical modelling through pole positions (Section 2.4). Conceptually, the idea is to reinterpret the recoiling $\bar{p}$ as being crossed to the initial state and the J/ψ as an effective virtual photon via vector meson dominance, yielding a photo-production process $p\gamma^* \to K^0_S\Sigma^+$ (see Figure 7.22). Furthermore, the decay $\Sigma^+ \to p\pi^0$ can be used to increase sensitivity to the polarisation of Σ^+. These ideas can be developed further once the modelling of the dynamics becomes more advanced.

7.3. Performance benchmarks

The analyses described in this section were performed on standard desktop PCs and laptops. Running on this environment, we found that JAX is fast enough as a computational backend for the evaluation of the uncertainties over the polarimeter vector field of Λ_c (Section 7.1) and the fit of the amplitude model in the J/ψ study (Section 7.2). All notebooks of the polarimetry study, including appendices, can be run in less than one hour on a 8-Core AMD Ryzen 7 5800X

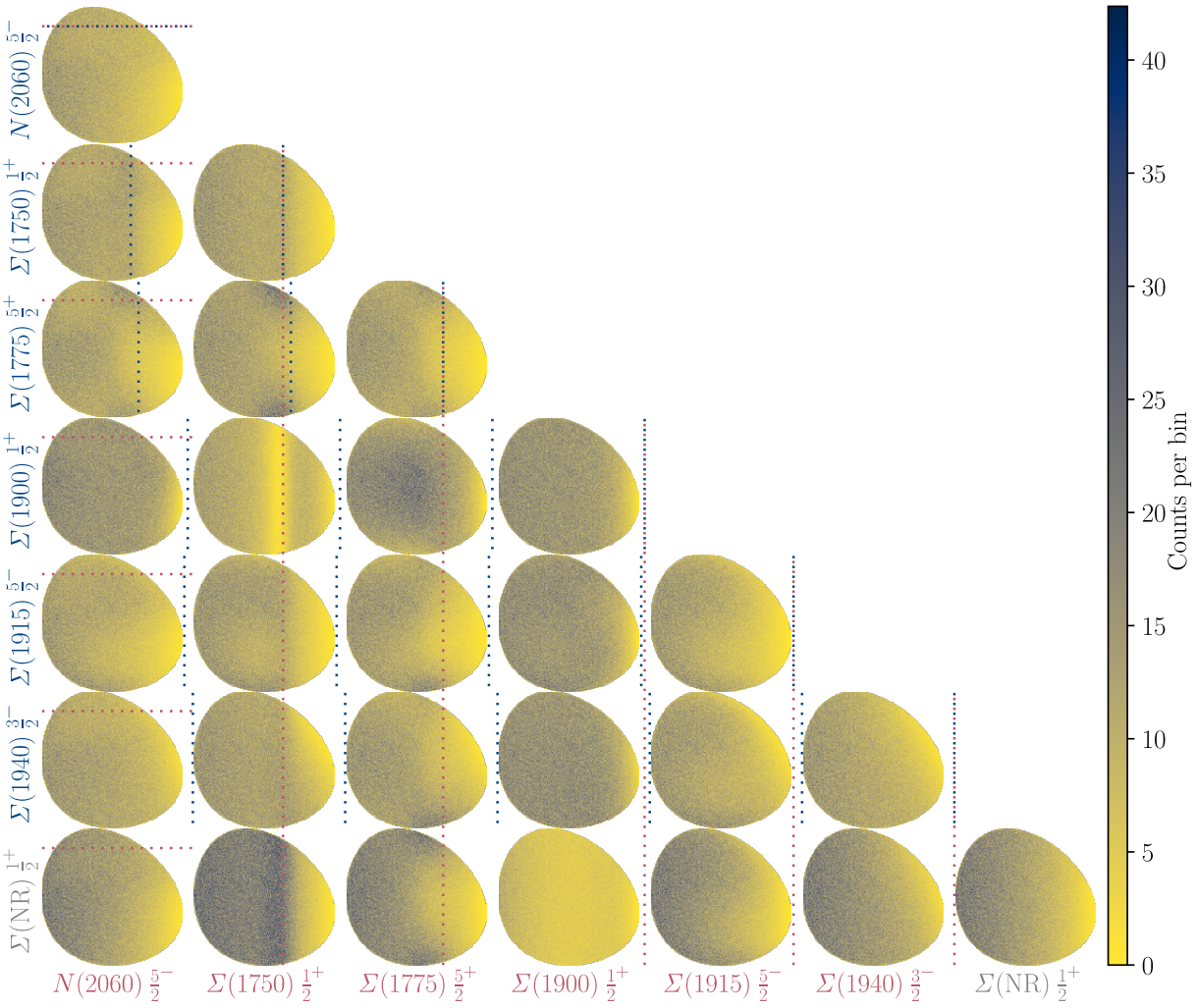


Figure 7.19. Sub-intensity distributions across the Dalitz plane for each element of the decay-rate matrix in Figure 7.18 for the non-DPD model. Axis labels are given in Figure 7.14. Resonance mass positions are marked by dashed lines: blue for resonances associated with the vertical axis and red for those associated with the horizontal axis. The non-resonant term has no defined mass position.

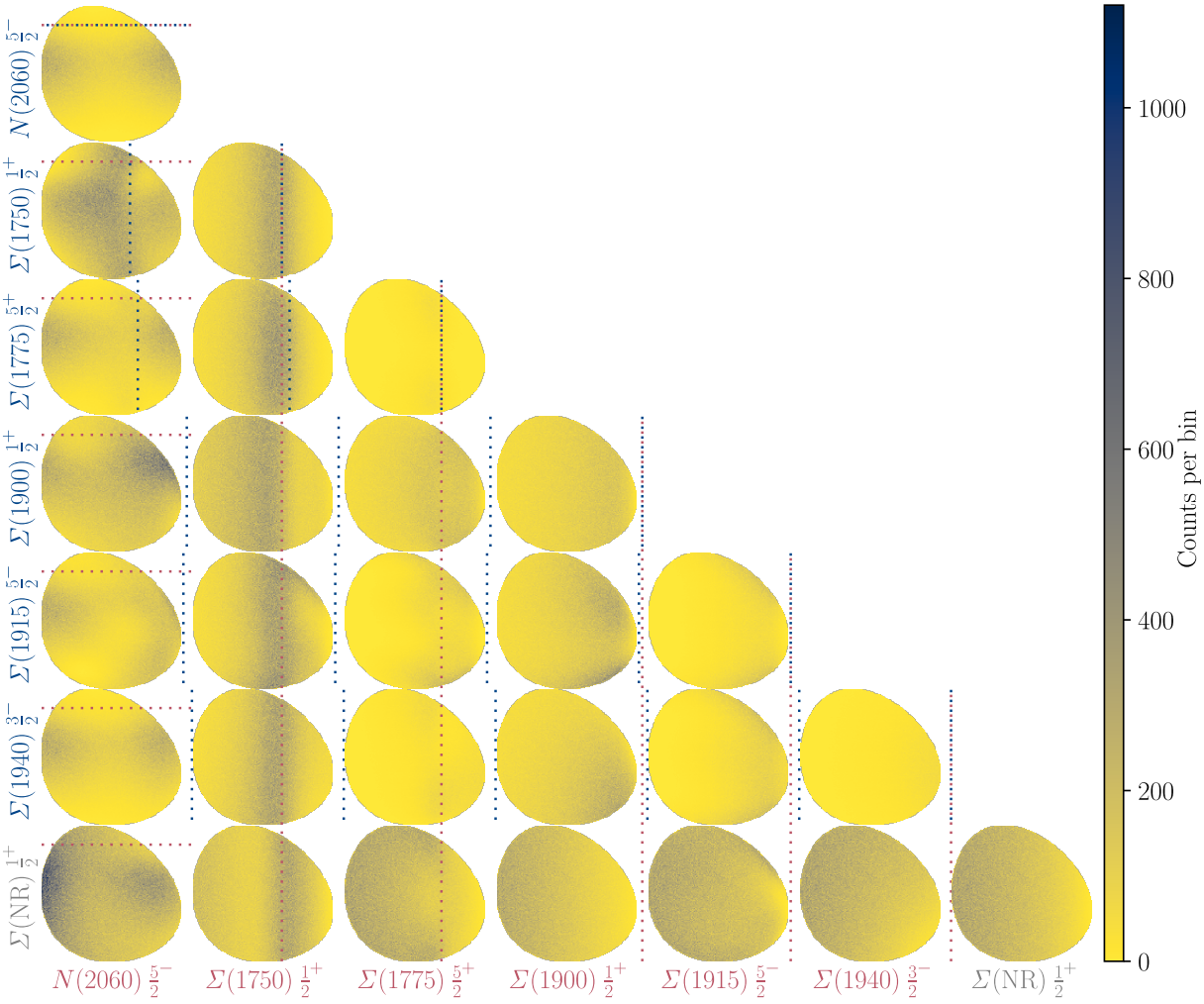


Figure 7.20. Sub-intensity distributions across the Dalitz plane for each element of the decay-rate matrix in Figure 7.18 for the DPD model. Axis labels are given in Figure 7.14. Resonance mass positions are marked by dashed lines: blue for resonances associated with the vertical axis and red for those associated with the horizontal axis. The non-resonant term has no defined mass position.

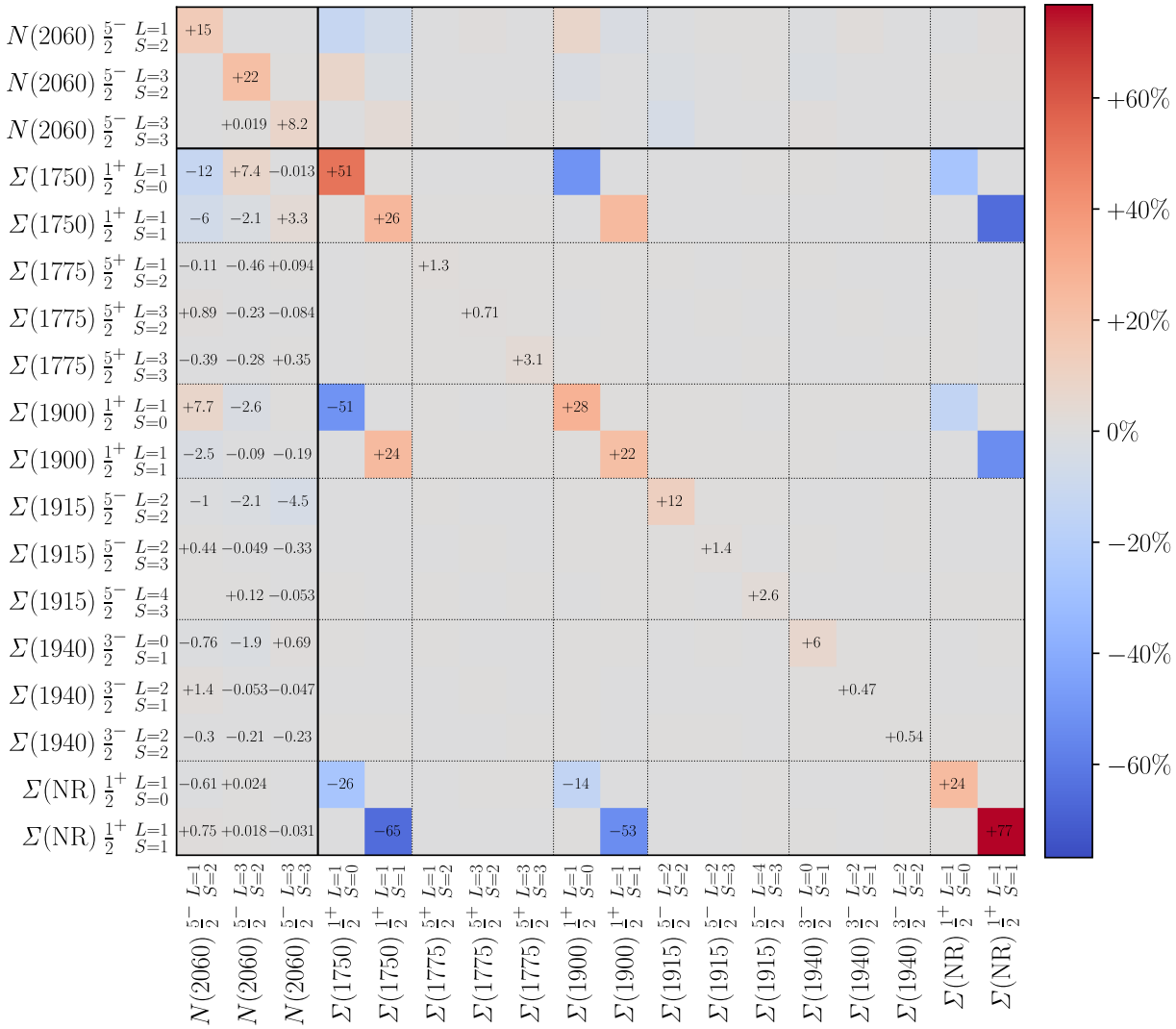


Figure 7.21. Decay-rate matrix for each partial wave in the DPD model. The decay rate is similar to that of Figure 7.18, but the sub-intensities are now computed for each coupling separately.

