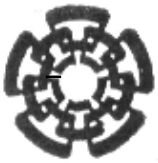


Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

DEPARTAMENTO DE FÍSICA

Simulated Quantum Annealing y SVD para el unfolding de decaimientos $\tau \rightarrow \pi\pi^0\nu$

Tesis que presenta

Carlos Pegueros Denis

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Física

Director de tesis: **Dr. Eduard De La Cruz Burelo**

Ciudad de México Noviembre, 2021

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITECNICO NACIONAL

PHYSICS DEPARTMENT

"Simulated Quantum Annealing and SVD as unfolding techniques for $\tau \to \pi\pi^0\nu$ decays"

Thesis submitted by

Carlos Pegueros Denis

In order to obtain the

Master of Science

degree, speciality in

Physics

Supervisor: **Dr. Eduard De la Cruz Burelo**

Mexico City November, 2021.

Este trabajo está dedicado a todos mis amigos y familiares que me acompañaron y apoyaron en esta etapa tan importante de mi vida; a Maricarmen por toda su paciencia, su amor e invaluables lecciones; a mi asesor, mis sinodales y profesores que me compartieron de su tiempo y su experiencia y a todas las personas con las que coincidí y de las que procuré siempre llevarme algún aprendizaje.

Contents

List of Figures

List of Tables

Abstract

Even though there has been an exponential increase in the development of computing resources in the last decades, the interest of solving even more complex problems has driven human kind to pursue the development of the so called "quantum computers". This new technology is of interest particularly to High Energy Physics because it is an area full of highly-demanding computing problems that could benefit from the quantum speed-up and one of them is the unfolding problem, which consists of removing the distortions induced to measurements by imperfect detectors.

In this work, a quantum algorithm called Quantum Annealing is explored as unfolding technique for the $\tau \to \pi\pi^0\nu$ decays via Monte Carlo simulations and the results are compared with a classic SVD algorithm. The quantity to unfold is the $\pi\pi^0$ invariant mass after going through a simulated version of the Belle II detector. Finally, discussion around the results is presented to explore the viability and benefits to expect from a real quantum computer and directions for future work are provided.

Resumen

A pesar de que en las últimas décadas se ha visto un crecimiento exponencial en la capacidad de procesamiento de cómputo, el interés por resolver problemas cada vez más complicados ha llevado a la humanidad a perseguir el desarrollo de las llamadas "computadoras cuánticas". Particularmente, el área de Física de Altas Energías tiene problemas computacionalmente demandantes que se podrían ver beneficiados por esta tecnología y uno de ellos es el unfolding, que consiste en limpiar las mediciones de las distorsiones inducidas por el detector.

En este trabajo, se explora el álgoritmo cuántico Quantum Annealing cómo técnica de unfolding en los decaimientos $\tau \to \pi\pi^0\nu$ a través de una simulación Monte Carlo y se comparan los resultados con un algoritmo clásico basado en SVD. La cantidad de interés es la masa invariante del par $\pi\pi^0$ y el aparato de medición simulado es el detector Belle II. Finalmente, a partir de los resultados se presenta una discusión sobre la viabilidad y los beneficios que se podrían obtener con una computadora cuántica funcional y se indican futuros caminos por explorar.

Introduction 1

1.1 A problem in High Energy Physics

Since the discovery of the electron back in 1897 by J. J. Thomson's experiment on cathode rays [1] , our theoretical knowledge of the building blocks of the universe (what we call now **"Elementary Particles"**) has vastly increased with the development of Quantum Mechanics and Special Relativity. Moreover, with the massive amount of new technologies that were engineered in the 20th century, we have been able to perform more and more sophisticated experiments, which have allowed us to ask more complex questions to probe the existence of new physics or to challenge the model that best describes the interactions between elementary particles, which is arguably one of the most accurate theories in the history of humankind: the Standard Model of Particle Physics [2] .

1: By observing that cathode rays emitted by a hot filament could be deflected by a magnet, Thomson deduced that they should carry electric charge and by observing the direction of the curvature, it seemed that the rays were, in fact, streams of particles.

2: The Standard Model of Particle Physics [1] [2] is a theory that describes the existence of all the known elementary particles and their interactions with three of the four fundamental forces.

This field of physics is called **"High Energy Physics"** (or just **"Particle Physics"**) because experiments are performed by filling big sophisticated machines (called "particle accelerators") with a large number of particles and splitting them into two beams which are accelerated to the highest possible energy in two opposite directions. These beams are then brought into collision and their interaction produces particles that then transform into multiple other particles in a spontaneous process that is called a **"decay"**. Whatever comes out, is recorded by all the measurement devices distributed in the accelerator and stored for further analysis in the hopes of finding new particles or collisions with interesting properties. From all the possible collisions, or **"events"**, that may be produced, usually only a few of them meet the requirements defined by what is called "interesting" and these events are given the name of **"signal"**, while the rest of the events are called **"background"**.

Measurements of physical observables performed by these experiments are often distorted by various effects related to the technological limitations of the measurement devices, such as a finite resolution, limited acceptance, limited efficiency, etc. As these observables are, for practical reasons, then reconstructed and analyzed as histograms [3] of the observed event counts called "the reconstructed signal", the differences between the ideal histogram described by the theory and the measured histogram gathered from the experiment can be described by the following three effects and are illustrated in figure 1.1:

3: A histogram is a graphical display of frequency distributions.

Detector effects As some event properties such as energies and scattering angles are measured with finite precision, events may be reconstructed in the wrong bin (that is, with the wrong value) or may get lost when building the histogram for analysis.

Statistical fluctuations As the measurements are given by counting the amount of times we observe a given value when repeating an experiment several times, these quantities are in fact random

variables given by a probability distribution which unavoidably include variations.

Background noise Similar events to the signal may be produced in the background by other processes. Some of these events may be too similar to be distinguished and may end up being part of the reconstructed histogram.

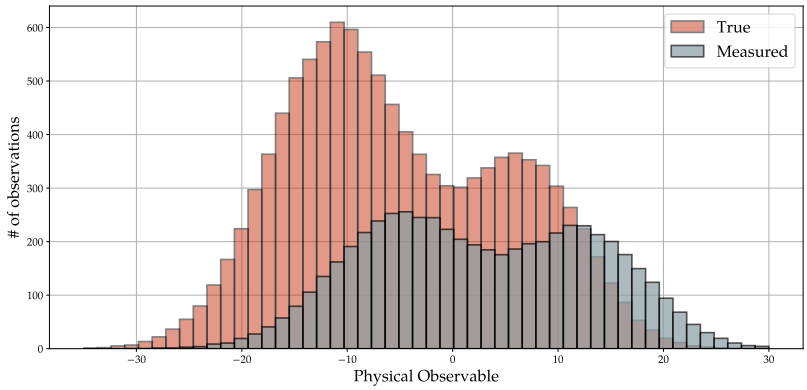


Figure 1.1: Histograms of a physical observable as predicted by the theory (True) and as seen by the detector (Measured). Bars with borders are plotted to show explicitly that these graphs are histograms, but only the outline will be presented for the rest of this work for the sake of a cleaner presentation.

The process of attempting to remove these effects from the reconstructed signal in order to recover the true signal (that is, the signal as predicted by the theory) is known as **"unfolding"**. It is illustrated in figure 1.2 and is of interest because of at least four reasons. Firstly, the beauty of scientific research relies (partly) on its nature of unceasingly looking for the truth, which makes a distorted histogram an unpleasant and unsatisfying result. Secondly, as the undesired effects depend on the measurement device (that is, the experiment that took the measurement), it is necessary to find the true signal if the measurements taken by different experiments want to be compared. Thirdly, unfolded histograms are required if the result is meant to be compared to the predictions made by one or several theoretical models [4] . Finally, if one particular histogram wants to be compared with the prediction of a new theory that was released several years after the measurement was taken, then both the whole data set containing the measurement and the information of the resolution effects of the detector need to be preserved, which is virtually impossible because of the tremendous amount of storage devices it would require [5] .

The unfolding problem is a challenging task to be solved by hand; luckily, the exponential growth of computational power in the last century has allowed us to develop many techniques based on software algorithms to help us tackle this problem. In fact, High Energy Physics is an area full of data-intensive computing because of the large amount of data that needs to be processed and even though we have now developed several techniques such as parallel processing or networks of supercomputers to make these tasks more manageable, it is a promising area to look for applications where quantum computers can help to accelerate the calculations to have a huge impact on the way research is performed.

4: This is partly true as it is possible to **"fold"** the true signal to get an estimate of how the prediction would look like if measured in a real-life scenario, which is usually more convenient.

5: For example, a typical analysis of generic $B^0\bar{B}^0$ events would require about 2.55 TB of data that needs to be shared among several collaborators physically located in different parts of the world.

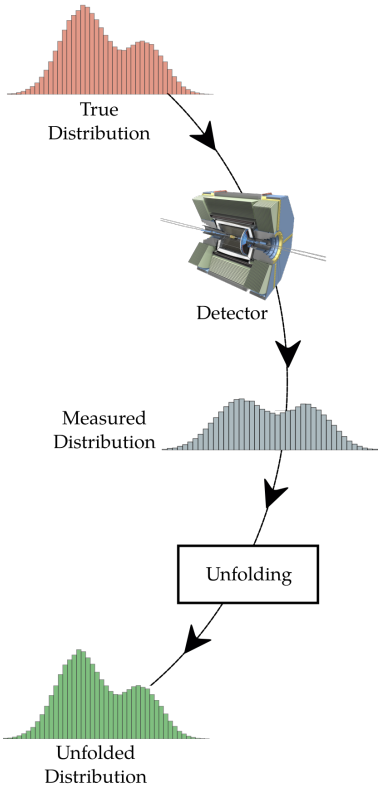


Figure 1.2: The unfolding procedure.

1.2 Quantum Computers. "Nature isn't classical, dammit"

Since Richard Feynman's 1982 seminal paper [3] proposing the idea of using computers governed by the rules of quantum mechanics to perform simulations of the physical world [6] , there has been a growing interest in developing these kind of devices and it is now a very active area of research around the world.

6: "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy." – Richard Feynman

This new type of computers, which we now call **"quantum computers"**, rely on two properties of quantum mechanics to speed up calculations and open a new way of designing algorithms: entanglement and superposition. In its most general form, a quantum computer must be able to prepare an arbitrary quantum state consisting of one or more qubits which then is transformed so that it evolves to output the solution of a problem when the quantum system is observed. A **qubit** can be understood as the quantum version of a bit [7] which, as opposite to its classical counterpart, can be in the states of 0 and 1 at the same time and "collapses" to one of them with a given probability when measured.