Figure 7.22. The three-body decay $J/\psi \to \bar{p}K\Sigma$ produced by e^+e^- (left) and scattering process $p\gamma \to K\Sigma$ (right) that are related through crossing and vector meson dominance. The charges of K_S^0 and Σ^+ are omitted to highlight that the same idea can be applied to other final states with a $\bar{p}$, a baryon, and a meson.

Processor. The fits for the BESIII analysis typically take around an hour to converge using Minuit2, while coupling-only fits over the same data take around 5 minutes.

Hardware accelerator comparison

To have an indication of computational performance across different hardware accelerators, we carried out several fits for the decay channel $J/\psi \to \bar{p}K_S^0\Sigma^+$ using the amplitude models of Equation (7.20), which JAX as numerical backend. The fits were performed on a hit-and-miss–generated data sample of the same size as that described in Section 7.2, together with a Monte Carlo–generated phase space sample of equal size to that used in the actual analysis. Both datasets assumed an ideal detector without efficiency losses. The data were generated with the same DPD model and parameter values as reported in Section 7.2. To maximise model complexity, all parameters were left free except for one coupling, which was fixed to avoid phase ambiguities.

- Data size: **180,296** events
- Phase space: **609,796** events
- Model complexity:
  - **82** free parameters: 18 real-valued masses, 32 complex
  - **2** subsystems: N^* and Σ^*
  - **9** resonances: 3 N^*, 6 Σ^*
  - **248** amplitudes: 96 N^*, 152 Σ^*
  - **405,342** operations in the symbolic expression tree (via `sympy.count_ops()`)

Table 7.2 shows the performance on different hardware accelerators. Of importance to the comparison is the number of iterations (function evaluations) per second that Minuit2 performs, because the fit does not converge to the global minimum. Minuit2 finishes the fit after exactly 34,241 iterations on all devices, even if the resulting likelihood differs slightly due to floating-point precision differences. The speed-up is computed with respect to the AMD Ryzen 7 5800X, which was used to perform the analyses reported in this chapter.

Device	Specifications	Duration (s)	Speed-up	Frequency (Hz)
A100-SXM4-40GB	19.5 TFLOPS, 400 W	280 ± 3	20x	122 ± 1
NVIDIA L4	30.3 TFLOPS, 72 W	$1\,352 \pm 4$	4.2x	25.3 ± 0.1
Tesla T4	8.1 TFLOPS, 70 W	$2\,484 \pm 13$	2.2x	13.8 ± 0.1
AMD Ryzen 7 5800X	3.8 GHz, 8-Core	$5\,675 \pm 124$	1x	6.04 ± 0.01
13th Gen Intel i7-1355U	2.6 GHz, 6-Core	$9\,916 \pm 43$	0.6x	3.45 ± 0.01

Table 7.2. Performance comparison of the Minuit2 fit of the DPD model for $J/\psi \to \bar{p}K_S^0\Sigma^+$ on three types of GPU devices and two CPU devices. The fit duration includes callbacks that write current parameter values to disk and the frequency is given by the number of iterations per second. The standard deviations are determined by rerunning the entire analysis two to three times.

Clearly, GPUs have an advantage over CPUs, even with large amplitude models that have non-linear parametrisations. As described in Section 4.1, the reason is that GPUs are designed

to perform many operations in parallel, as long as the operations are independent and do not contain branching logic. It should be noted that the analysis could simply be performed on each of these devices by simply installing `jax[cuda12]`. On the NVIDIA devices we tested, JAX automatically detected the GPU and could run the analysis without any changes to the codebase.

Parallelisation of parameter space

In each iteration during a gradient-descent optimisation, the likelihood is evaluated for one set of parameter values. Here, parallelisation on the hardware devices is achieved by parallelising the evaluation of the intensity function over the data sample and the phase space sample (which is used for normalising the intensity function to a probability-density function). From the simple benchmark presented in the previous section, fit performance depends on how well hardware devices parallelise the computation.

A bottleneck in the gradient-descent process is the copy operation to the optimiser and the decision step that determines the parameter update for the next iteration. Figure 7.23 shows the distribution of evaluation frequencies of the likelihood function during the fits reported in Table 7.2 for the three GPUs. The **frequency** is defined as the inverse of the time required to complete one iteration. Each iteration consists of one evaluation of the likelihood function on the hardware accelerator, in addition to the copy operation and any optimisation logic executed by the optimiser. The figure reveals two types of iterations: fast iterations, in which the backend only evaluates and returns the likelihood for a given parameter set, and slow iterations, in which the optimiser performs additional optimisation logic. The more performant the GPU, the larger the frequency gap between these two types of iterations, and the larger the computational bottleneck, with the NVIDIA A100 showing the largest gap between the evaluation band (high frequency) and optimisation band (lower frequency).

Because of memory locality, hardware accelerators are most efficient when as many operations as possible are executed on the same device. A gradient-descent optimiser such as Minuit2 cannot fully exploit this, as it is inherently sequential and runs in a separate process from the likelihood evaluation. To test how well parallelism can be used instead, we remove the minimiser and evaluate the likelihood for fully randomised parameter values directly on the accelerator. This setup is similar to the uncertainty evaluations in the polarimetry study and allows many likelihoods to be computed simultaneously through broadcasting (Section 4.1), keeping the computation entirely on the device and avoiding copy operations to the CPU until all results are available.

Figure 7.24 shows the evaluation time of the likelihood function in Equation (7.20) for 100,000 randomly generated production couplings on an AMD Ryzen 7 5800X. Since it is impossible to process all couplings simultaneously, the sample was divided into smaller chunks that were passed sequentially to CPU memory. Different chunk sizes, ranging from 1 to 6,000, were tested to assess how the total evaluation time depends on the array size that can be accommodated within the accelerator memory. As shown, filling the memory with batches of 6,000 couplings yields a speed-up of up to a factor of 10. The top panel displays the number of likelihood evaluations (iterations) per second as a function of chunk size; the middle panel shows the total evaluation time for all 100,000 couplings; and the bottom panel illustrates the corresponding memory consumption during the evaluation process. The result suggests that even higher performance could be achieved with devices offering larger memory capacity or with more highly parallelised hardware such as GPUs.

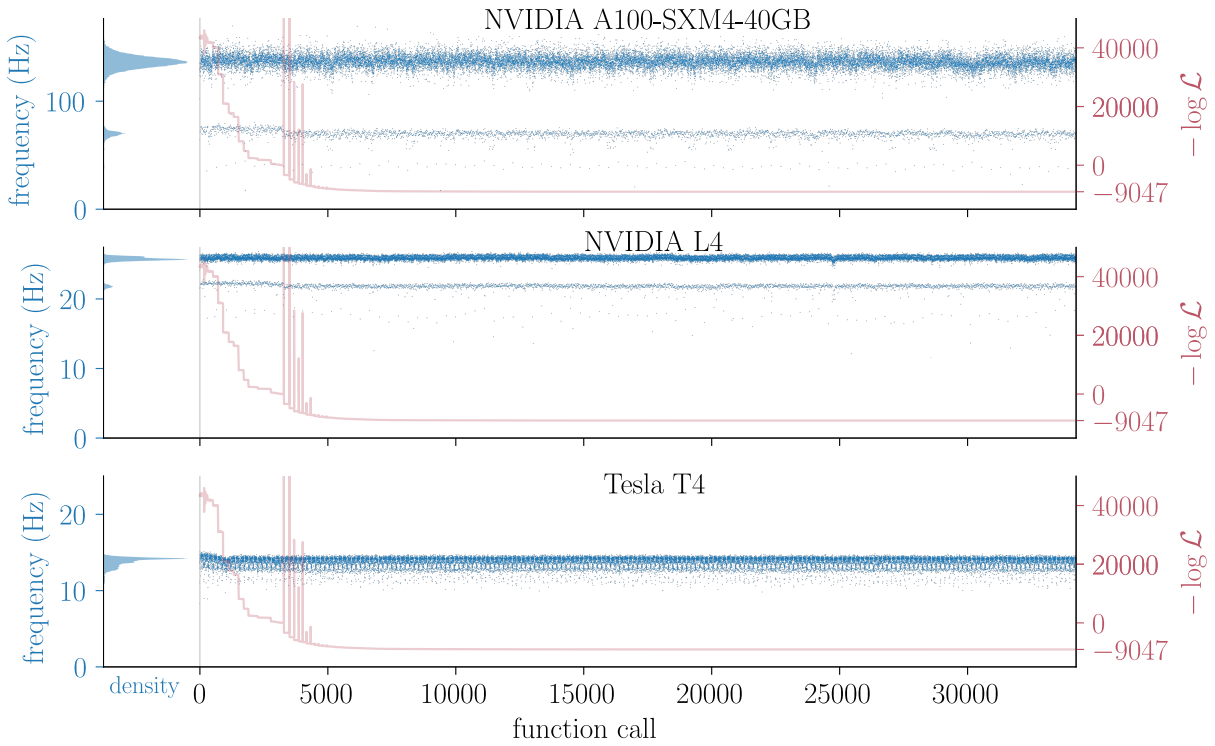


Figure 7.23. Distribution of the evaluation frequency of the likelihood function during the fits performed for Table 7.2 on the three GPUs. Indicated in blue is the duration (frequency in Hz) of each function call, with each histogram on the left showing the distribution of the frequencies during the entire optimisation process. The negative log-likelihood value returned by each function call is shown in red.

This "doubly-parallelised" likelihood evaluation keeps all computations on the hardware accelerator and within its memory. In the analyses described in this chapter, we used such parallelisation techniques primarily for fast uncertainty propagation, but it is equally applicable to likelihood scans. Moreover, whereas traditional gradient-descent methods such as Minuit2 act as local fitters that explore the parameter space sequentially around a minimum, this approach enables the use of inherently parallel global algorithms, such as particle swarm optimisation or stochastic gradient descent, which evaluate many parameter configurations simultaneously. In this way, both local and global strategies can benefit from accelerator memory locality, which makes amplitude-analysis fits with highly dimensional parameter spaces more performant.