7: On a more technical language, a qubit is any normalized state of a two-level quantum-mechanical system.

Despite the immense effort to bring Feynman's idea to life, scientists have encountered many challenges in its physical realization due to the delicate nature intrinsic in manipulating quantum systems, leaving us still years (probably decades or even more) away from having functional quantum computers of general purpose. Nevertheless, there is a select group of problems that can be solved with a very specific kind of quantum computers called **"quantum annealers"** which are easier to build and, ideally, only a few years ahead [8] , making them a promising technology for the near-term future of the quantum era.

8: Actually, there are already some implementations that can solve small problems made by private companies. See D-Wave, for example.

The excitement of the footprint that this coming technology could leave in the way we understand the universe, has led researchers to start thinking of several applications that could motivate the development of quantum computers and High Energy Physics being an area where demanding tasks in terms of processing power form the day-to-day of experimentalists, it represents an important playground where new ideas can be put to test, as is the case of this work.

1.3 The Present Work

In this work, a proposal made by Cormier, Di Sipio and Wittek of using a quantum annealer to solve the unfolding problem [4] is explored in a more realistic scenario given by computational simulations of the $\tau \to \pi\pi^0\nu$ decay folded with the effects of the Belle II detector [5] [6] and its results are compared with a more traditional approach that does not require quantum computers. The observable to unfold is the $\pi\pi^0$ invariant mass and this physics process was chosen because it is a pedagogical exercise used traditionally by the Belle II Collaboration as an introduction to the tools used for data analysis due to its rich content in physics such as resonances, π^0 reconstruction, discrimination between charged pions and leptons, etc[9] , but, in the end, this choice is arbitrary as it is not the main focus of this work.

[4]: Cormier et al. (2019), 'Unfolding measurement distributions via quantum annealing'

9: More on this on chapter 6.

On chapter 1, a description of the unfolding problem and a brief overview of quantum computing and quantum annealing were presented. Then, the first half of this work serves as an introduction of the concepts needed to understand the process and the results of the $\tau \longrightarrow \pi\pi^0\nu$ computational experiment.

Chapter 2 is aimed at giving a brief introduction on probability concepts required to lay the mathematical foundations of the unfolding problem as a more general kind of problem called **the inverse problem**. On chapter 3, the inverse problem is explored in order to depict the intuition behind the difficulties of its resolution, which are then required to assess possible solutions. Chapters 4 and 5 introduce the two main approaches explored in this work to solve the unfolding problem: Singular Value Decomposition (SVD) and Quantum Annealing (QA), respectively. A brief overview on their foundations is given along with some important remarks to be considered on their application.

The second half of this work then focuses on how the computational experiment was carried out.

Chapter 6 gives an introduction on the physics of the decay chosen for the experiment, explains the nature of the data used in the simulations and explains the rationale behind how the "signal" and "background" were selected and reconstructed from data. The analysis framework used for the reconstruction is called **basf2** [7] and is developed by the Belle II Collaboration under the GNU Lesser General Public License at [8].

On chapter 7, it is explained how the distortion effects induced by the simulated Belle II detector were characterized to then show the unfolding results for SVD and Quantum Annealing on a noisy data set and on a cleaner data set.

Chapter 8 presents the conclusions of this work along with discussion around the results and the possible future work.

The code to replicate the unfolding and the rest of this work is available under the MIT License at [9].

Formulating the Unfolding Problem

Unfolding as a Maths Problem 2

This section is heavily based on Chapter 1 and Chapter 11 of [10] by G. Cowan and starts by doing a brief recap of some probability concepts that are required to understand the unfolding problem, which is then formulated in the familiar terms of an equation and its unknowns.

2.1 Probability in Particle Physics

Probability 101

Carrying out experiments is vital for the development of any scientific discipline, but most of the times the outcomes cannot be predicted with complete certainty and are subject to unpredictable variations upon repetition of the experiment under the very same conditions.

Systems with such characteristic are said to be **random** and experiments in Particle Physics are not an exception. In fact, their randomness comes not only from the imperfections of the measurement devices (which is a feature existing in all kinds of experiments), but also to the apparent quantumness inherent in Nature.

The degree of randomness is quantified with the concept of **probability** and even though a formal description can be found on any standard book, for the purposes of this work it can be interpreted as a limiting relative frequency [1] . That is, given a set S called **sample space** containing all the possible outcomes of an experiment, the probability $P(A)$ of observing the outcome A (where A is a subset of S) is given by the fraction of times that A is observed in the limit that the measurement is repeated an infinite amount of times:

1: This is called "the frequentist interpretation."

$$P(A) = \lim_{n\to\infty} \frac{\text{number of occurrences of outcome } A \text{ in } n \text{ measurements}}{n} \tag{2.1}$$

For any two mutually exclusive subsets A and B, the probability of observing A or B is given by the probability assigned to its union and is the sum of the two corresponding probabilities:

$$P(A \cup B) = P(A) + P(B) \tag{2.2}$$

A variable, say x, that takes a specific value of the set S is called a **random variable**. An example of such variables would be the outcome of an experiment in Particle Physics [2] and, considering x a continuous variable, the probability of observing a value within an infinitesimal

2: For instance, the energy of an electron resulting from the beta decay of a radioactive nuclei.

interval $[x, x + dx]$ is given by the **probability density function** (p.d.f) $f(x)$:

$$\text{probability to observe } x \text{ in the interval } [x + dx] = f(x)dx \tag{2.3}$$

And it is normalized such that the total probability (that is, the probability of observing any outcome) is one:

$$\int_S f(x)dx = 1 \tag{2.4}$$

For two subsets A, B of S, one can define the **conditional probability** $P(A|B)$ as the probability of observing the subset A given that B has already been observed as [3] :

3: Equation (2.5) is called "Bayes' Theorem".

$$P(A|B) = \frac{P(B \cap A)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \tag{2.5}$$

And another important definition is the **law of total probability** which breaks up the probability of B to occur as the sum of probabilities of B occurring given that each possible condition for B to occur also occurs:

$$P(B) = \sum_i P(B|A_i)P(A_i) \tag{2.6}$$

What to "expect" from probability?

As the randomness that forces us to adopt a statistical description when describing results of experiments also prevents us from asking questions like "What is the energy of an electron from a beta decay?" [4] , we need to adopt a sensible approach asking questions like "What is the most likely energy to be observed from an electron in a beta decay?". Intuitively, this quantity can be understood as the arithmetic average and is defined as the **expectation value** of x:

4: As the energy is a random variable, there is no single value but instead a set of possible outcomes, so that question becomes non-sense.

$$E[x] = \int_S x f(x)dx \tag{2.7}$$

And another question of interest would be "How widely are the observed energies spread about the mean value?" which, mathematically speaking, can be defined as the variance σ^2:

$$\sigma^2 = E[(x - E[x])^2] \tag{2.8}$$

When generalized to two or more different random variables $\{x, y\}$, the variance is now defined as the **covariance**:

$$\begin{aligned} cov[x, y] &= E[(x - E[x])(y - E[y])] \\ &= E[xy] - E[x]E[y] \end{aligned} \tag{2.9}$$

Which, when generalized for a set of n different random variables $\boldsymbol{x} = \{x_1, x_2, ...\}$, can be defined as a matrix $cov[\boldsymbol{x}]$ which is called **covariance matrix** or **error matrix**:

$$cov[\boldsymbol{x}] = E[(\boldsymbol{x} - E[\boldsymbol{x}])(\boldsymbol{x} - E[\boldsymbol{x}])^T] \tag{2.10}$$

It is worth noting that:

1. Every element of the covariance matrix is the covariance between just two variables $cov[x_i, x_j]$ and by its definition in equation (2.9), the covariance matrix is diagonal if x_i and x_j are independent as $E[x_i x_j] = E[x_i]E[x_j]$ (see equation (2.7)).
2. The covariance can be interpreted as a measure of how two random variables vary together in average (i.e. how much one variable "goes up/down" when the other "goes up/down").
3. The covariance matrix contains all the information about the spread of the variables, which means that:
   - When they are all independent, the square root of the elements in the main diagonal can be interpreted as a confidence interval $x_i \pm \sigma_i$ of where we expect to find most of the values when performing an experiment.
   - When they are not independent, the covariance matrix still contains all the information about the spread of the expected outcome, but it cannot be interpreted as a confidence interval in a single parameter.

A useful property of the covariance matrix that will be used in the following chapter is that of a variable after undergoing a linear transformation A:

$$\boldsymbol{y} = A\boldsymbol{x} \quad \Longrightarrow \quad cov(\boldsymbol{y}) = Acov(\boldsymbol{x})A^T \tag{2.11}$$

Experiments are discrete

Probability in the frequentist interpretation as described in this section is particularly useful in Particle Physics because physical observables are measured by electronic devices that translate changes in its components into digital signals to be processed by computers, which then count the amount of times a value is observed. As digital signals are discrete, the set S of possible outcomes of the experiment is discrete as well, so x would no longer be a continuous variable.

The way of relating this set of observations of the new discrete set $\{x_1, x_2, ..., x_n\}$ to the continuous description we have already presented, is to display them graphically as a **histogram**. A histogram is a graph where the x axis is divided into m subintervals called **bins**, each of a defined width δx_i. The number of times n_i that a value of x in the subinterval/bin i is observed, is given by the vertical axis of the histogram. The total area of the histogram represents the amount of times a given experiment was repeated and, when normalized to one and considering the limit of zero bin width and an infinite amount of observations (that

is, the continuous limit of the discrete x), the histogram of x corresponds to the p.d.f. $f(x)$.

Transforming the description of this section for continuous variables to discrete variables is straightforward by just replacing integrals with sums.

2.2 Mathematical foundation of the unfolding problem

As predictions in High Energy Physics are given by Quantum Field Theory (which inherits the randomness of Quantum Mechanics), physical observables must be treated as random variables, which makes histograms pretty convenient as they can be mapped directly to the probability distributions produced by theory as a means of hypothesis test.

For this section, let us consider an arbitrary physical observable denoted by the random variable y distributed according to the p.d.f. $f_{\text{true}}(y)$ whose functional form is not known a priori. As described in chapter 1, several physical effects introduced by the measurement techniques result in the observation not of y, but of a slightly different variable which we will call x.

$$\begin{aligned} y &= \text{true value of the physical observable (which we do not know)} \\ x &= \text{observed value of the physical observable} \end{aligned} \tag{2.12}$$

And to consider the limited resolution of the electronic equipment that make up the measurement devices (that is, the possibility of an event passing through the detector completely undetected), we will define the probability of an event being measured as $\epsilon(y)$:

$$\epsilon(y) = P(\text{detecting an event with true value } y) \tag{2.13}$$

Remembering that we are not observing individual values of x but histograms of x, to consider a more realistic scenario x needs to be promoted to a discrete variable containing the number of occurrences for each bin. Considering an histogram of N bins, we will call this new set of variables $\boldsymbol{n} = \{n_1, n_2, ..., n_N\}$, but now we are forced to promote y to a discrete variable as well to put observations and predictions on the same ground. For an histogam of M bins, we will call this new set of variables $\boldsymbol{m} = \{m_1, m_2, ..., m_M\}$ [5] :

5: Note that M and N do not have to be necessarily equal.

$$\begin{aligned} \boldsymbol{m} &= \text{the true number of entries per bin (the prediction)} \\ \boldsymbol{n} &= \text{the actual number of entries observed per bin (data)} \end{aligned} \tag{2.14}$$

And the probability of finding the value of y in bin j is given by the integral over that bin:

$$p_j = \int_{\text{bin } j} f_{\text{true}}(y)dy \tag{2.15}$$

Now, $\boldsymbol{m}$ is not only a set of discrete variables, but a set of discrete random variables as it is the result of performing a statistical analysis on the observations, so it does not make sense to talk about a model predicting a concrete value for each m_i and, instead, what we can predict are its expectations values, which will be denoted by the Greek letter μ.

$$\boldsymbol{\mu} = \{\mu_1, \mu_2, ..., \mu_M\} \quad \text{where} \quad \mu_i = E[m_i] \tag{2.16}$$

Again, to put observations and predictions on the same ground, we need to consider expectation values of $\boldsymbol{n}$ as well which will be called ν_i.

$$\boldsymbol{\nu} = \{\nu_1, \nu_2, ..., \nu_N\} \quad \text{where} \quad \nu_i = E[n_i] \tag{2.17}$$

In Particle Physics, it is reasonable to consider the number of occurrences n_i as a random variable distributed by a Poisson distribution (due to the fact that the amount of observations must be an integer value [6]) as in equation (2.18) and make the following assumptions:

6: It is not possible to observe, say, 1.5 events. Either you observe 1 or 2.

1. Because of the the large amount of data collected, it is assumed that the number of occurrences observed in bin i is already the value of ν_i.
2. It is assumed that every event is observed independently of each other, so there is no correlation between the elements of $\boldsymbol{\nu}$ and the covariance matrix is diagonal. For a Poisson distribution, the variances are then given by the expected values $\sigma^2[\nu_i] = E[\nu_i]$

$$P(n_i, \nu_i) = \frac{\nu_i^{n_i} e^{-\nu_i}}{n_i!} \tag{2.18}$$

Note that equation (2.18) is the p.d.f. of each bin element n_i with expectation value ν_i and that the p.d.f. of $\boldsymbol{n}$ can then be obtained just by normalizing the complete histogram. What we want when unfolding is to obtain the p.d.f. of $\boldsymbol{\mu}$ by removing the effects of the detector to then infer the p.d.f. of $\boldsymbol{m}$.

Defining the expectation value of the total number of events found in a bin i as the proportion of events found in that bin from the total amount of events expected to be produced by the experiment μ_{tot}:

$$\nu_i = \mu_{\text{tot}} p_i \tag{2.19}$$

Where $p_i = P(\text{event observed in bin i})$.

By using the law of total probability (2.6) for p_i and considering the limit of infinitesimally small bins, we can relate the quantities we have with the following equation:

$$\begin{aligned} \nu_i &= \mu_{tot} \int dy \underbrace{P\left(\begin{array}{c} \text{observed} \\ \text{in bin i} \end{array} \middle| \begin{array}{c} \text{true value } y \text{ and} \\ \text{event detected} \end{array} \right)}_{\text{spreading effect}} \epsilon(y) f_{\text{true}}(y) \\ &= \mu_{tot} \int_{\text{bin i}} dx \int dy \quad \underbrace{s(x|y)\epsilon(y)}_{\text{response of the detector}} \quad f_{\text{true}}(y) \\ &= \mu_{tot} \int_{\text{bin i}} dx \int dy R(x,y) f_{\text{true}}(y) \end{aligned} \tag{2.20}$$

Where we have defined:

- A spread function $s(x|y)$ as the conditional p.d.f. for a detected event to be observed in bin i
- A response function $R(x,y) = s(x|y)\epsilon(y)$ as the probability to observe x (including the limited efficiency and the spreading effects) given that the true value was y.

Comparing last line of equation (2.20) with equation (2.19), we see that the p.d.f. of the true values $f_{\text{true}}(y)$ has been "folded" with the response function to result in the p.d.f. of the measured values $f_{\text{meas}}(x)$:

$$f_{\text{meas}}(x) = \int R(x,y) f_{\text{true}}(y) dy \tag{2.21}$$

Replacing the integral in y with a sum over the more realistic discrete bins and multiplying both numerator and denominator by μ_j, we can get an expression for ν_i in terms of a simple product:

$$\begin{aligned} \nu_i &= \sum_{j=1}^{M} \frac{\int_{\text{bin i}} dx \int_{\text{bin j}} dy R(x,y) f_{\text{true}}(y)}{(\mu_j/\mu_{\text{tot}})} \mu_j \\ &= \sum_{j=1}^{M} R_{ij} \mu_j \end{aligned} \tag{2.22}$$

where we will denote R as the **response matrix** given by:

$$\begin{aligned} R_{ij} &= \frac{\int_{\text{bin i}} dx \int_{\text{bin j}} dy R(x,y) f_{\text{true}}(y)}{\int_{\text{bin j}} dy f_{\text{true}}(y)} \\ &= \frac{P\left(\text{observed in bin i and true value in bin j}\right)}{P\left(\text{true value in bin j}\right)} \\ &= P(\text{observed in bin j} \mid \text{true value in bin j}) \end{aligned} \tag{2.23}$$

Where we can note that every element R_{ij} in the response matrix is just the conditional probability of finding an event with measured value x in bin i given that the true value y should have been found in bin j. This frequentist interpretation of the response matrix is well-suited for High Energy Physics as one way to find each element R_{ij} is by performing a simulation of the physical experiment and counting the amount of events that should have been observed in bin j, but were observed in bin i after going through the simulated detector.[7]

7: An example on how to obtain the response matrix will be explained in the last part of this work.

A more convenient way of writing equation (2.22) is in terms of the column vectors $\boldsymbol{\nu}$, $\boldsymbol{\mu}$:

$$\boldsymbol{\nu} = R\boldsymbol{\mu} \tag{2.24}$$

The unfolding problem is now described in two versions:

1. The **continuous version** where we need to solve the integral equation (2.21) for $f_{\text{true}}(y)$.
2. The **discrete version** which is described by equation (2.24) where we need to invert matrix R to find the value of $\boldsymbol{\mu}$.

In any case, the result of performing the unfolding will be called **an estimator** for the desired quantity as it is an estimation of the true value. Estimators will be denoted with a hat:

$$\hat{\boldsymbol{\mu}} = \text{estimator for the true value of } \boldsymbol{\mu} \tag{2.25}$$

This problem belongs to a more general class of problems called **the inverse problem** and even though looking harmless, there are many challenges not easy to overcome for its resolution.

The Inverse Problem 3

This chapter aims to present a general overview of the inverse problem to understand why it is not a trivial problem to solve and the issues that need to be addressed when trying to propose a solution.

Firstly, based on Chapter 11 of [10], intuition will be given with a simple example that shows what happens if the most naive solution to the discrete inverse problem is attempted [1] . Then (heavily based on [11]) a mathematical explanation, both for the continuous and discrete case, will be provided and a general approach to avoid these issues will be presented.

1: We pay special attention to the discrete inverse problem as that is the one concerning this work.

3.1 A simple example

To motivate the study of the inverse problem, let us see what happens when trying to naively solve equation (2.24) for $\boldsymbol{\mu}$ by inverting the response matrix.

That is, the estimator for $\boldsymbol{\mu}$ will be given by:

$$\hat{\boldsymbol{\mu}} = R^{-1}\boldsymbol{\nu} \tag{3.1}$$

The first problem that arises is the fact that **there is no guarantee that the response matrix is invertible**. This could happen if, for example, the matrix is not squared (if we would expect more/less bins in one histogram than in the other), or if the matrix happens to be singular. For now, the response matrix was chosen to be squared and invertible to depict the issues of the unfolding, but we will come back to this general case in section 3.4.

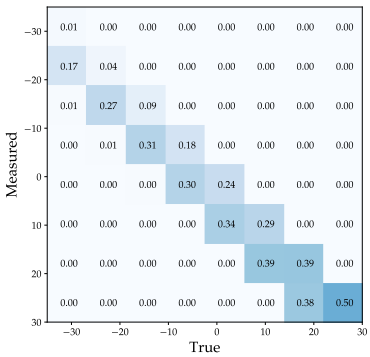
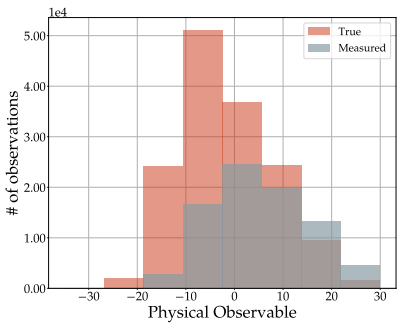


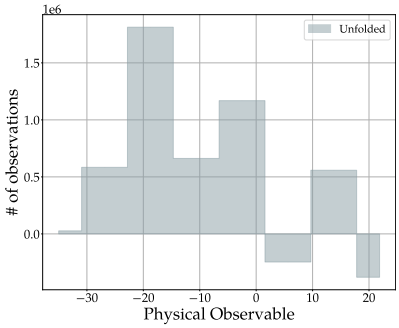
Figure 3.1: The response matrix of the simple example.

Let us consider a detector that is characterized by the response matrix shown in figure 3.1 [2] , that the measured histogram $\boldsymbol{n}$ is shown as the gray plot in figure 3.2a and the goal would be to obtain the true histogram $\boldsymbol{\mu}$ (shown in red in the same figure).