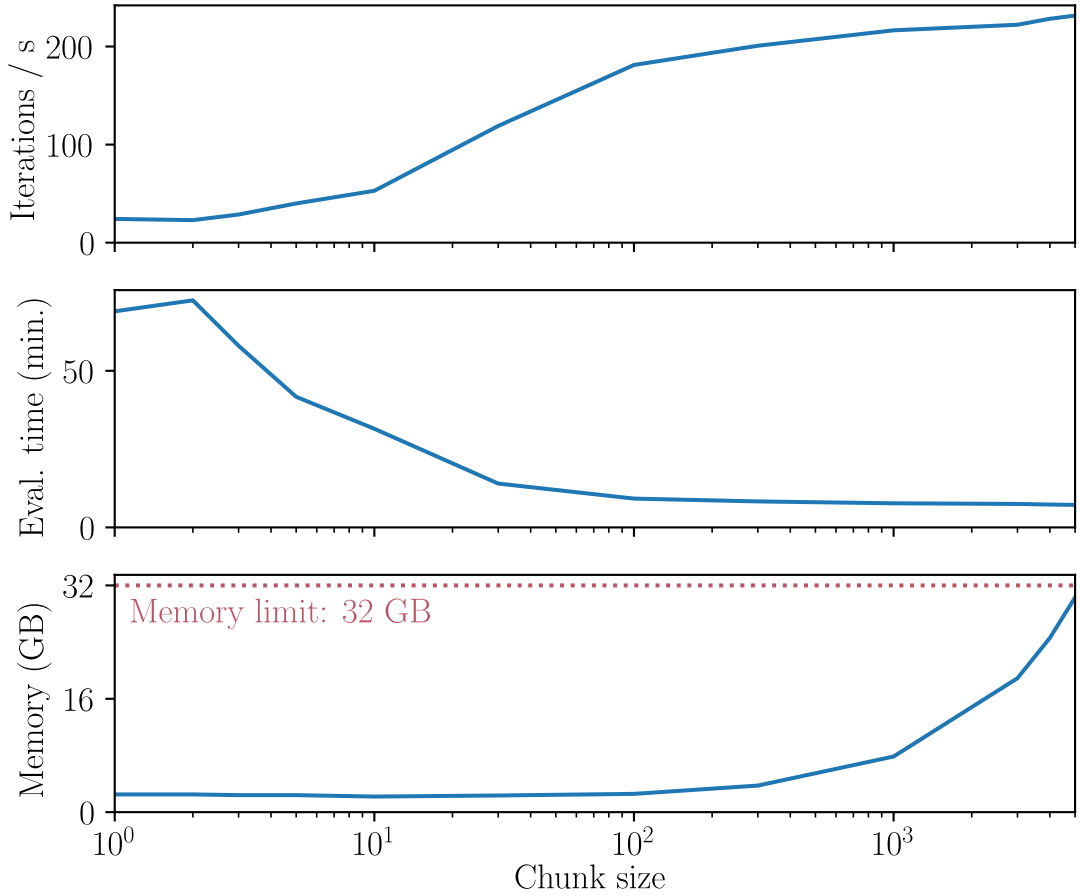


Figure 7.24. Performance of the doubly-parallelised likelihood evaluation of the DPD model in Equation (7.20) for 100 000 randomly generated production couplings on an AMD Ryzen 7 5800X, using JAX broadcasting over the phase-space and data sample with varying array sizes ("chunks"). Top: number of coupling combinations (iterations) evaluated per second. Middle: total time to evaluate the full coupling sample. Bottom: average memory usage during evaluation, with the red line marking the 32 GB RAM limit.

8 Summary and outlook

Hadron physics occupies a distinctive place in the study of nature: it addresses the strongly interacting particles and resonances that make up the vast majority of the visible universe. Within it, hadron spectroscopy is concerned with mapping the spectrum of hadronic states and interpreting their excitations to uncover the principles of the strong interaction. This pursuit relies on a broad set of methods – from lattice QCD, which can make first-principles predictions, to amplitude analyses, which apply the principles of scattering theory to disentangle complex experimental data. The latter approach is particularly challenging and inherently interdisciplinary, as it requires combining theoretical constraints, precise measurements, and advanced computational tools. In this context, the present thesis has focused on baryon spectroscopy – particularly the spectrum of nucleon excitations – and has used this domain as a testbed for a new analysis workflow.

For this work, we transferred the original Common Partial-Wave Analysis (ComPWA) C++ framework to the Python programming language. This opened the door to outsourcing the computations over amplitude models to modern, array-based libraries such as NumPy, JAX, and TensorFlow, removing the workload of maintaining low-level code for numerical evaluation. In addition, these array computing libraries provide many additional advantages out-of-the-box, including automatic parallelisation, GPU acceleration, JIT-compilation, and automatic differentiation.

As a follow-up, we realised that the concept of a function tree around which the ComPWA framework was built, could be generalised with the use of a Computer Algebra System (CAS). This led us to completely formulate amplitude models in terms of symbolic expressions, which the CAS represents as trees of mathematical operations in the back. These trees in turn can be used to generate numerical code – array-based code in particular – for evaluating the model efficiently over large data samples.

The switch to a dynamical programming language also enabled more flexible workflows in the form of Jupyter notebooks and other convenient development tools. While these tools are ideal for quick prototyping when exploring specific challenges in an analysis, they naturally evolve into more general code libraries that can be reused in other analysis pipelines. In addition, with the combination of CAS-assisted model building, the mathematical expressions of the implemented amplitude models can directly visualised in the analysis notebook. Combined with the latest advancements in data science – to render collections of analysis notebooks themselves in the form of a website or as static documents – this turns the formulation and evaluation of amplitude models through notebooks into a transparent and self-documenting workflow.

The transformation of the ComPWA framework has resulted in three main open-source Python libraries that can be used independently. QRules encodes quantum number conservation rules, which can be used to quickly check decay hypotheses and build trees of decay topologies. AmpForm and its extension Ampform-DPD offer a collection of functions for formulating amplitude models as symbolic expressions with SymPy, the most commonly used CAS in the Python ecosystem.

And TensorWaves streamlines the conversion of symbolic models to multiple array-oriented libraries for fitting the models to large data samples with high performance. Not only are these Python libraries easily be extended externally in scripts or extension modules, but the generated symbolic expressions can be modified algebraically to fit the needs of an analysis.

On the physics level, we focused on a correct formulation of amplitude models for three-body decays that have a spinful final state. As discussed, when using the helicity formalism, transition amplitudes of decay chains with different topological structures have phase differences that originate from different Lorentz boost sequences. This work has outlined some of the techniques to deal with these phase differences. The Dalitz-plot decomposition method has been implemented in the AmpForm-DPD library.

The new approach has been validated to floating-point precision by modelling the decay $\Lambda_c \to pK^-\pi^+$ and evaluating it over data from the LHCb experiment. The symbolic workflow with array-based computations turned out to have another advantage: it allowed for the computation of a vector field over the Dalitz plane rather than a conventional intensity model.

Using the same library, we performed the next step in the ongoing analysis of the decay $J/\psi \to \bar{p}\Sigma^+K^0_S$. The previous study formulated the amplitude model with the helicity formalism, but without taking account of spin alignment. A new model formulated with AmpForm-DPD was fit to the same data sample and resulted in an improved description of the data distributions. The spin-aligned model also greatly improved the description of interference between the different subsystems, resulting in decay rates that better agree with expectations.

Future directions

In the course of this study, we encountered many promising opportunities to either broaden the application of the devised workflow or enhance its performance, but these lay beyond the scope of the present thesis. They can be grouped into two broad categories: advances in computation and refinements in physics modelling.

The use of a Computer Algebra System aligns well with a field that relies heavily on theoretical modelling. So far, its immediate benefit has been that implemented amplitude models can be directly inspected as explicit mathematical expressions and manipulated symbolically. Beyond this, a CAS may be leveraged to derive parts of the amplitude models from first principles, rather than constructing them methodologically. AmpForm already provides a small example: the Blatt–Weisskopf form factors are derived from Hankel functions instead of being hard-coded as polynomials. Further possibilities include the use of SymPy's bra–ket classes to obtain specific linear representations of states, or symbolic integration and differentiation to generate parametrisations of the dynamics. A potential drawback is SymPy's speed, which is typically slower than CAS libraries in other languages such as Mathematica. This limitation, however, may be mitigated by serialising and importing pre-derived expressions.

The algebraic features of a CAS may also translate into computational benefits. For instance, we found that symbolic substitution of fixed parameters yields modest performance gains by simplifying the expression trees. Another unexplored avenue would be to algebraically substitute the DPD angles directly into the Wigner d-functions, thereby eliminating many trigonometric functions from the expression tree of the full amplitude model.

In preliminary testing, we found that automatic differentiation does not apply efficiently to large amplitude models. Gradient-descent fits with Minuit showed no overall improvement in

speed or stability, and the strong non-linearity of the models leads to very large function trees that consume substantial memory. This limitation may stem from the current code-generation strategy: our implementation relies on SymPy to produce a single numerical function for the entire model. While numerical frameworks such as JAX or NumPy appear to handle this well in direct model evaluation, it remains to be investigated whether such large, single-function representations hinder differentiation algorithms. If this is the case, code generation would have to be improved by partitioning the generated function into smaller definitions that can be differentiated or JIT-compiled separately.

Another challenge is the parallelisation of computations across multiple devices, such as in a cluster environment. Thus far, we have not explored this avenue, since all computations and fits could be performed on a single device – a preferable setup due to memory locality. In the long term, however, growing data volumes may exceed the capacity of individual devices. Most computational frameworks already provide mechanisms for distributed computing, as they are designed to train large neural networks that demand larger amounts of memory. In the case of JAX, community-maintained extensions such as `mpi4jax` specifically address this issue.

In the short term, a promising opportunity is to parallelise the evaluation of amplitude models across parameter space. Traditionally, amplitude-analysis frameworks reduced fit times by parallelising only the likelihood evaluation. With array-oriented code, however, parallelism can be applied just as easily along a second axis: the parameter space itself. First tests of this "doubly-parallelised" model evaluation performed for this work already indicate a twenty-fold speed-up on simple multithreaded CPU devices. However, to exploit this technique fully, non-sequential fit algorithms would be required. This points toward global fitting strategies rather than the commonly used local gradient-descent Minuit optimiser, and could in turn facilitate more systematic scans of the likelihood landscape, as well as the exploration of alternative hypotheses associated with different local minima of an amplitude model.

The High-Energy Physics community offers a growing ecosystem of Python libraries for data analysis in particle physics. QRules already makes use of one of these – the Scikit-HEP `particle` package – but the ComPWA project could benefit from deeper integration with other tools in this ecosystem. Most notably, the `zfit` optimisation library, which offers a valuable alternative to the optimisation functionality currently provided by TensorWaves.

Another direction for improvement is to broaden the range of numerical backends supported in SymPy's code generation, for example by targeting PyTorch or the upcoming Mojo language. Much of this is already being explored within the SymPy project itself, but it remains to be investigated whether these backends are suitable for the specific requirements of amplitude analysis.

Finally, code generation could be extended to serialise amplitude models into human-readable formats. Conversely, the ComPWA libraries would need to provide methods for importing such serialised amplitude models. This would enable interoperability between different amplitude-analysis frameworks, which would be a large step forwards in the cross-verification of analysis results.

As demonstrated in this thesis, the DPD method yields an improved description of the data distributions, though certain regions of the fit remain imperfect. As outlined, a key requirement for amplitude models is that they respect fundamental constraints such as unitarity and analyticity. Only by employing coupled-channel techniques can channel-specific structures in the spectrum, such as threshold-induced drops and kinks, be described correctly. This in turn enables the reliable

extraction of deeper properties like pole positions and residues. The $\boldsymbol{K}$-matrix framework provides the necessary tools for such analyses. This is particularly relevant to the nucleon excitation spectrum in J/ψ decays, where threshold effects can only be captured by simultaneously modelling related channels such as $J/\psi \to \bar{p}p\eta$, $J/\psi \to \bar{p}\Lambda K^+$, and $J/\psi \to \bar{p}p\eta'$.

CAS-assisted model building appears well suited to this task, especially within the $\boldsymbol{K}$-matrix framework. While AmpForm already provides helper functionality for constructing coupled-channel dynamics parametrisations, its model-building framework does not yet support amplitude models with multiple channels. The main challenge is the pre-computation of the Chew–Mandelstam function for higher angular momenta, which must be evaluated through numerical integration before a fit can begin. Although such analytic continuations have been demonstrated within the symbolic workflow, the class structure for amplitude models needs to be reorganised so that the pre-computed values can be inserted at the appropriate stage of the analysis workflow.

Finally, the techniques developed for evaluating the polarimeter vector field could be applied to the J/ψ analyses. These fields encode information about the spin correlations of the final state, which may help in inspecting and comparing different models. Up to now, model comparisons have relied mainly on conventional information criteria, such as AIC, BIC, or χ^2, and on fit fractions, while a systematic study of interference between subsystems and partial waves has not yet been undertaken. In addition, it may be possible to relate the polarimeter vector field of three-body decay analyses to spin-transfer observables and moment expansions that have been reported for corresponding channels in scattering experiments.

References

[1] E. Castellani, "Reductionism, emergence, and effective field theories," *Stud. Hist. Philos. Sci. B*, vol. 33, no. 2, pp. 251–267, Jun. 2002, 10.1016/S1355-2198(02)00003-5.

[2] R. Harlander, J.-P. Martinez, and G. Schiemann, "The end of the particle era?" *Eur. Phys. J. H*, vol. 48, no. 1, p. 6, Dec. 2023, 10.1140/epjh/s13129-023-00053-4.

[3] S. Capstick, S. Dytman, R. Holt, X. Ji, J. Negele, and E. Swanson, "Key Issues in Hadronic Physics," Dec. 2000, http://arxiv.org/abs/hep-ph/0012238

[4] Y. Ne'eman, *The particle hunters*, 2nd ed. New York: Cambridge University Press, 1986. ISBN: 978-0-521-47107-7

[5] A. Pickering, *Constructing Quarks: A Sociological History of Particle Physics*. University of Chicago Press, 1984. ISBN: 978-0-85224-458-6

[6] S. Schweber, "A Historical Perspective on the Rise of the Standard Model," in *The Rise of the Standard Model*, L. Hoddeson, L. Brown, M. Riordan, and M. Dresden, Eds., New York: Cambridge University Press, 1997, pp. 645–684. 10.1017/CBO9780511471094.040.