2: An example on how the response matrix can be computed will be left for the last part of this work.



(a) True and Measured histograms



(b) The Unfolded histogram

Figure 3.2: a) The "Measured" histogram to unfold $\boldsymbol{n}$ and the "True" result $\boldsymbol{\mu}$ to find. b) The result of $R^{-1}\boldsymbol{n}$

As mentioned in section 2.2, it will be assumed that the measured data $\boldsymbol{n}$ is already the estimator for $\boldsymbol{\nu}$:

$$\hat{\boldsymbol{\nu}} = \boldsymbol{n} \tag{3.2}$$

So, replacing $\boldsymbol{\nu}$ in equation (3.1) with its estimator (3.2), the unfolded histogram is given by equation (3.3) and is shown as the histogram in figure 3.2b.

$$\hat{\boldsymbol{\mu}} = R^{-1}\boldsymbol{n} \tag{3.3}$$

From this result, we notice that the unfolded histogram is nothing like the True histogram that we were expecting. Not only its shape is different as the unfolded histogram appears to be oscillating, but the oscillations are considerably larger than the expected values on the True histogram (check the differente in the y-axis scale).

In order to understand why this is happening and why alternative solutions are needed, it is illustrative to understand the continuous inverse problem.

3.2 Continuous Inverse Problem

The unfolding problem was given by equation (2.21) and has the general form of a Fredholm Integral equation of the first kind:

$$g(s) = \int_a^b K(s,t)f(t)dt \tag{3.4}$$

Solving equation (3.4) for $g(s)$ given $K(s,t)$ and $f(t)$ is known as the **direct problem** and solving for $f(t)$ given $K(s,t)$ and $g(s)$ is known as the **inverse problem** [3] .

Defining f_p as:

$$f_p(t) = \sin(2\pi pt), \quad p = 1, 2, \ldots \tag{3.5}$$

The Riemann-Lebesgue lemma states that for any arbitrary kernel $K(s,t)$ integrable on an interval $[a,b]$:

$$g_p(s) = \int_a^b K(s,t)f_p(t)dt \to 0 \quad \text{for} \quad p \to \infty \tag{3.6}$$

That is:

- The solution $g(s)$ to the direct problem will be "smoothed" and high-frequency function will be mitigated. The lower the frequency, the stronger the mitigation.
- The solution $f(t)$ to the inverse problem will have the opposite effect and high-frequency functions will be amplified. The higher the frequency, the stronger the amplification.

3: A particular case is when the Kernel function $K(s,t)$ is function of the difference between s and t:

$$g(s) = \int_a^b h(s-t)f(t)dt$$

In that case, the direct/inverse problem is known as a convolution/deconvolution.

In practice, for solving the inverse problem this means that even slight perturbations on $g(s)$ could result in wild amplifications on $f(t)$.

3.3 Discrete Inverse Problem

Going back to the example on section 3.1, we can relate the wild oscillations on the unfolding result if we notice that the estimator used in equation 3.2 is not correct as $\boldsymbol{\nu} \neq \boldsymbol{n}$ because of the background noise and the statistical fluctuations as described in chapter 1.

If we denote these effects with the column vector $\boldsymbol{\beta}$, developing equation (3.3) shows how the result is polluted:

$$\hat{\boldsymbol{\mu}} = R^{-1}\boldsymbol{n} = R^{-1}(\boldsymbol{\nu} + \boldsymbol{\beta}) = \boldsymbol{\mu} + R^{-1}\boldsymbol{\beta} \tag{3.7}$$

If we were able to get an accurate expression for $\boldsymbol{\beta}$, we could directly remove its effects from equation (3.7). As that is never the case and, moreover, statistical fluctuations can be understood as "slight high-frequency perturbations" which get amplified by the Riemann-Lebesgue lemma, huge oscillations in the previous example are expected.

3.4 General Solution to the Inverse Problem

Focusing now on the discrete case, the short example in the beginning of this chapter and the more detailed understanding of the inverse problem have served us to identify the main difficulties to overcome when solving the inverse problem:

1. The response matrix R may not be invertible.
2. Even if R is invertible, the naive solution of inverting the matrix will not work as we are still haunted by the statistical fluctuations that unfold as wild oscillations.

A common approach to overcome both of these issues [4] can be explained in two steps:

4: This is the approach followed in [4] and [12] which are the core of this work, so it will be the only one covered in this thesis.

1. If the matrix is not invertible, instead of trying to solve the inverse problem for an "exact" solution by trying to invert the response matrix, we can instead look for a solution that is not exact, but "good enough" and that looks quite similar to the unfolded histogram we would be expecting.
2. To tame wild oscillations, one could use a Lagrange multiplier to explicitly enforce a well-behaved histogram by looking for a smooth solution of the inverse problem. That is, a solution whose second order derivative is small.

In mathematical terms, a solution to the discrete inverse problem (which in fact represents a solution to the unfolding problem) that includes these two proposals can be posed as a **linear least squares problem**:

$$\min_{\mu} \left\{ \|R\boldsymbol{\mu} - \boldsymbol{\nu}\|_2^2 + \lambda \|D^2\boldsymbol{\mu}\|_2^2 \right\} \tag{3.8}$$

Where:

- $\|\cdot\|_2$ refers to the L2 norm [5] .
- D^2 refers to the discrete Laplacian operator which takes the following form [6] :

$$D^2 = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & -2 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & \ddots & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 & -2 \end{pmatrix} \tag{3.9}$$

- The second term corresponds to the so called **regularization term** which mitigates abrupt changes in the solution $\boldsymbol{\mu}$.
- λ is a scalar called the **regularization weight** which defines the relative weight of this condition. It is a parameter that needs to be chosen when solving the inverse problem.

This general idea of adding a regularization term to enforce some conditions in the desired solution is called **Tikhonov Regularization** and we can notice that this approach allows us to handle situations where $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ have different amount of bins:

$$R \in \mathbb{R}^{m \times n}, \quad \boldsymbol{\mu} \in \mathbb{R}^n, \quad \boldsymbol{\nu} \in \mathbb{R}^m \tag{3.10}$$

That is, the case where the system of linear equations is overdetermined ($m > n$) or underdetermined ($m < n$).

Several computational techniques can be applied to solve the unfolding problem by solving (3.8), but the rest of this work will be focused on two: Singular Value Decomposition (SVD) and Quantum Annealing.

5: The L2 norm for a vector:

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Is defined as:

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{k=1}^{n} |x_k|^2}$$

6: This can be understood as a second-order finite difference with a step $\Delta t = 1$:

$$\begin{aligned} \frac{\delta_t^2 f(x)}{\Delta t^2} &= \frac{\frac{f(x+\Delta t)-f(x)}{\Delta t} - \frac{f(x)-f(x-\Delta t)}{\Delta t}}{\Delta t} \\ &= \frac{f(x+\Delta t) - 2f(x) + f(x-\Delta t)}{\Delta t^2} \\ &= f(x+1) - 2f(x) + f(x-1) \end{aligned}$$

4 Solving via SVD

This chapter is based on [12] and [13] to present an overview on the Singular Value Decomposition (SVD) and how it can be used to solve the inverse problem and, more specifically, the unfolding problem. In the end, some important details to be considered for the computational experiment presented later in this work are addressed.

4.1 Singular Value Decomposition

The Singular Value Decomposition (SVD) is a matrix factorization method which can be used to perform pseudo-inverses of non-square matrices.

For any real valued matrix $X \in \mathbb{R}^{n \times m}$, its SVD is given by:

$$X = U \Sigma V^T \tag{4.1}$$

where:

- $U \in \mathbb{R}^{n \times n}$ is an orthogonal matrix whose columns are called "left singular vectors".
$$UU^T = U^T U = I \tag{4.2}$$
- $V \in \mathbb{R}^{m \times m}$ is an orthogonal matrix whose columns are called "right singular vectors".
$$VV^T = V^T V = I \tag{4.3}$$
- $\Sigma \in \mathbb{R}^{n \times m}$ is a diagonal matrix with non-negative diagonal elements s_i which are called "singular values" and, without loss of generality, are ordered from largest to smallest.
$$\Sigma_{ij} = 0 \text{ for } i \neq j, \quad \Sigma_{ii} \equiv s_i \geq 0, \quad s_i > s_j \text{ for } i > j \tag{4.4}$$

The SVD of a matrix is unique and its definition can be extended to complex valued matrices by replacing the transpose operation $(\cdot)^T$ with the complex conjugate transpose, but as the matrices in the unfolding are real valued, only those will be covered in this work. A pictoric description of the SVD factorization can be seen on figure 4.1.

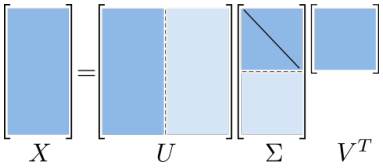


Figure 4.1: SVD of a matrix X

4.2 As a Solution to the Least Squares Problem

The SVD factorization can be used directly to solve a linear least squares problem in the shape:

$$\min_x \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 \tag{4.5}$$

by using the first order optimality condition.

Given:

$$\begin{aligned} e &= \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\ &= (A\boldsymbol{x} - \boldsymbol{b})^T (A\boldsymbol{x} - \boldsymbol{b}) \\ &= \boldsymbol{x}^T A^T A\boldsymbol{x} - 2\boldsymbol{x}^T A^T \boldsymbol{b} + \boldsymbol{b}^T \boldsymbol{b} \end{aligned} \tag{4.6}$$

Its minimum value of $\boldsymbol{x}$ will be given by the solution of:

$$\begin{aligned} \frac{\partial e}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}_{\min}} &= 2A^T A\boldsymbol{x}_{\min} - 2A^T\boldsymbol{b} = 0 \\ A^T A\boldsymbol{x}_{\min} &= A^T\boldsymbol{b} \\ \boldsymbol{x}_{\min} &= (A^T A)^{-1} A^T \boldsymbol{b} \end{aligned} \tag{4.7}$$

Where we have used the fact that $(A^T A)$ is invertible if columns of A are linearly independent.