[7] E. Rutherford, "The scattering of alpha and beta particles by matter and the structure of the atom," *Philos. Mag.*, vol. 21, no. 125, pp. 669–688, May 1911, 10.1080/14786440508637080.

[8] E. Rutherford, "Collision of α particles with light atoms. IV. An anomalous effect in nitrogen," *Philos. Mag.*, vol. 37, no. 222, pp. 581–587, Jun. 1919, 10.1080/14786431003659230.

[9] E. Rutherford, "Bakerian Lecture: Nuclear constitution of atoms," *Proc. R. Soc. Lond. A*, vol. 97, no. 686, pp. 374–400, Jun. 1920, 10.1098/rspa.1920.0040.

[10] J. Chadwick, "Possible Existence of a Neutron," *Nature*, vol. 129, no. 3252, pp. 312–312, Feb. 1932, 10.1038/129312a0.

[11] G. Uhlenbeck and S. Goudsmit, "Ersetzung der Hypothese vom unmechanischen Zwang durch eine Forderung bezüglich des inneren Verhaltens jedes einzelnen Elektrons," *Naturwiss.*, vol. 13, no. 47, pp. 953–954, Nov. 1925, 10.1007/bf01558878.

[12] W. Pauli, "Über den Zusammenhang des Abschlusses der Elektronengruppen im Atom mit der Komplexstruktur der Spektren," *Z. Phys.*, vol. 31, no. 1, pp. 765–783, Feb. 1925, 10.1007/BF02980631.

[13] E. Fermi, "Zur Quantelung des idealen einatomigen Gases," *Z. Phys.*, vol. 36, no. 11–12, pp. 902–912, Nov. 1926, 10.1007/BF01400221.

[14] P. A. M. Dirac, "On the Theory of quantum mechanics," *Proc. R. Soc. Lond. A*, vol. 112, no. 762, pp. 661–677, Oct. 1926, 10.1098/rspa.1926.0133.

[15] Bose, "Plancks Gesetz und Lichtquantenhypothese," *Z. Phys.*, vol. 26, no. 1, pp. 178–181, Dec. 1924, 10.1007/BF01327326.

[16] H. Yukawa, "On the Interaction of Elementary Particles. I," *Prog. Theor. Phys. Suppl.*, vol. 17, no. PRINT–92–144, pp. 1–10, Feb. 1935, 10.1143/PTPS.1.1.

[17] S. H. Neddermeyer and C. D. Anderson, "Note on the Nature of Cosmic-Ray Particles," *Phys. Rev.*, vol. 51, no. 10, pp. 884–886, May 1937, 10.1103/PhysRev.51.884.

[18] J. Street and E. Stevenson, "New Evidence for the Existence of a Particle of Mass Intermediate Between the Proton and Electron," *Phys. Rev.*, vol. 52, no. 9, pp. 1003–1004, Nov. 1937, 10.1103/PhysRev.52.1003.

[19] C. D. Anderson and S. H. Neddermeyer, "Mesotron (Intermediate Particle) as a Name for the New Particles of Intermediate Mass," *Nature*, vol. 142, no. 3602, pp. 878–878, Nov. 1938, 10.1038/142878c0.

[20] E. Fermi, E. Teller, and V. Weisskopf, "The Decay of Negative Mesotrons in Matter," *Phys. Rev.*, vol. 71, no. 5, pp. 314–315, Mar. 1947, 10.1103/PhysRev.71.314.

[21] C. M. G. Lattes, H. Muirhead, G. P. S. Occhialini, and C. F. Powell, "Processes involving charged mesons," *Nature*, vol. 159, no. 4047, pp. 694–697, May 1947, 10.1038/159694a0.

[22] C. M. G. Lattes, G. P. S. Occhialini, and C. F. Powell, "Observations on the Tracks of Slow Mesons in Photographic Emulsions – Part 1," *Nature*, vol. 160, no. 4066, pp. 453–456, Oct. 1947, 10.1038/160453a0.

[23] L. Rosenfeld, *Nuclear Forces*. Amsterdam: North Holland Publishing Company, 1948. https://archive.org/details/in.ernet.dli.2015.169077

[24] W. Heisenberg, "Über den Bau der Atomkerne. I," *Z. Phys.*, vol. 77, no. 1–2, pp. 1–11, Jun. 1932, 10.1007/BF01342433.

[25] E. P. Wigner, "On the Consequences of the Symmetry of the Nuclear Hamiltonian on the Spectroscopy of Nuclei," *Phys. Rev.*, vol. 51, no. 2, pp. 106–119, Jan. 1937, 10.1103/PhysRev.51.106.

[26] G. Rasche, "Zur Geschichte des Begriffes „Isospin"," *Arch. Rational Mech.*, vol. 7, no. 4, pp. 257–276, 1971, 10.1007/BF00328045.

[27] N. Kemmer, "Quantum theory of Einstein–Bose particles and nuclear interaction," *Proc. R. Soc. Lond. A*, vol. 166, no. 924, pp. 127–153, May 1938, 10.1098/rspa.1938.0084.

[28] H. Fröhlich, W. Heitler, and N. Kemmer, "On the nuclear forces and the magnetic moments of the neutron and the proton," *Proc. R. Soc. Lond. A*, vol. 166, no. 924, pp. 154–177, May 1938, 10.1098/rspa.1938.0085.

[29] R. Bjorklund, W. E. Crandall, B. J. Moyer, and H. F. York, "High Energy Photons from Proton–Nucleon Collisions," *Phys. Rev.*, vol. 77, no. 2, pp. 213–218, Jan. 1950, 10.1103/PhysRev.77.213.

[30] A. Pais, "Ω-Space Theory," in *Proceedings, International Conference on Theoretical Physics: Kyoto & Tokyo, September 14-24, 1953*, I. Imai, A. Harasima, M. Kotani, R. Kubo, R. Nozawa, T. Toyoda, and T. Yamanouchi, Eds., Kyoto & Tokyo: Science Council of Japan, Sep. 1953, pp. 156–163.

[31] J. W. Cronin, "The 1953 Cosmic Ray Conference at Bagnères de Bigorre: The Birth of Sub Atomic Physics," *Eur. Phys. J. H*, vol. 36, no. 2, pp. 183–201, Sep. 2011, 10.1140/epjh/e2011-20014-4.

[32] J. J. Thomson, "Cathode Rays," *Philos. Mag.*, vol. 44, no. 269, pp. 293–316, Oct. 1897, 10.1080/14786449708621070.

[33] C. D. Anderson, "The Positive Electron," *Phys. Rev.*, vol. 43, no. 6, pp. 491–494, Mar. 1933, 10.1103/PhysRev.43.491.

[34] P. A. M. Dirac, "Quantised Singularities in the Electromagnetic Field," *Proc. R. Soc. Lond. A*, vol. 133, no. 821, pp. 60–72, Sep. 1931, 10.1098/rspa.1931.0130.

[35] C. D. Anderson and S. H. Neddermeyer, "Cloud Chamber Observations of Cosmic Rays at 4300 Meters Elevation and Near Sea-Level," *Phys. Rev.*, vol. 50, no. 4, pp. 263–271, Aug. 1936, 10.1103/PhysRev.50.263.

[36] G. D. Rochester and C. C. Butler, "Evidence for the Existence of New Unstable Elementary Particles," *Nature*, vol. 160, no. 4077, pp. 855–857, Dec. 1947, 10.1038/160855a0.

[37] R. Brown, U. Camerini, P. H. Fowler, H. Muirhead, C. F. Powell, and D. M. Ritson, "Observations with Electron-Sensitive Plates Exposed to Cosmic Radiation. Part 2: Further Evidence for the Existence of Unstable Charged Particles of Mass $\sim 1,000\, m_e$, and observations on their mode of decay," *Nature*, vol. 163, no. 4133, pp. 82–87, Jan. 1949, 10.1038/163082a0.

[38] W. B. Fowler, R. P. Shutt, A. M. Thorndike, and W. L. Whittemore, "Production of V_1^0 Particles by Negative Pions in Hydrogen," *Phys. Rev.*, vol. 91, no. 5, p. 1287, Sep. 1953, 10.1103/PhysRev.91.1287.

[39] R. Armenteros, K. H. Barker, C. C. Butler, A. Cachon, and C. M. York, "The properties of charged *V*-particles," *Philos. Mag.*, vol. 43, no. 341, pp. 597–611, Jun. 1952, 10.1080/14786440608520216.

[40] E. W. Cowan, "A *V*-Decay Event with a Heavy Negative Secondary, and Identification of the Secondary *V*-Decay Event in a Cascade," *Phys. Rev.*, vol. 94, no. 1, pp. 161–166, Apr. 1954, 10.1103/PhysRev.94.161.

[41] A. Bonetti, R. L. Setti, M. Panetti, and G. Tomasini, "On the existence of unstable charged particles of hyperprotonic mass," *Nuovo Cimento*, vol. 10, no. 12, pp. 1736–1743, Dec. 1953, 10.1007/BF02781667.

[42] A. Pevsner *et al.*, "Evidence for a Three-Pion Resonance Near 550 Mev," *Phys. Rev. Lett.*, vol. 7, no. 11, pp. 421–423, Dec. 1961, 10.1103/PhysRevLett.7.421.

[43] M. Ikeda, S. Ogawa, and Y. Ohnuki, "A Possibile Symmetry in Sakata's Model for Bosons-Baryons System," *Prog. Theor. Phys.*, vol. 22, no. 5, pp. 715–724, Nov. 1959, 10.1143/PTP.22.715.

[44] V. E. Barnes *et al.*, "Observation of a Hyperon with Strangeness Minus Three," *Phys. Rev. Lett.*, vol. 12, no. 8, pp. 204–206, Feb. 1964, 10.1103/PhysRevLett.12.204.

[45] M. Gell-Mann, "The Eightfold Way: A Theory of Strong Interaction Symmetry," California Institute of Technology, Pasadena, CTSL-20, Mar. 1961. 10.2172/4008239.

[46] V. V. Ezhela *et al.*, Eds., *Particle Physics: One Hundred Years of Discoveries: An annotated chronological bibliography.* Woodbury, NY: American Institute of Physics, 1996. ISBN: 978-1-56396-642-2

[47] H. Kragh, *Quantum Generations: A History of Physics in the Twentieth Century.* Princeton University Press, 1999. ISBN: 978-0-691-01206-3

[48] M. Gell-Mann, "Isotopic Spin and New Unstable Particles," *Phys. Rev.*, vol. 92, no. 3, pp. 833–834, Nov. 1953, 10.1103/PhysRev.92.833.

[49] M. Gell-Mann, "The interpretation of the new particles as displaced charge multiplets," *Nuovo Cimento*, vol. 4, no. S2, pp. 848–866, Apr. 1956, 10.1007/BF02748000.

[50] K. Nishijima, "Charge Independence Theory of *V* Particles," *Prog. Theor. Phys.*, vol. 13, no. 3, pp. 285–304, Mar. 1955, 10.1143/PTP.13.285.

[51] M. Gell-Mann and A. Pais, "Behavior of Neutral Particles under Charge Conjugation," *Phys. Rev.*, vol. 97, no. 5, pp. 1387–1389, Mar. 1955, 10.1103/PhysRev.97.1387.

[52] T. D. Lee and C. N. Yang, "Charge conjugation, a new quantum number G, and selection rules concerning a nucleon-antinucleon system," *Nuovo Cimento*, vol. 3, no. 4, pp. 749–753, Apr. 1956, 10.1007/BF02744530.

[53] L. B. Okun, "The Theory of Weak Interaction," in *11th international conference on high-energy physics*, J. Prentki, Ed., Geneva: CERN, 1962, pp. 845–866. https://inspirehep.net/conferences/1187697

[54] E. C. G. Stückelberg, "Die Wechselwirkungskräfte in der Elektrodynamik und in der Feldtheorie der Kernkräfte (Teil II und III)," *Helv. Phys. Acta*, vol. 11, pp. 299–328, 1938, http://link.springer.com/10.1007/978-3-7643-8878-2_17

[55] Y. Ne'eman, "Derivation of strong interactions from a gauge invariance," *Nucl. Phys.*, vol. 26, no. 2, pp. 222–229, Aug. 1961, 10.1016/0029-5582(61)90134-1.