The result of (4.7) is a common and well-known form of the pseudo-inverse of the SVD factorization as defined in (4.1), but to make this result clearer, we can develop using the SVD factorization of A:

$$\begin{aligned} \boldsymbol{x}_{\min} &= (A^T A)^{-1} A^T b \\ &= (V\Sigma^T \cancel{U^T U} \Sigma V^T)^{-1} V\Sigma^T U^T \boldsymbol{b} \\ &= (V\Sigma^T \Sigma V^T)^{-1} V\Sigma^T U^T \boldsymbol{b} \end{aligned} \tag{4.8}$$

Where we have used the fact that U is orthogonal. Developing the inverse of the term inside the parenthesis [1] :

1: In this case, Σ^{-1} is the pseudo-inverse of Σ.

$$\begin{aligned} \boldsymbol{x}_{\min} &= (V\Sigma^T \Sigma V^T)^{-1} V\Sigma^T U^T \boldsymbol{b} \\ &= (V^T)^{-1} \Sigma^{-1} \cancel{(\Sigma^T)^{-1}} \cancel{V^{-1}} \cancel{V} \cancel{\Sigma^T} U^T \boldsymbol{b} \\ &= (V^T)^{-1} \Sigma^{-1} U^T \boldsymbol{b} \\ &= V\Sigma^{-1} U^T \boldsymbol{b} \end{aligned} \tag{4.9}$$

Where, in the last line, we used the fact that V is orthogonal.

4.3 As a Solution to the Unfolding Problem

Using the SVD decomposition to solve the unfolding problem translates to mapping the general solution with Tikhonov regularization of the inverse problem (3.8) (which is written below for reference) to the general shape (4.5) to then use the result of the previous section (4.9) to find the histogram that best solves our problem.

$$\min_{\mu} \left\{ \|R\boldsymbol{\mu} - \boldsymbol{\nu}\|_2^2 + \lambda \|D^2 \boldsymbol{\mu}\|_2^2 \right\}$$

The details of the approach can be found at [12] and the main idea is summarised below[2] :

First, the covariance matrix N of the histogram to unfold $\boldsymbol{\nu}$ has to be computed as it will be used in the procedure and the response matrix R needs to be estimated by counting the bin migrations from a simulated histogram $\boldsymbol{\mu}^{\text{ini}}$ to a folded histogram $\boldsymbol{\nu}^{\text{ini}}$.

Then, as is common in optimization problems, unknowns $\boldsymbol{\mu}$ need to be normalized to make sure all of them are taken on the same ground without priorities and this is achieved by dividing them with the values of the bins $\boldsymbol{\mu}^{\text{ini}}$ used to build the response matrix R and performing the unfolding on the new scaled unknowns $\boldsymbol{\omega}$ (note that this procedure can easily be undone by solving equation (4.10) for μ_j once ω_j is found) [3] :

$$\omega_j = \mu_j / \mu_{\text{ini},j} \tag{4.10}$$

Normalization is important as it prevents minimization techniques from focusing on finding the minimum of the unknown that impacts the most on the overall value while neglecting the rest [4] , but equations need to be normalized as well so all of them are equally weighted and this can be achieved in a similar fashion by multiplying by the inverse of N to rescale R and $\boldsymbol{\nu}$ (which will be labeled as $\hat{R}$ and $\hat{\boldsymbol{\nu}}$, respectively):

$$\|R\boldsymbol{\omega} - \boldsymbol{\nu}\|_2^2 \rightarrow (R\boldsymbol{\omega} - \boldsymbol{\nu})^T N^{-1} (R\boldsymbol{\omega} - \boldsymbol{\nu}) \tag{4.11}$$

So, after applying transformations (4.10) and (4.11) to the main equation (3.8), the least squared problem to solve is now:

$$\min_{\omega} \left\{ (\tilde{R}\boldsymbol{\omega} - \tilde{\boldsymbol{\nu}})^T (\tilde{R}\boldsymbol{\omega} - \tilde{\boldsymbol{\nu}}) + \lambda (D^2\boldsymbol{\omega})^T (D^2\boldsymbol{\omega}) \right\} \tag{4.12}$$

And we can now use the fact that, for arbitrary vectors y and z:

$$\left\| \begin{pmatrix} y \\ z \end{pmatrix} \right\|_2^2 = \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} y \\ z \end{pmatrix} = y^T y + z^T z = \|y\|_2^2 + \|z\|_2^2 \tag{4.13}$$

To rewrite the argument of (4.12) in terms of a single matrix equation to minimize:

$$\begin{pmatrix} \tilde{R} \\ \sqrt{\lambda} D^2 \end{pmatrix} \boldsymbol{\omega} = \begin{pmatrix} \tilde{\boldsymbol{\nu}} \\ 0 \end{pmatrix} \quad \text{equivalent to} \quad \begin{pmatrix} \tilde{R} D^{-2} \\ \sqrt{\lambda} \end{pmatrix} D^2 \boldsymbol{\omega} = \begin{pmatrix} \tilde{\boldsymbol{\nu}} \\ 0 \end{pmatrix} \tag{4.14}$$

Which now can be solved using the result (4.9) in two steps following the procedure of [12].

The first step is to solve for the case when $\lambda = 0$, which is straightforward as defining M and $\boldsymbol{y}$ as [5] :

$$\tilde{R} D^{-2} \equiv M \quad , \quad D^2 \boldsymbol{\omega} \equiv \boldsymbol{y} \tag{4.15}$$

2: Note that the notation has been changed to keep it uniform throughout this work:

- A in [12] → R in here
- x → $\boldsymbol{\mu}$
- b → $\boldsymbol{\nu}$
- B → N
- $\tau \rightarrow \lambda$
- C → D^2
- $x^{\text{ini}} \rightarrow \boldsymbol{\mu}^{\text{ini}}$
- $b^{\text{ini}} \rightarrow \boldsymbol{\nu}^{\text{ini}}$

3: In this case, the notation does not change with respect to [12].

4: Which in this case would traduce in finding the best fit for the bin with the highest amount of events while overlooking bins with very few events

5: Note that this is not explained the same on [12], so notation varies.

Allows to write the upper element of (4.14) in the shape of (4.5):

$$(\tilde{R}D^{-2})D^2\boldsymbol{\omega} = \tilde{\boldsymbol{\nu}} \quad \Longrightarrow \quad M\boldsymbol{y} = \tilde{\boldsymbol{\nu}} \tag{4.16}$$

Which, by using the SVD factorization of $M = U\Sigma V^T$, allows the direct use of (4.9) to solve for $\boldsymbol{y}_{\text{min}}$ and then for $\boldsymbol{\omega}_{\text{min}}$ by inverting the relation of (4.15):

$$\boldsymbol{y}_{\text{min}} = V\Sigma^{-1}U^T\tilde{\boldsymbol{\nu}} \quad \Longrightarrow \quad \boldsymbol{\omega}_{\text{min}} = D^{-2}V\Sigma^{-1}U^T\tilde{\boldsymbol{\nu}} \tag{4.17}$$

The second step is to use the result of [14] (section 4) so the solution with regularization ($\lambda \neq 0$) can be worked to be:

$$y^{(\lambda)}_{\text{min},i} = \frac{(U^T\tilde{\boldsymbol{\nu}})_i s_i}{s_i^2 + \lambda} \quad \Longrightarrow \quad \boldsymbol{\omega}^{(\lambda)}_{\text{min}} = D^{-2}V\boldsymbol{y}^{(\lambda)}_{\text{min}} \tag{4.18}$$

Finally, for both cases with or without regularization (that is, for $\boldsymbol{\omega}^{(\lambda)}_{\text{min}}$ or $\boldsymbol{\omega}_{\text{min}}$), the histogram of interest $\boldsymbol{\mu}_{\text{min}}$ can be derived by undoing the scaling (4.10) of $\boldsymbol{\omega}_{\text{min}}$:

$$\begin{gathered} \boldsymbol{\mu}_{\text{min}} = \boldsymbol{\mu}_{\text{ini}} \odot \boldsymbol{\omega}_{\text{min}} \\ \text{or} \\ \boldsymbol{\mu}^{(\lambda)}_{\text{min}} = \boldsymbol{\mu}_{\text{ini}} \odot \boldsymbol{\omega}^{(\lambda)}_{\text{min}} \end{gathered} \tag{4.19}$$

Where $\odot$ represents the element-wise product.

Important Remarks

One peculiarity of this unfolding technique is that [12] offers a way to choose the regularization weight by plotting the log magnitude of the coefficient $d_i = (U^T\tilde{\boldsymbol{b}})_i$ [6] vs the index i and setting λ to the square of the k-th singular vector s_k where k represents the number of significant coefficients d_i with a significant amplitude ($|d_i| \gg 1$).

6: The i-th column of the vector d is the coefficient in the decomposition of the measured (and rescaled) histogram $\tilde{\boldsymbol{b}}$ in front of a basis function defined by the i-th column of the rotation matrix U.

Also, the covariance matrix $cov(\boldsymbol{\mu}_{\text{min}})$ of the resulting histogram can be computed explicitly by using the error propagation technique of (2.11) as equations (4.19) and (4.18) allow us to write $\boldsymbol{\mu}_{\text{min}}$ as a linear transformation of z_{min} for which the covariance matrix is calculated on [12]:

$$\begin{aligned} cov(\boldsymbol{\mu}_{\text{min}}) &= \boldsymbol{\mu}_{\text{ini}} cov(\omega_{\text{min}})\boldsymbol{\mu}^T_{\text{ini}} \\ &= \boldsymbol{\mu}_{\text{ini}} D^{-2}V cov(z^{(\lambda)}_{\text{min}})V^T(D^{-2})^T\boldsymbol{\mu}^T_{\text{ini}} \end{aligned} \tag{4.20}$$

Where:

$$cov(z^{(\lambda)}_{\text{min}})_{ik} = \frac{s_i^2}{(s_i^2 + \lambda)^2}\delta_{ik} \tag{4.21}$$

Note that this resulting covariance matrix will not necessarily be diagonal, so there is no simple interpretation for its relation with the errors or the confidence interval of the unfolded prediction. On [12], the square root of the main diagonal is what is shown in the plots of the predictions as erorr bars and that is the approach followed in this work as well.

For a broader discussion on the error interpretation, see [15].

Solving via Quantum Annealing

5

This chapter is based on [16] and [4] to present an overview on Quantum Annealing (QA) and how it can be used to solve the inverse problem and, more specifically, the unfolding problem. In the end, some important details to be considered for the computational experiment presented later in this work are addressed.

5.1 Quantum Annealing

Quantum Annealing is a technique proposed in 1998 [17] to solve binary optimization problems by encoding them in a transverse Ising model such that introducing quantum fluctuations to find the ground state of its Hamiltonian translates to finding the optimal solution to a problem.