[56] M. Gell-Mann, "Symmetries of Baryons and Mesons," *Phys. Rev.*, vol. 125, no. 3, pp. 1067–1084, Feb. 1962, 10.1103/PhysRev.125.1067.

[57] M. Gell-Mann, "A schematic model of baryons and mesons," *Phys. Lett.*, vol. 8, no. 3, pp. 214–215, Feb. 1964, 10.1016/S0031-9163(64)92001-3.

[58] G. Zweig, "An SU_3 model for strong interaction symmetry and its breaking. Version 1." CERN Document Server, Jan. 1964. 10.17181/CERN-TH-401.

[59] M. Gell-Mann, *The Quark and the Jaguar: Adventures in the Simple and the Complex*, Repr. London: Abacus, 1994. ISBN: 978-0-8050-7253-2

[60] M. Riordan, *The Hunting of the Quark: A True Story of Modern Physics.* in A Touchstone book. New York: Simon and Schuster, 1987. ISBN: 978-0-671-50466-3

[61] J. D. Bjorken and S. L. Glashow, "Elementary particles and SU(4)," *Phys. Lett.*, vol. 11, no. 3, pp. 255–257, Aug. 1964, 10.1016/0031-9163(64)90433-0.

[62] S. L. Glashow, J. Iliopoulos, and L. Maiani, "Weak Interactions with Lepton-Hadron Symmetry," *Phys. Rev. D*, vol. 2, no. 7, pp. 1285–1292, Oct. 1970, 10.1103/PhysRevD.2.1285.

[63] M. Gell-Mann, "The symmetry group of vector and axial vector currents," *Phys. Phys. Fiz.*, vol. 1, no. 1, pp. 63–75, Jul. 1964, 10.1103/PhysicsPhysiqueFizika.1.63.

[64] E. D. Bloom *et al.*, "High-Energy Inelastic e − p Scattering at 6° and 10°," *Phys. Rev. Lett.*, vol. 23, no. 16, pp. 930–934, Oct. 1969, 10.1103/PhysRevLett.23.930.

[65] M. Breidenbach *et al.*, "Observed Behavior of Highly Inelastic Electron–Proton Scattering," *Phys. Rev. Lett.*, vol. 23, no. 16, pp. 935–939, Oct. 1969, 10.1103/PhysRevLett.23.935.

[66] Particle Data Group Collaboration *et al.*, "Review of Particle Physics," *Phys. Rev. D*, vol. 110, no. 3, p. 030001, Aug. 2024, 10.1103/PhysRevD.110.030001.

[67] H. Fritzsch, M. Gell-Mann, and P. Minkowski, "Vectorlike weak currents and new elementary fermions," *Phys. Lett. B*, vol. 59, no. 3, pp. 256–260, Nov. 1975, 10.1016/0370-2693(75)90040-4.

[68] O. W. Greenberg, "Spin and Unitary-Spin Independence in a Paraquark Model of Baryons and Mesons," *Phys. Rev. Lett.*, vol. 13, no. 20, pp. 598–602, Nov. 1964, 10.1103/PhysRevLett.13.598.

[69] H. Fritzsch and M. Gell-Mann, "Current algebra: Quarks and what else?" in *Proceedings of the XVI International Conference on High Energy Physics*, J. D. Jackson and A. Roberts, Eds., Chicago, 1972, pp. 135–165. https://inspirehep.net/literature/76232

[70] H. Fritzsch, M. Gell-Mann, and H. Leutwyler, "Advantages of the color octet gluon picture," *Phys. Lett. B*, vol. 47, no. 4, pp. 365–368, Nov. 1973, 10.1016/0370-2693(73)90625-4.

[71] H. D. Politzer, "Reliable Perturbative Results for Strong Interactions?" *Phys. Rev. Lett.*, vol. 30, no. 26, pp. 1346–1349, Jun. 1973, 10.1103/PhysRevLett.30.1346.

[72] D. J. Gross and F. Wilczek, "Ultraviolet Behavior of Non-Abelian Gauge Theories," *Phys. Rev. Lett.*, vol. 30, no. 26, pp. 1343–1346, Jun. 1973, 10.1103/PhysRevLett.30.1343.

[73] L. R. Wollenberg, "Measurement of the Branching Fraction of the Decay Channel $J/\psi \to \bar{p}\Sigma^+ K_S^0 + \text{c.c.}$," PhD thesis, Ruhr University Bochum, 2024.

[74] Wikimedia Commons, "Standard Model of Elementary Particles." 2025. https://w.wiki/EeMv

[75] G. F. Chew and S. C. Frautschi, "Regge Trajectories and the Principle of Maximum Strength for Strong Interactions," *Phys. Rev. Lett.*, vol. 8, no. 1, pp. 41–44, Jan. 1962, 10.1103/PhysRevLett.8.41.

[76] A. Thiel, F. Afzal, and Y. Wunderlich, "Light Baryon Spectroscopy," *Prog. Part. Nucl. Phys.*, vol. 125, p. 103949, Jul. 2022, 10.1016/j.ppnp.2022.103949.

[77] S. J. Brodsky, G. F. De Téramond, H. G. Dosch, and J. Erlich, "Light-front holographic QCD and emerging confinement," *Physics Reports*, vol. 584, pp. 1–105, Jul. 2015, 10.1016/j.physrep.2015.05.001.

[78] N. Isgur and G. Karl, "P-wave baryons in the quark model," *Phys. Rev. D*, vol. 18, no. 11, pp. 4187–4205, Dec. 1978, 10.1103/PhysRevD.18.4187.

[79] O. Krehl, C. Hanhart, S. Krewald, and J. Speth, "What is the structure of the Roper resonance?" *Phys. Rev. C*, vol. 62, no. 2, p. 025207, Jul. 2000, 10.1103/PhysRevC.62.025207.

[80] V. Burkert, G. Eichmann, and E. Klempt, "The impact of γN and $\gamma^* N$ interactions on our understanding of nucleon excitations." arXiv, Jun. 2025. 10.48550/arxiv.2506.16482.

[81] U. Löring, B. Ch. Metsch, and H. R. Petry, "The light-baryon spectrum in a relativistic quark model with instanton-induced quark forces: The non-strange-baryon spectrum and ground states," *Eur. Phys. J. A*, vol. 10, no. 4, pp. 395–446, Jun. 2001, 10.1007/s100500170105.

[82] M. Ronniger and B. Ch. Metsch, "Effects of a spin-flavour-dependent interaction on the baryon mass spectrum," *Eur. Phys. J. A*, vol. 47, no. 12, p. 162, Dec. 2011, 10.1140/epja/i2011-11162-8.

[83] N. Isgur, "Why N*'s are important," Newport News, VA: Thomas Jefferson National Accelerator Facility, Jul. 2000. 10.48550/arxiv.nucl-th/0007008.

[84] L. Tiator *et al.*, "Eta and etaprime photoproduction on the nucleon with the isobar model EtaMAID2018," *Eur. Phys. J. A*, vol. 54, no. 12, p. 210, Dec. 2018, 10.1140/epja/i2018-12643-x.

[85] R. A. Arndt, "Partial wave analyses of elastic meson–nucleon scattering," in *Advanced Methods in the Evaluation of Nuclear Scattering Data*, vol. 236, H. J. Krappe and R. Lipperheide, Eds., Berlin ; Heidelberg: Springer, 1985, pp. 166–178. 10.1007/3-540-15990-8_11.

[86] R. A. Arndt, R. L. Workman, Z. Li, and L. D. Roper, "Partial-wave analysis of pion photoproduction," *Phys. Rev. C*, vol. 42, no. 5, pp. 1853–1863, Nov. 1990, 10.1103/PhysRevC.42.1853.

[87] D. M. Manley, "Isospin analysis of low-energy $\pi N \to \pi\pi N$ data and chiral-symmetry breaking," *Phys. Rev. D*, vol. 30, no. 3, pp. 536–540, Aug. 1984, 10.1103/PhysRevD.30.536.

[88] D. Drechsel, O. Hanstein, S. S. Kamalov, and L. Tiator, "A unitary isobar model for pion photo- and electroproduction on the proton up to 1 GeV," *Nucl. Phys. A*, vol. 645, no. 1, pp. 145–174, Jan. 1999, 10.1016/S0375-9474(98)00572-7.

[89] A. V. Anisovich, E. Klempt, A. V. Sarantsev, and U. Thoma, "Partial-wave decomposition of pion and photoproduction amplitudes," *Eur. Phys. J. A*, vol. 24, no. 1, pp. 111–128, Feb. 2005, 10.1140/epja/i2004-10125-6.

[90] M. Döring, C. Hanhart, F. Huang, S. Krewald, and U.-G. Meißner, "Analytic properties of the scattering amplitude and resonances parameters in a meson exchange model," *Nucl. Phys. A*, vol. 829, no. 3–4, pp. 170–209, Oct. 2009, 10.1016/j.nuclphysa.2009.08.010.

[91] M. Mai *et al.*, "Jülich–Bonn–Washington model for pion electroproduction multipoles," *Phys. Rev. C*, vol. 103, no. 6, p. 065204, Jun. 2021, 10.1103/PhysRevC.103.065204.

[92] A. Švarc, M. Hadžimehmedović, H. Osmanović, J. Stahov, L. Tiator, and R. L. Workman, "Introducing the Pietarinen expansion method into the single-channel pole extraction problem," *Phys. Rev. C*, vol. 88, no. 3, p. 035206, Sep. 2013, 10.1103/PhysRevC.88.035206.

[93] BESIII Collaboration *et al.*, "Observation of Two New N^* Resonances in the Decay $\psi(3686) \to p\bar{p}\pi^0$," *Phys. Rev. Lett.*, vol. 110, no. 2, p. 022001, Jan. 2013, 10.1103/PhysRevLett.110.022001.

[94] B. C. Hunt and D. M. Manley, "Updated determination of N^* resonance parameters using a unitary, multichannel formalism," *Phys. Rev. C*, vol. 99, no. 5, p. 055205, May 2019, 10.1103/PhysRevC.99.055205.

[95] J. R. Taylor, *Scattering Theory: The Quantum Theory on Nonrelativistic Collisions*. New York: Wiley, 1972. ISBN: 978-0-471-84900-1

[96] J. A. Wheeler, "On the Mathematical Description of Light Nuclei by the Method of Resonating Group Structure," *Phys. Rev.*, vol. 52, no. 11, pp. 1107–1122, Dec. 1937, 10.1103/PhysRev.52.1107.

[97] W. Heisenberg, "Die „beobachtbaren Größen" in der Theorie der Elementarteilchen," *Z. Phys.*, vol. 120, no. 7–10, pp. 513–538, Jul. 1943, 10.1007/BF01329800.

[98] J. T. Cushing, *Theory Construction and Selection in Modern Physics: The S Matrix*. Cambridge University Press, 1990. 10.1017/CBO9781139170123.

[99] A. S. Blum, "The state is not abolished, it withers away: How quantum field theory became a theory of scattering," *Stud. Hist. Philos. Sci. B*, vol. 60, pp. 46–80, Nov. 2017, 10.1016/j.shpsb.2017.01.004.

[100] S. Weinberg, *The Quantum Theory of Fields, Volume 1: Foundations*. New York: Cambridge University Press, 1995. ISBN: 978-0-521-55001-7

[101] M. L. Goldberger and K. M. Watson, *Collision theory*. New York: John Wiley & Sons, Inc., 2004. ISBN: 978-0-486-43507-7

[102] A. D. Martin and T. D. Spearman, *Elementary Particle Theory*. Amsterdam: North Holland Publishing Company, 1970. ISBN: 978-0-7204-0157-8

[103] R. A. Briceño, J. J. Dudek, and R. D. Young, "Scattering processes and resonances from lattice QCD," *Rev. Mod. Phys.*, vol. 90, no. 2, p. 025001, Apr. 2018, 10.1103/RevModPhys.90.025001.

[104] S. Mizera, "Physics of the analytic S-matrix," *Physics Reports*, vol. 1047, pp. 1–92, Jan. 2024, 10.1016/j.physrep.2023.10.006.