To begin with, **"annealing"** refers to a process of letting a system "cool down" in order to improve its stability. In Classical Annealing, this cooling down refers to a heat treatment where temperatures are varied from high to low, while in Quantum Annealing it refers to the amount of quantum fluctuations induced to the system.

The Ising model consists of discrete variables that represent magnetic dipole moments of atomic spins that can be in one of two states (+1 or -1) when aligned or anti-aligned with an external magnetic field[1] . The spins are arranged in a graph, usually a lattice, allowing each spin to interact primarily with its neighbors.

1: The convention is to set the magnetic field in the z-direction so that +1 represents "up" and -1 represents "down".

The Hamiltonian for a lattice of n spins $\{\hat{\sigma}_1^z, ..., \hat{\sigma}_n^z\}$ has the following shape:

$$\hat{H} = -\sum_{i=1}^{n} h_i \hat{\sigma}_i^z - \sum_{i<j} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z \tag{5.1}$$

Where:

- The first term represents the energy that arises from the interaction of each spin with the external magnetic field and h_i represents the energy when the spin is aligned or anti-aligned [2] .
- The second term represents the interaction between neighboring spins and J_{ij} is the exchange energy between spin i and spin j such that aligned spins contribute to a decrease in the energy of J while anti-aligned spins increase it by J. [3]

2: In many textbooks, the h constant is split into the product of the magnetic moment of the spins μ and the external magnetic field B as $h_i = B\mu_i$.

3: The sign of J tells whether neighbors prefer to align or anti-align. In this case, a positive J indicates that they prefer to align as that is the configuration with the lowest energy.

And a general state $|\Psi\rangle$ in the eigenbasis of $\hat{H}$ is given by:

$$|\Psi\rangle = |q_0\rangle \otimes |q_1\rangle \otimes ... \otimes |q_n\rangle \tag{5.2}$$

Where:

$$|q_i\rangle = \begin{cases} |0\rangle & \text{if spin is down} \\ |1\rangle & \text{if spin is up} \end{cases} \tag{5.3}$$

The Ising Model is convenient for quantum computing as every spin can be thought as a qubit due to it being a two-level system and it can be translated to binary bits by replacing the Pauli operators $\hat{\sigma}_i^z$ with new operators $\hat{q}_i$ such that a bit is found when measured:

$$\hat{\sigma}_i^z \quad \rightarrow \quad \hat{q}_i \equiv \frac{1-\hat{\sigma}_i^z}{2} \quad \text{such that} \quad \begin{matrix} \hat{q}_i|0\rangle_i = 0|0\rangle_i \\ \hat{q}_i|1\rangle_i = 1|1\rangle_i \end{matrix} \tag{5.4}$$

This model is of special interest in computational complexity theory because it belongs to a special kind of decision problems whose resolution is too complicated to be performed on classical computers called **NP (nondeterministic polynomial time) problems** [18].

For short, in computational complexity theory there are four kinds of decision problems:

- **P:** all problems solvable, deterministically, in polynomial time.
- **NP:** problems that are hard to solve [4] , but easy to test once a solution candidate is given.
- **NP-Hard:** not necessarily decision problems [5] , but still considered as difficult to solve as NP problems.
- **NP-Complete:** problems such that they can formulate any other NP problem.

4: They are heuristically considered impossible to solve, but no formal proof has been given.

5: They can be optimization problems as every optimization problem can be casted to a decision problem if instead of asking "What is the minimum/maximum value?", we ask "Is there a value lower/greater than X?"

In fact, solving the Ising Model (that is, finding its ground state) in its full 3D form is an NP-Complete problem [18], so finding a feasible way to solve it in a reasonable amount of time, would be a huge improvement in today's technology and an indisputable supremacy of quantum computers.

Merging quantum computing with the annealing technique involves the use of the adiabatic theorem, which states [19] that:

```
A physical system remains in its instantaneous eigenstate if a
    given perturbation is acting on it slowly enough and if there
     is a gap between the eigenvalue and the rest of the
    Hamiltonian's spectrum.
```

That is, if a quantum system starts on its ground state and evolves adiabatically [6] such that it ends up being a completely different system, it will be found in its new ground state.

6: "Adiabatically" is the physics argot for "slow enough".

This makes annealing doable on a quantum computer if it is capable of booting on the ground state of a simple Hamiltonian that adiabatically[7] evolves such that its Hamiltonian is converted to the Ising Model of the problem of interest. If the problem to solve has its solution as the ground state of its Ising Hamiltonian, measuring the quantum system will result with a high probability in the correct answer.

7: Adiabatic process do not exist in real life, but very slow processes are good approximations.

In practice, this is achieved by booting the quantum computer on the ground state of a transverse magnetic field (let us say, in the x direction)[8] whose strength $\Gamma(t)$ is slowly decreased while the magnetic field in the

8: Any observable whose operator does not commute with $\hat{\sigma}^z$ would work, but using $\hat{\sigma}^x$ is the simplest.

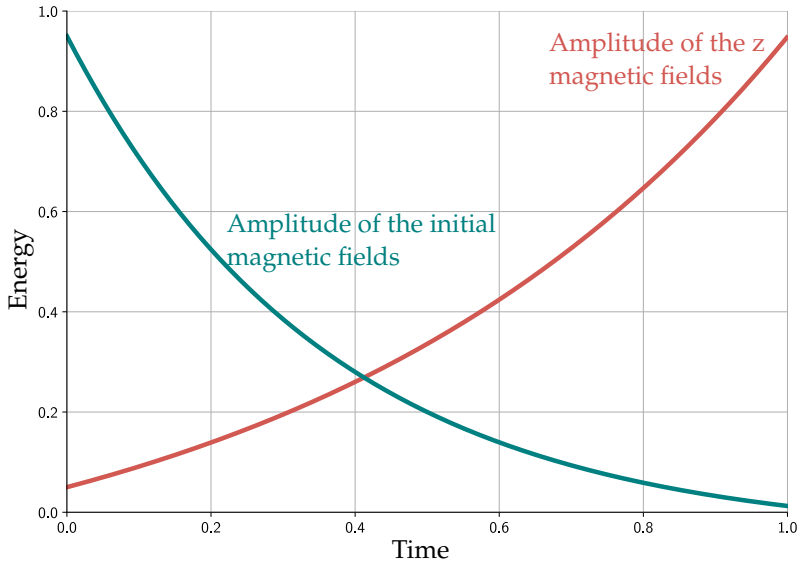


Figure 5.1: The annealing procedure in terms of magnetic fields.

z direction is turned on at a similar pace such that, in the end, it becomes the only magnetic field remaining (as shown in plot 5.1).

The whole process is depicted on figure 5.2 and will have the following Hamiltonian:

$$\hat{H} = \underbrace{-\Gamma(t)\sum_i \hat{\sigma}_j^x}_{H_0} \underbrace{-\sum_{i=1}^{n} h_i\hat{\sigma}_i^z - \sum_{i<j} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z}_{H_1} \tag{5.5}$$

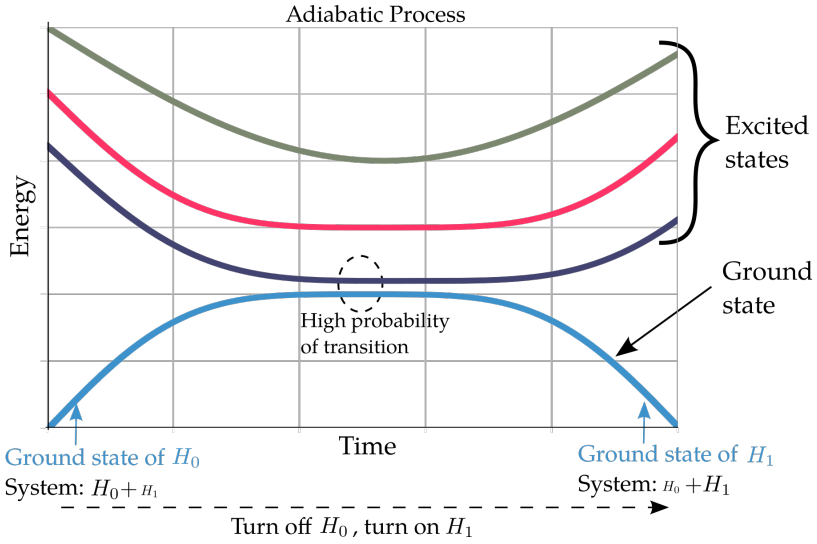


Figure 5.2: An illustration of the quantum annealing process.

5.2 As a Solution to the Least Squares Problem

As solving the least squares problem is already an optimization problem, it can be directly translated to a Quantum Annealing problem if it can be posed as a **binary** optimization problem.

The most interesting part of trying to solve a problem via Quantum Annealing is how to cast it as a least squares problem (which involves adding conditions, finding appropriate Lagrange multipliers, etc) as it depends very specifically on the problem to solve. An extensive list of examples can be found at [20] and the only common step is the need to figure out how to encode the variables of interest as binary bits $q_i \in \{0, 1\}$.

$$\begin{aligned} |x_1\rangle &\rightarrow |01010...0\rangle \\ |x_2\rangle &\rightarrow |11101...1\rangle \\ &\vdots \\ |x_n\rangle &\rightarrow |q_0, q_1, q_2, ..., q_n\rangle \end{aligned} \tag{5.6}$$

There are many encoding schemes and choosing the best one is usually dependent on the problem as well, but for the unfolding problem this is already given in [4]. The only thing left is to plug it on a quantum computer to let Nature find the state with the lowest energy.