[105] H. M. Nussenzveig, *Causality and dispersion relations*. in Mathematics in science and engineering, no. v. 95. New York: Academic Press, 1972. ISBN: 978-0-12-523050-6

[106] R. G. Newton, *Scattering Theory of Waves and Particles*, 2nd ed. New York: Springer, 1982. ISBN: 978-3-540-10950-1

[107] R. J. Eden, P. V. Landshoff, D. I. Olive, and J. C. Polkinghorne, *The Analytic S-Matrix*. Cambridge University Press, 1966. ISBN: 978-0-521-52336-3

[108] G. F. Chew, *The Analytic S Matrix: A Basis For Nuclear Democracy*. New York: W.A. Benjamin, 1966.

[109] S. Mandelstam, "Determination of the Pion-Nucleon Scattering Amplitude from Dispersion Relations and Unitarity. General Theory," *Phys. Rev.*, vol. 112, no. 4, pp. 1344–1360, Nov. 1958, 10.1103/PhysRev.112.1344.

[110] T. W. B. Kibble, "Kinematics of General Scattering Processes and the Mandelstam Representation," *Phys. Rev.*, vol. 117, no. 4, pp. 1159–1162, Feb. 1960, 10.1103/PhysRev.117.1159.

[111] G. Källén, *Elementary Particle Physics*. Reading, MA: Addison-Wesley, 1964. ISBN: 978-0-201-03575-9

[112] R. Omnès and M. Froissart, *Mandelstam Theory and Regge Poles: An Introduction for Experimentalists*. New York: W.A. Benjamin, 1963. ISBN: 978-1-258-40714-8

[113] G. F. Chew and S. Mandelstam, "Theory of the Low-Energy Pion-Pion Interaction," *Phys. Rev.*, vol. 119, no. 1, pp. 467–477, Jul. 1960, 10.1103/PhysRev.119.467.

[114] A. V. Anisovich, V. V. Anisovich, M. A. Matveev, V. A. Nikonov, J. Nyiri, and A. V. Sarantsev, *Three-Particle Physics and Dispersion Relation Theory*. Singapore: World Scientific, 2013. 10.1142/8779.

[115] N. N. Khuri and S. B. Treiman, "Pion–pion scattering and $K^{\pm} \to 3\pi$ decay," *Phys. Rev.*, vol. 119, no. 3, pp. 1115–1121, Aug. 1960, 10.1103/PhysRev.119.1115.

[116] T. Regge, "Introduction to complex orbital momenta," *Nuovo Cimento*, vol. 14, no. 5, pp. 951–976, Dec. 1959, 10.1007/BF02728177.

[117] M. Jacob and G. C. Wick, "On the general theory of collisions for particles with spin," *Ann. Phys.*, vol. 7, no. 4, pp. 404–428, Aug. 1959, 10.1016/0003-4916(59)90051-X.

[118] L. D. Landau and E. M. Lifšic, *Quantum Mechanics: Non-Relativistic Theory*, 3rd ed. in Course of Theoretical Physics, no. Vol. 3. Singapore: Elsevier, 2007. ISBN: 978-0-7506-3539-4

[119] R. E. Cutkosky *et al.*, "Pion–nucleon partial-wave analysis," *Phys. Rev. D*, vol. 20, no. 11, pp. 2804–2838, Dec. 1979, 10.1103/PhysRevD.20.2804.

[120] R. E. Cutkosky, C. P. Forsyth, J. B. Babcock, R. L. Kelly, and R. E. Hendrick, "Pion–Nucleon Partial Wave Analysis," in *4th international conference on baryon resonances*, Jul. 1980, p. 19. https://inspirehep.net/literature/154488

[121] G. Breit and E. P. Wigner, "Capture of Slow Neutrons," *Phys. Rev.*, vol. 49, no. 7, pp. 519–531, Apr. 1936, 10.1103/PhysRev.49.519.

[122] M. L. Perl, *High Energy Hadron Physics*. New York: Wiley, 1974. ISBN: 978-0-471-68049-9

[123] F. von Hippel and C. Quigg, "Centrifugal-Barrier Effects in Resonance Partial Decay Widths, Shapes, and Production Amplitudes," *Phys. Rev. D*, vol. 5, no. 3, pp. 624–638, Feb. 1972, 10.1103/PhysRevD.5.624.

[124] R. A. Briceño, J. J. Dudek, R. G. Edwards, D. J. Wilson, and Hadron Spectrum Collaboration, "Isoscalar $\pi\pi, K\bar{K}, \eta\eta$ scattering and the σ, f_0, f_2 mesons from QCD," *Phys. Rev. D*, vol. 97, no. JLAB–THY–17–2534, p. 054513, Mar. 2018, 10.1103/PhysRevD.97.054513.

[125] J. M. Blatt and V. F. Weisskopf, *Theoretical Nuclear Physics*. New York: Springer, 1952. 10.1007/978-1-4612-9959-2.

[126] R. H. Dalitz, "On the Strong Interactions of the Strange Particles," *Rev. Mod. Phys.*, vol. 33, no. 3, pp. 471–492, Jul. 1961, 10.1103/RevModPhys.33.471.

[127] S.-U. Chung, J. Brose, R. Hackmann, E. Klempt, S. Spanier, and C. Strassburger, "Partial wave analysis in K-matrix formalism," *Ann. Phys.*, vol. 507, no. 5, pp. 404–430, May 1995, 10.1002/andp.19955070504.

[128] I. J. R. Aitchison, "The K-matrix formalism for overlapping resonances," *Nucl. Phys. A*, vol. 189, no. 2, pp. 417–423, Jul. 1972, 10.1016/0375-9474(72)90305-3.

[129] JPAC Collaboration *et al.*, "Novel approaches in Hadron Spectroscopy," *Prog. Part. Nucl. Phys.*, vol. 127, p. 103981, Nov. 2022, 10.1016/j.ppnp.2022.103981.

[130] M. Mai, U.-G. Meißner, and C. Urbach, "Towards a theory of hadron resonances," *Physics Reports*, vol. 1001, pp. 1–66, Feb. 2023, 10.1016/j.physrep.2022.11.005.

[131] L. D. Roper, "Evidence for a P_{11} Pion-Nucleon resonance at 556 MeV," *Phys. Rev. Lett.*, vol. 12, no. 12, pp. 340–342, Mar. 1964, 10.1103/PhysRevLett.12.340.

[132] V. N. Gribov, Y. Dokshitzer, and J. Nyiri, *Strong Interactions of Hadrons at High Energies: Gribov Lectures on Theoretical Physics*. in Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology, no. 27. New York: Cambridge University Press, 2009. 10.1017/CBO9780511534942.

[133] L. V. Ahlfors, *Complex Analysis: An Introduction to the Theory of Analytic Functions of one Complex Variable*, 2nd ed. in International Series in Pure and Applied Mathematics. New York: MacGraw-Hill, 1966. ISBN: 978-0-07-000656-0

[134] O. Deineka, "Coupled-channel dynamics in hadronic systems," PhD thesis, Johannes Gutenberg University, Mainz, 2023. 10.25358/openscience-8981.

[135] M. Sugawara and A. Kanazawa, "Subtractions in Dispersion Relations," *Phys. Rev.*, vol. 123, no. 5, pp. 1895–1902, Sep. 1961, 10.1103/PhysRev.123.1895.

[136] B. J. Edwards and G. H. Thomas, "Inelastic thresholds and dibaryon resonances," *Phys. Rev. D*, vol. 22, no. 11, pp. 2772–2783, Dec. 1980, 10.1103/PhysRevD.22.2772.

[137] J. H. Reid and N. N. Trofimenkoff, "A generating function for Chew–Mandelstam functions," *J. Math. Phys.*, vol. 25, no. 12, pp. 3540–3544, Dec. 1984, 10.1063/1.526093.

[138] B. Ketzer, B. Grube, and D. Ryabchikov, "Light-meson spectroscopy with COMPASS," *Prog. Part. Nucl. Phys.*, vol. 113, p. 103755, Jul. 2020, 10.1016/j.ppnp.2020.103755.

[139] A. Asokan, M.-N. Tang, F.-K. Guo, C. Hanhart, Y. Kamiya, and U.-G. Meißner, "Can the two-pole structure of the $D_0^*(2300)$ be understood from recent lattice data?" *Eur. Phys. J. C*, vol. 83, no. 9, p. 850, Sep. 2023, 10.1140/epjc/s10052-023-11953-6.

[140] J. L. Basdevant and E. L. Berger, "Unitary coupled-channel analysis of diffractive production of the A_1 resonance," *Phys. Rev. D*, vol. 16, no. 3, pp. 657–678, Aug. 1977, 10.1103/PhysRevD.16.657.

[141] R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Phil. Trans. Roy. Soc. Lond. A*, vol. 222, no. 594–604, pp. 309–368, Jan. 1922, 10.1098/rsta.1922.0009.

[142] L. Lyons, *Statistics for nuclear and particle physicists*. Cambridge University Press, 1986. 10.1017/cbo9781139167710.

[143] F. James and M. Roos, "Minuit – A System for Function Minimization and Analysis of the Parameter Errors and Correlations," *Comput. Phys. Commun.*, vol. 10, no. 6, pp. 343–367, Dec. 1975, 10.1016/0010-4655(75)90039-9.

[144] W. Rarita and J. Schwinger, "On a Theory of Particles with Half-Integral Spin," *Phys. Rev.*, vol. 60, no. 1, pp. 61–61, Jul. 1941, 10.1103/PhysRev.60.61.

[145] S. U. Chung, "Helicity-coupling amplitudes in tensor formalism," *Phys. Rev. D*, vol. 48, no. 3, pp. 1225–1239, Aug. 1993, 10.1103/PhysRevD.48.1225.

[146] V. Filippini, A. Fontana, and A. Rotondi, "Covariant spin tensors in meson spectroscopy," *Phys. Rev. D*, vol. 51, no. 5, pp. 2247–2261, Mar. 1995, 10.1103/PhysRevD.51.2247.

[147] A. V. Anisovich and A. V. Sarantsev, "Partial decay widths of baryons in the spin-momentum operator expansion method," *Eur. Phys. J. A*, vol. 30, no. 2, pp. 427–441, Nov. 2006, 10.1140/epja/i2006-10102-1.

[148] S.-U. Chung and J. M. Friedrich, "Covariant helicity-coupling amplitudes: A new formulation," *Phys. Rev. D*, vol. 78, no. 7, Oct. 2008, 10.1103/PhysRevD.78.074027.

[149] J. D. Richman, "An Experimenter's Guide to the Helicity Formalism." Jun. 1984. https://cds.cern.ch/record/153636

[150] S.-U. Chung, "Spin Formalisms (Updated Version)," Brookhaven National Laboratory, BNL–76975-2006-IR, 890945, Jul. 2014. https://suchung.web.cern.ch/spinfm1.pdf

[151] R. Kutschke, "An Angular Distribution Cookbook." Jan. 1996. https://home.fnal.gov/~kutschke/Angdist/angdist.ps

[152] E. Leader, *Spin in Particle Physics*. Cambridge University Press, 2023. 10.1017/9781009402040.

[153] E. P. Wigner, *Gruppentheorie und Ihre Anwendung Auf Die Quantenmechanik der Atomspektren*, Unveränderter Nachdruck 1977. Wiesbaden: Springer, 1931. 10.1007/978-3-663-02555-9.

[154] S. U. Chung, "Spin formalisms," *CDS*, Mar. 1971, 10.5170/CERN-1971-008.

[155] H. Chen and R.-G. Ping, "Coherent helicity amplitude for sequential decays," *Phys. Rev. D*, vol. 95, no. 7, p. 076010, Apr. 2017, 10.1103/PhysRevD.95.076010.

[156] K. S. Habermann and M. Mikhasenko, "Wigner rotations for cascade reactions," *Phys. Rev. D*, vol. 111, no. 5, p. 056015, Mar. 2025, 10.1103/PhysRevD.111.056015.

[157] M. Mikhasenko *et al.*, "Dalitz-plot decomposition for three-body decays," *Phys. Rev. D*, vol. 101, no. 3, p. 034033, Feb. 2020, 10.1103/PhysRevD.101.034033.

[158] R. Brun and F. Rademakers, "ROOT — An object oriented data analysis framework," *Nucl. Instrum. Methods Phys. Res. A*, vol. 389, no. 1–2, pp. 81–86, Apr. 1997, 10.1016/S0168-9002(97)00048-X.

[159] I. Antcheva *et al.*, "ROOT — A C++ framework for petabyte data storage, statistical analysis and visualization," *Comput. Phys. Commun.*, vol. 180, no. 12, pp. 2499–2512, Dec. 2009, 10.1016/j.cpc.2009.08.005.

[160] B. Stroustrup, "Evolving a language in and for the real world: C++ 1991–2006," in *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, San Diego California: ACM, Jun. 2007. 10.1145/1238844.1238848.

[161] TIOBE, "TIOBE Index," *TIOBE*. Aug. 2025. https://web.archive.org/web/20250826100003/https://www.tiobe.com/tiobe-index

[162] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. New York: Cambridge University Press, 1992. ISBN: 978-0-521-43108-8

[163] K. Iglberger, G. Hager, J. Treibig, and U. Rüde, "Expression Templates Revisited: A Performance Analysis of Current Methodologies," *SIAM J. Sci. Comput.*, vol. 34, no. 2, pp. C42–C69, Jan. 2012, 10.1137/110830125.

[164] M. Herlihy, *The Art of Multiprocessor Programming*, Rev. 1st ed. Waltham, MA: Morgan Kaufmann, 2012. ISBN: 978-0-12-397337-5

[165] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, Mar. 2011, 10.1109/MCSE.2011.37.

[166] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, 10.1038/s41586-020-2649-2.

[167] M. E. Wolf and M. S. Lam, "A Data Locality Optimizing Algorithm," *SIGPLAN Not.*, vol. 26, no. 6, pp. 30–44, Jun. 1991, 10.1145/113446.113449.

[168] K. E. Iverson, "Notation as a tool of thought," *Commun. ACM*, vol. 23, no. 8, pp. 444–465, Aug. 1980, 10.1145/358896.358899.

[169] M. J. Flynn, "Some Computer Organizations and Their Effectiveness," *IEEE Trans. Comput.*, vol. C–21, no. 9, pp. 948–960, Sep. 1972, 10.1109/TC.1972.5009071.

[170] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Trans. Math. Softw.*, vol. 5, no. 3, pp. 308–323, Sep. 1979, 10.1145/355841.355847.

[171] E. Angerson *et al.*, "LAPACK: A portable linear algebra library for high-performance computers," in *Supercomputing '90: Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, New York: IEEE, 1990, pp. 2–11. 10.1109/SUPERC.1990.129995.

[172] N. P. Jouppi *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, Toronto: ACM Press, Jun. 2017, pp. 1–12. 10.1145/3079856.3080246.

[173] G. Sapunov, *Deep Learning with JAX*, 1st ed. New York: Manning Publications Co. LLC, 2024. ISBN: 978-1-63343-888-0

[174] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed. Philadelphia: SIAM, 2008. 10.1137/1.9780898717761.

[175] J. Moses, "Macsyma: A personal history," *J. Symb. Comput.*, vol. 47, no. 2, pp. 123–130, Feb. 2012, 10.1016/j.jsc.2010.08.018.

[176] W. A. Martin and R. J. Fateman, "The MACSYMA system," in *Proceedings of the second ACM Symposium on Symbolic and Algebraic manipulation - SYMSAC '71*, Los Angeles: ACM Press, 1971, pp. 59–75. 10.1145/800204.806267.

[177] S. Wolfram, *Mathematica: A System for Doing Mathematics by Computer*. Redwood City, CA: Addison-Wesley, 1988. ISBN: 978-0-201-19330-5

[178] A. Meurer *et al.*, "SymPy: Symbolic computing in Python," *PeerJ Comput. Sci.*, vol. 3, p. e103, Jan. 2017, 10.7717/peerj-cs.103.

[179] The mpmath development team, "Mpmath: A Python library for arbitrary-precision floating-point arithmetic (version 1.3.0)." 2023. https://mpmath.org

[180] L. Z. Pöpping, "High-performance computations with symbolic K matrix expressions tested on N^* resonances in J/ψ decays with data provided by BESIII," MSc thesis, Ruhr University Bochum, 2024.

[181] U. Schmitt, B. Moser, C. S. Lorenz, and A. Refregier, "`sympy2c`: From symbolic expressions to fast C/C++ functions and ODE solvers in Python." arXiv, Mar. 2022. 10.48550/arxiv.2203.11945.

[182] H. Martiros *et al.*, "SymForce: Symbolic Computation and Code Generation for Robotics," in *Robotics: Science and Systems XVIII*, New York: RSS Foundation, Jun. 2022. 10.15607/RSS.2022.XVIII.041.

[183] S. Gowda, "Symbolic-numeric programming in scientific computing," PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2024. https://dspace.mit.edu/handle/1721.1/155320

[184] H. Barthels, C. Psarras, and P. Bientinesi, "Linnea: Automatic Generation of Efficient Linear Algebra Programs," *ACM Trans. Math. Softw.*, vol. 47, no. 3, pp. 1–26, Sep. 2021, 10.1145/3446632.

[185] J. P. A. Ioannidis, "Why Most Published Research Findings Are False," *PLoS Med.*, vol. 2, no. 8, p. e124, Aug. 2005, 10.1371/journal.pmed.0020124.

[186] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, no. 7604, pp. 452–454, May 2016, 10.1038/533452a.

[187] B. Antunes and D. R. C. Hill, "Reproducibility, Replicability and Repeatability: A survey of reproducible research with a focus on high performance computing," *Comput. Sci. Rev.*, vol. 53, p. 100655, Aug. 2024, 10.1016/j.cosrev.2024.100655.

[188] G. Wilson, J. Bryan, K. Cranston, J. Kitzes, L. Nederbragt, and T. K. Teal, "Good enough practices in scientific computing," *PLoS Comput. Biol.*, vol. 13, no. 6, p. e1005510, Jun. 2017, 10.1371/journal.pcbi.1005510.

[189] J. Kitzes, D. Turek, and F. Deniz, Eds., *The Practice of Reproducible Research: Case Studies and Lessons from the Data-Intensive Sciences.* Oakland, CA: University of California Press, 2018. 10.1525/9780520967779.

[190] V. Stodden, P. Guo, and Z. Ma, "Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals," *PLoS ONE*, vol. 8, no. 6, p. e67111, Jun. 2013, 10.1371/journal.pone.0067111.

[191] T. J. Khoo *et al.*, "Constraints on Future Analysis Metadata Systems in High Energy Physics," *Comput. Softw. Big Sci.*, vol. 6, no. 1, p. 13, Dec. 2022, 10.1007/s41781-022-00086-2.

[192] C. Bierlich *et al.*, "Robust Independent Validation of Experiment and Theory: Rivet version 3," *SciPost Phys.*, vol. 8, no. 2, p. 026, Feb. 2020, 10.21468/SciPostPhys.8.2.026.

[193] V. Stodden and S. Miguez, "Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research," *J. Open Res. Softw.*, vol. 2, no. 1, Jul. 2014, 10.5334/jors.ay.

[194] Y. Gamage, D. Tiwari, M. Monperrus, and B. Baudry, "The Design Space of Lockfiles Across Package Managers." arXiv, Jul. 2025. 10.48550/arxiv.2505.04834.

[195] T. T. Procko and O. Ochoa, "Semantic Science: Publication Beyond the PDF," in *SoutheastCon 2024*, Atlanta, GA: IEEE, Mar. 2024, pp. 207–215. 10.1109/SoutheastCon52093.2024.10500258.

[196] J. Heer, M. Conlen, V. Devireddy, T. Nguyen, and J. Horowitz, "Living Papers: A Language Toolkit for Augmented Scholarly Communication," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, San Francisco: ACM Press, Oct. 2023, pp. 1–13. 10.1145/3586183.3606791.

[197] D. E. Knuth, "Literate Programming," *Comput. J.*, vol. 27, no. 2, pp. 97–111, Feb. 1984, 10.1093/comjnl/27.2.97.

[198] E. Schulte, D. Davison, T. Dye, and C. Dominik, "A Multi-Language Computing Environment for Literate Programming and Reproducible Research," *J. Stat. Soft.*, vol. 46, no. 3, Jan. 2012, 10.18637/jss.v046.i03.

[199] F. Perez and B. E. Granger, "IPython: A System for Interactive Scientific Computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 21–29, 2007, 10.1109/MCSE.2007.53.

[200] J. M. Perkel, "Cut the tyranny of copy-and-paste with these coding tools," *Nature*, vol. 603, no. 7899, pp. 191–192, Mar. 2022, 10.1038/d41586-022-00563-z.

[201] T. Kluyver *et al.*, "Jupyter Notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, IOS Press, 2016, pp. 87–90. 10.3233/978-1-61499-649-1-87.

[202] K. J. Millman and F. Pérez, "Developing Open-Source Scientific Practice," in *Implementing Reproducible Research*, V. Stodden, F. Leisch, and R. D. Peng, Eds., Chapman and Hall/CRC, 2018. 10.1201/9781315373461.

[203] J. M. Perkel, "Why Jupyter is data scientists' computational notebook of choice," *Nature*, vol. 563, no. 7729, pp. 145–146, Nov. 2018, 10.1038/d41586-018-07196-1.

[204] J. J. Allaire, C. Teague, C. Scheidegger, Y. Xie, C. Dervieux, and G. Woodhull, "Quarto." Nov. 2024. 10.5281/zenodo.5960048.

[205] R. Cockett, S. Purves, F. Koch, and M. Morrison, "Continuous Tools for Scientific Publishing: Using MyST Markdown and Curvenote to encourage continuous science practices," in *Python in Science Conference*, Tacoma, WA, Jul. 2024, pp. 121–136. 10.25080/NKVC9349.

[206] ComPWA, "Requirement Specifications, Status August 20, 2012: Next generation Partial Wave Analysis software." Sep. 2012.

[207] T. F. Degener, "TARA — An object-oriented program for a partial wave analysis of sequential two body decays," *Comput. Phys. Commun.*, vol. 118, no. 1, pp. 34–48, Apr. 1999, 10.1016/S0010-4655(98)00194-5.

[208] J. P. Cummings and D. P. Weygand, "An Object-Oriented Approach to Partial Wave Analysis." arXiv, 2003. 10.48550/arXiv.physics/0309052.

[209] N. Berger, "Partial wave analysis at BES III harnessing the power of GPUs," *J. Phys. Conf. Ser.*, vol. 331, no. 3, p. 032005, Dec. 2011, 10.1088/1742-6596/331/3/032005.

[210] J. Back *et al.*, "Laura++: A Dalitz plot fitter," *Comput. Phys. Commun.*, vol. 231, pp. 198–242, May 2018, 10.1016/j.cpc.2018.04.017.

[211] M. Michel *et al.*, "ComPWA: A common amplitude analysis framework for PANDA," *J. Phys. Conf. Ser.*, vol. 513, no. 2, Jun. 2014, 10.1088/1742-6596/513/2/022025.

[212] M. Michel, "Extraction of the scalar wave in $J/\psi \to \gamma\pi^0\pi^0$ using the ComPWA framework," PhD thesis, Johannes Gutenberg University, Mainz, 2016. 10.25358/openscience-4105.

[213] P. Weidenkaff, "Analysis of the decay $D^0 \to K^0_S K^+ K^-$ with the BESIII experiment," PhD thesis, Johannes Gutenberg University, Mainz, 2017. 10.25358/openscience-2781.

[214] S. Pflüger, "Precise determination of the luminosity with the PANDA-luminosity detector and implementation of the helicity formalism for the ComPWA framework for an extraction of the scalar wave in the channel $J/\psi \to \pi^0\pi^0\gamma$," PhD thesis, Ruhr University Bochum, 2018. https://nbn-resolving.org/urn:nbn:de:hbz:294-56142

[215] R. E. Johnson and B. Foote, "Designing Reusable Classes," *J. Object Oriented Program.*, vol. 1, no. 2, pp. 22–35, 1988.

[216] M. Fritsch *et al.*, "(Py)ComPWA - The common Partial Wave Analysis framework." Zenodo, Oct. 2019. 10.5281/zenodo.3479482.

[217] S. Pflüger *et al.*, "PWA Expert System – Rule-based particle reaction problem solver on a quantum number level." Common Partial Wave Analysis, Apr. 2021. https://github.com/ComPWA/expertsystem/releases/tag/0.7.3

[218] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A LLVM-based Python JIT compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, Austin, TX: ACM Press, Nov. 2015, pp. 1–6. 10.1145/2833157.2833162.

[219] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." arXiv, 2016. 10.48550/arXiv.1603.04467.

[220] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library." arXiv, 2019. 10.48550/arXiv.1912.01703.