5.3 As a Solution to the Unfolding Problem

Using Quantum Annealing to solve the unfolding problem translates to mapping the general solution with Tikhonov regularization of the inverse problem (3.8) (which is written below for reference) to a binary optimization problem to be plugged on a quantum computer.

$$\min_{\mu} \left\{ \|R\boldsymbol{\mu} - \boldsymbol{\nu}\|_2^2 + \lambda \|D^2\boldsymbol{\mu}\|_2^2 \right\}$$

The key to make this translation (as explained with more detailed in [4]) is to express the unfolded histogram as a proportional variation of the histogram used to build the response matrix $\boldsymbol{\mu}^{\text{ini}}$. These variations are encoded in scaling parameters α, β (both function of a user argument called `range`) such that the resulting histogram can be expressed in terms of n bits $\{q_1, ..., q_n\}$ that will turn on/off the variations when the values are 1/0, respectively:

$$x_i = \alpha_i + \beta_i \sum_{j=0}^{n-1} 2^j q_{n \times i + j} \tag{5.7}$$

Where:

- $\boldsymbol{\alpha_i}$: is an offset that represents the value around which the proportional variations are to be found. The default configuration [21] is 50% of the corresponding μ_i^{ini} and is given by:

$$\alpha_i = (1 - \text{scale})\mu_i^{\text{ini}} \tag{5.8}$$

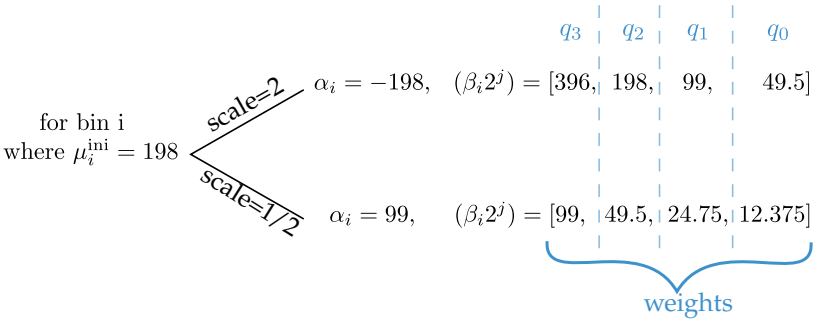


Figure 5.3: Examples to show the encoding schema of the quantum annealing.

- $\boldsymbol{\beta_i}$: is the value that will define the weight $(\beta_i 2^j)$ of each bit q when doubled for every j (that is, when being multiple of 2^j) and is given by:

$$\beta_i = \frac{2 \times \text{scale}}{2^n} \mu_i^{\text{ini}} \tag{5.9}$$

To illustrate better the encoding schema, an example of an encoding with 4 bits of a bin i is shown in figure 5.3 for two different values of `scale`.

The scaling parameters are important as depending on how similar we expect the unfolded histogram to be with respect to $\boldsymbol{\mu}^{\text{ini}}$, we might need to set lower or higher offsets and allow more or less variations with respect to it. The schema overviewed here is proposed by the source code of [4] (a Github repository can be found in the paper), though nothing prevents us from defining new encoders.

Important Remarks

In contrast with the SVD unfolding, with Quantum Annealing there is no simple way to find the covariance matrix of the resulting histograms, which makes error prediction not doable.

For testing, this algorithm requires access to a real quantum computer which is not possible by the time of writing this work [9] and that quantum computer would need to have enough qubits to encode the histograms to unfold, which is not the case of the current state-of-art devices [10] .

The approach to follow in this work is to use a Simulated Quantum Annealing (SQA) [22] implementation provided by [23], which is a classical algorithm that makes use of a Monte Carlo approach to mimic the quantum fluctuations in the optimization process. More details on appendix A.

To mimic what would be the variance to expect when measuring the quantum state in a real quantum computer, the simulations are performed several times and the mean value is presented as a result. Note that this makes the variance presented on the annealing results uncomparable with the variance computed by the SVD. More on chapter 8.

9: D-Wave offers access to its quantum computers, but not in Mexico. Amazon Web Services can serve as a proxy, but it is a paid service.

10: Workarounds are possible when having access to D-Wave's devices as it offers hybrid approaches.

Computational Experiment

The $\tau \longrightarrow \pi\pi^0\nu$ Decay

6

6.1 About

As mentioned in chapter one, in real-life conditions it is virtually impossible to repeat a collision and get always the same result. This is in part because statistical fluctuations, but also because the physics of the collisions are governed by the probabilistic nature of Quantum Mechanics.

Even if we make two particles collide several times under the exact same conditions (this would require to accelerate the particles to always the same energy and momentum before colliding), there are many possible outcomes each occurring with a different probability, all given by the Standard Model of Particle Physics and described at [24].

For example, if Belle II would like to produce the tau pair decay from the annihilation of an electron and a positron in the energy range around $\sqrt{s} = 10.58$ [GeV][1] , the observed events would include not only the desired $\tau^+\tau^-$ decay, but many other possibilities as well (this is depicted in figure 6.1).

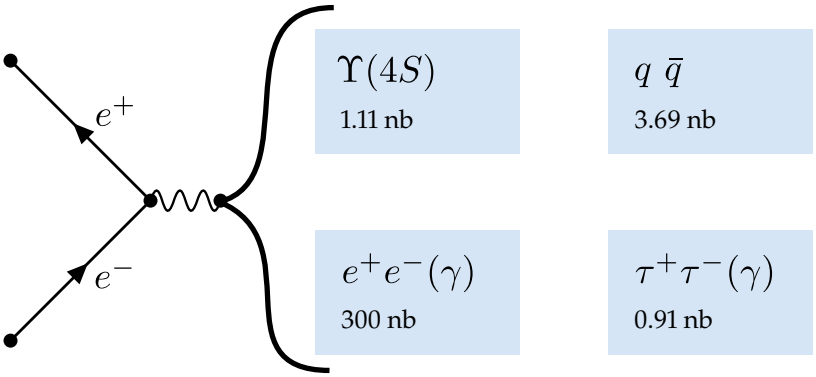


Figure 6.1: Some physics processes that are possible at $\sqrt{s} = 10.58$ [GeV] with their cross-sections. For a more comprehensive list, see [5], table 18.

In this work, the arbitrarily chosen decay to study is the $\tau \longrightarrow \pi\pi^o\nu$ (its data is available via the Belle II Collaboration) where the invariant mass of the $\pi\pi^0$ will be used as "signal". In order to showcase the unfolding techniques as straightforward as possible, data was taken from simulated experiments and by the way they are run, their data is available in such a way that we do not need to care about the background events coming from the e^-e^+ as we can already consider only tau decays, but these will still be polluted by every other possible decay of the τ which need to be filtered out by some pre-processing that we will call **data selection**.

In this case, the events generated by the simulator will end up being one of the possible decays of table 6.1 which we will call **decay modes** and are identified by an integer. The decay of interest corresponds to decay mode 4 in which the τ decays into a rho meson $\rho(770)$[2] which then decays to $\pi\pi^0$ as is illustrated in figure 6.2. This rho meson serves

1: s is one of the so called "Mandelstam variables" representing the invariant mass of a decay, which in the case of e^-e^+ is the total energy squared.

2: 770 is the mass of the particle in MeV. For convenience, this particle will be called only ρ for the rest of this document.

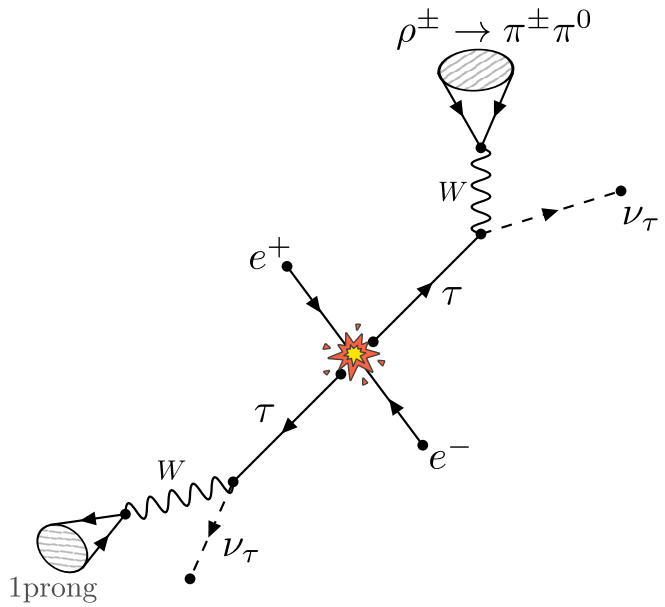


Figure 6.2: Quasi Feynman Diagram of the $\tau \to \pi\pi^0\nu$ decay.

as an intermediate particle for the process of interest and these particles are called **resonances**.

It is worth noting on table 6.1 that there are some other decays that look similar to the decay we are looking for. For example, the decay 21 where the τ produces the $\pi\pi^0\nu$ but with a companion photon which, if not detected, could lead to confusing that decay with the one we are looking for. A more problematic (as is more probable) decay is the mode 5 which will result in a very similar process as the a_1 can decay to $\rho\pi^0$ [24], so we will end up observing $\tau \to \pi\pi^0\pi^0\nu$ which can be mistaked as our decay if one π^0 is not detected.

1	$e^-\nu\nu$	15	$K^0\pi^-\bar{K^0}\nu$	29	$K^-\pi^+\pi^-\pi^0\nu$
2	$\mu^-\nu\nu$	16	$K^-K^0\pi^0\nu$	30	$K^-\pi^0\pi^0\pi^0\nu$
3	$\pi^-\nu$	17	$K^-\pi^0\pi^0\nu$	31	$K^0\pi^-\pi^+\pi^-\nu$
4	$\rho^-\nu$	18	$K^-\pi^-\pi^+\nu$	32	$\pi^-\bar{K^0}\pi^0\pi^0\nu$
5	$a_1^-\nu$	19	$\pi^-\bar{K^0}\pi^0\nu$	33	$\pi^-K^+K^-\pi^0\nu$
6	$K^-\nu$	20	$\eta\pi^-\pi^0\nu$	34	$\pi^-K^0\bar{K^0}\pi^0\nu$
7	$K^{*-}\nu$	21	$\pi^-\pi^0\gamma\nu$	35	$\pi^-\omega\pi^+\pi^-\nu$
8	$\pi^-\pi^-\pi^+\pi^0\nu$	22	$K^-K^0\nu$	36	$\pi^-\omega\pi^0\pi^0\nu$
9	$\pi^-\pi^0\pi^0\pi^0\nu$	23	$\pi^-4\pi^0\nu$	37	$e^-e^-e^+\nu\nu$
10	$2\pi^-\pi^+2\pi^0\nu$	24	$\pi^-\omega\pi^0\nu$	38	$f_1\pi^-\nu$
11	$3\pi^-2\pi^+\nu$	25	$\pi^-\pi^+\pi^-\eta\nu$	39	$K^-\omega\nu$
12	$3\pi^-2\pi^+\pi^0\nu$	26	$\pi^-\pi^0\pi^0\eta\nu$	40	$K^-K^0\pi^+\pi^-\nu$
13	$2\pi^-\pi^+3\pi^0\nu$	27	$K^-\eta\nu$	41	$K^-K^0\pi^0\pi^0\nu$
14	$K^-\pi^-K^+\nu$	28	$K^{*-}\eta\nu$	42	$\pi^-K^+\bar{K^0}\pi^-\nu$

Table 6.1: Decay modes of the τ^- lepton. Decay modes of the τ^+ are the same after applying charge conjugation.