[221] J. Bradbury *et al.*, "JAX: Composable transformations of Python+NumPy programs." 2018. http://github.com/google/jax

[222] A. Poluektov, A. Merli, A. Mathad, and A. Morris, "TensorFlowAnalysis – A collection of useful functions and example scripts for performing amplitude fits using Google TensorFlow." Nov. 2021. https://gitlab.cern.ch/poluekt/TensorFlowAnalysis/-/tree/5fc1172

[223] A. Poluektov, "AmpliTF – Library of primitives for amplitude analyses in high-energy physics using TensorFlow V2." Jul. 2022. https://github.com/apoluekt/AmpliTF/tree/70c51cb

[224] J. Eschle, A. Puig Navarro, R. Silva Coutinho, and N. Serra, "`zfit`: Scalable pythonic fitting," *SoftwareX*, vol. 11, p. 100508, Jan. 2020, 10.1016/j.softx.2020.100508.

[225] R. E. de Boer and S. Pflüger, "[ADR-001] Amplitude model," *PWA Expert System*. Jan. 2021. https://expertsystem.readthedocs.io/0.7.3/adr/001.html

[226] S. Celles, "Python-constraint – Constraint Solving Problem resolver for Python." Nov. 2018. https://github.com/python-constraint/python-constraint/releases/tag/1.4.0

[227] M. Fritsch, S. Pflüger, R. E. de Boer, W. Gradl, and K. Peters, "ComPWA/qrules: Rule-based particle reaction problem solver on a quantum number level." Zenodo, Jan. 2025. 10.5281/zenodo.14680290.

[228] E. Rodrigues and H. Schreiner, "scikit-hep/particle – Package to deal with particles, the PDG particle data table, PDGIDs, etc." Zenodo, Oct. 2024. 10.5281/zenodo.13897537.

[229] M. Fritsch *et al.*, "ComPWA/ampform: Automatically generate symbolic amplitude models for Partial Wave Analysis." Zenodo, Mar. 2025. 10.5281/zenodo.16800099.

[230] M. Fritsch, S. Pflüger, R. E. de Boer, W. Gradl, and K. Peters, "ComPWA/tensorwaves: Python fitter package for multiple computational back-ends." Zenodo, Jan. 2025. 10.5281/zenodo.14765376.

[231] M. Hatlo, F. James, P. Mato, L. Moneta, M. Winkler, and A. Zsenei, "Developments of mathematical software libraries for the LHC experiments," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2818–2822, Dec. 2005, 10.1109/TNS.2005.860152.

[232] H. Dembinski *et al.*, "scikit-hep/iminuit." Zenodo, Apr. 2025. 10.5281/zenodo.15157028.

[233] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nat. Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2020, 10.1038/s41592-019-0686-2.

[234] M. Fritsch *et al.*, "Common Partial Wave Analysis: A collaboration-independent organisation for amplitude analysis software." Zenodo, Jul. 2022. 10.5281/zenodo.6908149.

[235] G. H. Stark, "The Search for Supersymmetry in Hadronic Final States Using Boosted Object Reconstruction," PhD thesis, University of Chicago, 2018. 10.6082/2r2s-4590.

[236] LHCb Collaboration *et al.*, "Design and performance of the LHCb trigger and full real-time reconstruction in Run 2 of the LHC," *J. Instrum.*, vol. 14, no. 4, pp. P04013–P04013, Apr. 2019, 10.1088/1748-0221/14/04/P04013.

[237] I. Belyaev, G. Carboni, N. Harnew, C. Matteuzzi, and F. Teubert, "The history of LHCb," *Eur. Phys. J. H*, vol. 46, no. 1, p. 3, Dec. 2021, 10.1140/epjh/s13129-021-00002-z.

[238] LHCb Collaboration *et al.*, "The LHCb Detector at the LHC," *J. Instrum.*, vol. 3, no. 8, pp. S08005–S08005, Aug. 2008, 10.1088/1748-0221/3/08/S08005.

[239] LHCb Collaboration, "LHCb detector performance," *Int. J. Mod. Phys. A*, vol. 30, no. 7, p. 1530022, Mar. 2015, 10.1142/S0217751X15300227.

[240] D. Müller, "Adopting new technologies in the LHCb Gauss simulation framework," *EPJ Web Conf.*, vol. 214, p. 02004, 2019, 10.1051/epjconf/201921402004.

[241] BESIII Collaboration *et al.*, "Luminosities and energies of e^+e^- collision data taken between $\sqrt{s}$=4.61 GeV and 4.95 GeV at BESIII," *Chin. Phys. C*, vol. 46, no. 11, p. 113003, Nov. 2022, 10.1088/1674-1137/ac84cc.

[242] BESIII Collaboration, "The Layout of BEPC," *Photo and Video Gallery – Institute of High-Energy Physics*. Jun. 2009. https://web.archive.org/web/20250418140031/https://english.ihep.cas.cn/nw/pavg/index_3.html

[243] M. Küßner, "Coupled channel partial wave analysis of two-photon reactions at BESIII," PhD thesis, Ruhr University Bochum, 2022. 10.13154/294-8590.

[244] R. A. Briere, F. A. Harris, and R. E. Mitchell, "Physics Accomplishments and Future Prospects of the BES Experiments at the Beijing Electron–Positron Collider," *Annu. Rev. Nucl. Part. Sci.*, vol. 66, no. 1, pp. 143–170, Oct. 2016, 10.1146/annurev-nucl-102115-044802.

[245] BESIII Collaboration *et al.*, "Design and construction of the BESIII detector," *Nucl. Instrum. Methods Phys. Res. A*, vol. 614, no. 3, pp. 345–399, Mar. 2010, 10.1016/j.nima.2009.12.050.

[246] Z. Zhu *et al.*, "The BESIII detector magnet," in *Proceedings of the Twentieth International Cryogenic Engineering Conference (ICEC20)*, Beijing: Elsevier, 2005, pp. 593–596. 10.1016/B978-008044559-5/50140-X.

[247] LHCb collaboration *et al.*, "Λ_c^+ polarimetry using the dominant hadronic mode," *J. High Energy Phys.*, vol. 2023, no. 7, p. 228, Jul. 2023, 10.1007/JHEP07(2023)228.

[248] N. Brambilla *et al.*, "Heavy quarkonium: Progress, puzzles, and opportunities," *Eur. Phys. J. C*, vol. 71, no. 2, p. 1534, Feb. 2011, 10.1140/epjc/s10052-010-1534-9.

[249] P. Faccioli, C. Lourenço, J. Seixas, and H. K. Wöhri, "Towards the experimental clarification of quarkonium polarization," *Eur. Phys. J. C*, vol. 69, no. 3, pp. 657–673, Oct. 2010, 10.1140/epjc/s10052-010-1420-5.

[250] M. Butenschoen and B. A. Kniehl, "J/ψ polarization at the Tevatron and the LHC: Nonrelativistic-QCD Factorization at the Crossroads," *Phys. Rev. Lett.*, vol. 108, no. 17, p. 172002, Apr. 2012, 10.1103/PhysRevLett.108.172002.

[251] B. Konig, J. G. Korner, and M. Kramer, "Determination of the $b \to c$ handedness using nonleptonic Λ_c decays," *Phys. Rev. D*, vol. 49, no. 5, pp. 2363–2368, Mar. 1994, 10.1103/PhysRevD.49.2363.

[252] R. Dutta, "$\Lambda_b \to (\Lambda_c, p)\tau\nu$ decays within standard model and beyond," *Phys. Rev. D*, vol. 93, no. 5, p. 054003, Mar. 2016, 10.1103/PhysRevD.93.054003.

[253] S. Shivashankara, W. Wu, and A. Datta, "$\Lambda_b \to \Lambda_c\tau\bar{\nu}_\tau$ decay in the standard model and with new physics," *Phys. Rev. D*, vol. 91, no. 11, p. 115003, Jun. 2015, 10.1103/PhysRevD.91.115003.

[254] X.-Q. Li, Y.-D. Yang, and X. Zhang, "$\Lambda_c \to \Lambda_c\tau\bar{\nu}_\tau$ decay in scalar and vector leptoquark scenarios," *J. High Energy Phys.*, vol. 2, p. 068, Feb. 2017, 10.1007/JHEP02(2017)068.

[255] A. Datta, S. Kamali, S. Meinel, and A. Rashed, "Phenomenology of $\Lambda_b \to \Lambda_c\tau\bar{\nu}_\tau$ using lattice QCD calculations," *J. High Energy Phys.*, vol. 2017, no. 8, p. 131, Aug. 2017, 10.1007/JHEP08(2017)131.

[256] A. Ray, S. Sahoo, and R. Mohanta, "Probing new physics in semileptonic Λ_b decays," *Phys. Rev. D*, vol. 99, no. 1, p. 015015, Jan. 2019, 10.1103/PhysRevD.99.015015.

[257] E. Di Salvo, F. Fontanelli, and Z. J. Ajaltouni, "Detailed study of the $\Lambda_b \to \Lambda_c\tau\bar{\nu}_\tau$ decay," *Int. J. Mod. Phys. A*, vol. 33, no. 29, p. 1850169, Oct. 2018, 10.1142/S0217751X18501695.

[258] N. Penalva, E. Hernández, and J. Nieves, "Further tests of lepton flavour universality from the charged lepton energy distribution in $b \to c$ semileptonic decays: The case of $\Lambda_b \to \Lambda_c\ell\bar{\nu}\ell$," *Phys. Rev. D*, vol. 100, no. 11, p. 113007, Dec. 2019, 10.1103/PhysRevD.100.113007.

[259] M. Ferrillo, A. Mathad, P. Owen, and N. Serra, "Probing effects of new physics in $\Lambda_b^0 \to \Lambda_c^+\mu^-\bar{\nu}_\mu$ decays," *J. High Energy Phys.*, vol. 2019, no. 12, p. 148, Dec. 2019, 10.1007/JHEP12(2019)148.

[260] LHCb Collaboration *et al.*, "Isospin Amplitudes in $\Lambda_b^0 \to J/\psi\Lambda(\Sigma^0)$ and $\Sigma_b^0 \to J/\psi\Sigma^0(\Lambda)$ Decays," *Phys. Rev. Lett.*, vol. 124, no. 11, p. 111802, Mar. 2020, 10.1103/PhysRevLett.124.111802.

[261] M. Davier, L. Duflot, F. Le Diberder, and A. Rougé, "The optimal method for the measurement of tau polarization," *Phys. Lett. B*, vol. 306, no. 3–4, pp. 411–417, Jun. 1993, 10.1016/0370-2693(93)90101-M.

[262] J. F. Cornwell, *Group Theory in Physics: An Introduction*. San Diego, CA: Academic Press, 1997. 10.1016/B978-0-12-189800-7.X5000-6.

[263] R. E. de Boer, M. Mikhasenko, and M. Fritsch, "Λ_c^+ polarimetry using the dominant hadronic mode." Zenodo, Jan. 2023. 10.5281/zenodo.7549056.

[264] LHCb collaboration *et al.*, "Amplitude analysis of the $\Lambda_c^+ \to pK^-\pi^+$ decay and Λ_c^+ baryon polarization measurement in semileptonic beauty hadron decays," *Phys. Rev. D*, vol. 108, no. 1, p. 012023, Jul. 2023, 10.1103/PhysRevD.108.012023.

[265] M. Mikhasenko, "`ThreeBodyDecay.jl` – Julia implementation of the Dalitz-plot decomposition." Zenodo, Oct. 2022. 10.5281/zenodo.7256812.

[266] BESIII Collaboration *et al.*, "Number of J/ψ events at BESIII," *Chin. Phys. C*, vol. 46, no. 7, p. 074001, Jul. 2022, 10.1088/1674-1137/ac5c2e.

[267] L. R. Wollenberg and M. Fritsch, "Measurement of the Branching Fraction for the decay $J/\psi \to \bar{p}\Sigma^+K_S^0 + \text{c.c.}$," BESIII Analysis Memo 484v1.7, Dec. 2022. https://docbes3.ihep.ac.cn/cgi-bin/DocDB/ShowDocument?docid=926&version=42

[268] BESIII Collaboration *et al.*, "Observation and branching fraction measurement of the decay $J/\psi \to \bar{p}\Sigma^+K_S^0 + \text{c.c.}$," *Phys. Rev. D*, vol. 109, no. 1, p. 012006, Jan. 2024, 10.1103/PhysRevD.109.012006.

[269] R.-G. Ping, "Event generators at BESIII," *Chin. Phys. C*, vol. 32, no. 8, pp. 599–602, Aug. 2008, 10.1088/1674-1137/32/8/001.