More on this will be discussed when explaining how events are filtered and reconstructed.

6.2 Data Selection

For the purposes of this work (performing the unfolding), data was gathered from the simulation campaigns carried out by the Data Production

group of the Belle II experiment. For short, the collection of data will be called **taupair data set** and is identified by the following specs:

```
Campaign: MC13a
Beam Energies: 4S
Release: release-04-00-03
MC Event Types: taupair
```

These simulations are carried out by generators that have not been developed within Belle II but have been inherited from the HEP community and the simulated effects of the detector response include simulations of the triggers [3] , of the interaction of the MC particles with matter and calorimeters. More details can be found on section 4 of [5].

3: "Triggers" are sensors placed inside the Belle II detector that notify the rest of the measurement devices when an important physics process is observed so they can turn on in time to perform their respective measurements.

As every τ lepton can decay in multiple events (see table 6.2 for some examples) and simulation campaigns are not specific enough to produce events only of a certain type, some pre-processing is needed to filter out most of the background events as possible. Two types of pre-processing were applied:

1. It is possible to remove events where some measurements of physical observables (such as energy of a particle, momentum, etc) do not match the physical conditions (usually defined as thresholds) expected from the signal. Filtering out events by conditions is known as "applying cuts".
2. Even after applying cuts to the data, there is the possibility that some conditions that escape the human intuition can still be identified to distinguish each type of event, so a machine learning classifier called **Boosted Decision Trees** (BDT) was used to classify events as signal and background.

Table 6.2: Some possible decays of the τ^-.

Decay Mode $\tau^- \to$	Fraction Γ_i/Γ
$e^-\bar{\nu}_e\nu_\tau$	17.83 ± .04%
$\mu^-\bar{\nu}_\mu\nu_\tau$	17.41 ± .04%
$\pi^-\nu_\tau$	10.83 ± .06%
$\pi^-\pi_0\nu_\tau$	25.52 ± .09%
...	...

Machine learning techniques like BDTs usually require splitting the data set into two groups to assess the capacity of the model to generalize and to prevent it from overfitting: one to train the classifier called "training set", and one to evaluate the model called "validation set".

In a similar fashion, the rest of the data set was split into two more groups. One to mock the simulation which is used to prepare the unfolding (identify the response matrix) and one to mock the real data to which the unfolding techniques will be applied.

In summary, the whole taupair data set is split into three groups after the cuts are applied:

- One group to train the BDT. This group corresponds to the 30% of the taupair data set. This one will be called **BDT set**.
- One group to prepare the unfolding. This group corresponds to the 30% as well and will be called **preparing set**.
- One group to apply the unfolding to. This group corresponds to the remaining 40% and will be called **unfolding set**.

First Stage: With Cuts

In this section, the analysis on how the decay of interest is reconstructed by cutting the data with **basf2** will be presented. As this stage is necessary to set up the unfolding problem but not the main content of this work,

only the important pieces of the code will be explained along with the rationale behind, but the full code can be found on appendix B.

As in a τ decay information is lost because neutrinos (ν) cannot be detected, the customary way to perform the reconstruction is to split the event into two sides using a plane defined by the **thrust axis** (as illustrated in figure 6.3), that are then used to reconstruct their mother particles by going backwards in the physics process.

The thrust axis $\hat{n}_{\text{thrust}}$ is defined as the axis such that the thrust V_{thrust} defined in equation (6.1) is maximum and it can be understood as the direction (in the Center of Mass frame) in which most of the momentum is concentrated.

$$V_{\text{thrust}} = \frac{\sum_i |\boldsymbol{p}_i^{\text{CM}} \cdot \hat{n}_{\text{thrust}}|}{\sum_i |\boldsymbol{p}_i^{\text{CM}}|} \tag{6.1}$$

After splitting the event, some particles will end up in one side and some in the other side. The side with the decay of interest will be called **signal** and other side will be called **tag**.

After loading all the data into a basf2 pipeline called "main", particle candidates need to be loaded. As charged particles and photons are the easiest to identify directly by the measurements of the Electromagnetic Calorimeters (ECL) clusters [4] , it is convenient to reconstruct them first.

Particle identification is done automatically by basf2 which translates signals and quantities measured by the detectors into the most probable particles and each one of them is stored in a given list of variables. The naming of these lists (which will be used throughout the section) is given by two parts separated by a colon: **particle:label**. The left-most part corresponds to the type of particle and the right-most part to the name with which the list is identified.

The particles reconstructed at this stage are:

- **e-:all** stores all the electron candidates
- **mu-:all** stores all the muon candidates
- **pi+:all** stores all the charged pion candidates
- **gamma:all** stores all the photon candidates

In basf2, these lists contain both particles and anti-particles and, up to this point, the only difference between these lists is the mass hypothesis used in the track fit [5] , so we still need to correctly identify muon candidates as muons, electron candidates as electrons, etc.

```
ma.fillParticleList("e-:all", "", path=main)
ma.fillParticleList("mu-:all", "", path=main)
ma.fillParticleList("pi+:all", "", path=main)
ma.fillParticleList("gamma:all", "", path=main)
```

After filling these lists, each particle candidate will have a property called **particleID** (where "particle" can be one of the possible charged particles available for direct reconstruction [6]) containing the probability of that candidate being identified as that specific particle, which we will call **particle identification probability**.

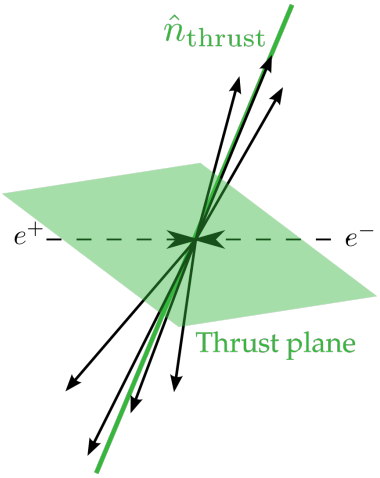


Figure 6.3: Illustration of the thrust axis of an electron-positron collision.

4: Their momenta can be inferred by the tracks they leave behind and their energy can be measured by ECLs, which is enough to identify the particle as the dynamics are known.

5: Track fitting is the process of determining the best estimate of the kinematic variables describing the particle trajectories corresponding to each of the patterns found.

Listing 6.1: Code snippet to reconstruct charged particles where **ma** stands for "Modular Analysis".

6: For example, "electronID" will contain the probability of that particle being an electron.

The first cut to be applied is for the list of photons as they will be used to reconstruct neutral pions via the $\pi^0 \to \gamma\gamma$ decay, which occurs most of the time [7].

7: The branching ratio is of almost 98.8% [24].

This cut will restrict photon candidates **gamma:all** to:

- Those that fall within the angular acceptance of the detector $-0.866 < \cos(\theta) < 0.9563$ (where θ corresponds to the polar angle which should be in the range $17° < \theta < 150°$ [8]).
- Those that satisfy the typical minimum energy threshold $E > 100$ [MeV] used to reconstruct "good photons" (photons that do not come from the background) in the electromagnetic calorimeter placed in the barrel region [9].
- Those that are produced by a neutral pion with a low-enough energy so the angular separation between the photons could make them overlap a bit when hitting the ECL clusters (that is, the ECL clusters would not necessarily detect two hits as the two photons could have been detected by almost the same pixel) [10].

8: See [6] section 5.1.2.

9: See [6] section 7.3.4.2.

10: See [5] section 5.6.1.

```
ma.cutAndCopyLists(
    "gamma:looseForPi0", "gamma:all",
    "E > 0.1 and -0.8660 < cosTheta < 0.9563
        and clusterNHits > 1.5",
    path=main
)
```

Listing 6.2: Code snippet with the cut used to find photons coming from a π^0.

These photons will then be used to reconstruct the neutral pion, considering a nominal mass around 134.97 [MeV][24], so the mass will be constrained within a window of ±8 [MeV]. The amount of reconstructed π^0 and the amount of γ used in the reconstruction by these cuts are stored in variables called **nPi0** and **nPhotonsFromPi0**, respectively.

```
ma.reconstructDecay(
    "pi0:fromLooseGammas -> gamma:looseForPi0 gamma:looseForPi0",
    "0.115 < M < 0.152",
    path=main,
)
ma.cutAndCopyLists(
    "gamma:pi0",
    "gamma:looseForPi0",
    "isDescendantOfList(pi0:fromLooseGammas) == 1",
    path=main,
)
variables.addAlias("nPhotonsFromPi0", "countInList(gamma:pi0)")
variables.addAlias("nPi0", "countInList(pi0:fromLooseGammas)")
```

Listing 6.3: Code snippet with the π^0 reconstruction.

To reconstruct the charged pions from the tracks that were loaded (that is, to make sure we remove tracks from `pi:all` that could be electrons or muons mistakenly labeled as pions), we can use the suggested cut [11] $E/P < 0.8$ which can accurately separate electrons from muons and pions (but not muons from pions) based on the energy deposited in the calorimeters E and the momentum of the particle P. To discriminate muons, we can keep the tracks from which the reconstruction yields a pion only with a probability greater than 50%.

11: See [5] section 5.5.4.

Additionally, only tracks that originate from a region very close to the primary vertex (a sensible cut is $-3.0 < dz < 3.0, dr < 1.0$) [12] will be considered as this allows to filter charged particles originating from a τ

12: The "primary vertex" is the point in space where the electron and the positron came into contact with each other. The variable dr gives the transverse distance, while dz is the z-component of the point of closest approach with respect to the interaction point. The cut is based on [6] section 12.1.

decay from tracks that come from other particles with a longer lifetime. This is not strictly necessary in this case as we are already using the taupair data set, but is added for completeness.

These correctly reconstructed pions will be stored in a list called **pi:pidsig**.

In a similar way, cuts on the tracking variables and on particle identification probabilities are applied to reconstruct electrons and muons and they are stored in lists called **e:pid** and **mu:pid**, respectively. These are not part of the decay of interest, but are used to count the amount of tracks that were present in the event, both in the signal side and in the tag side, as we will restrict the search to decays with one track on each side, which is a topology called **1x1 prong**.

Note that, in principle, we are only interested in the tau decay on the signal side, but we need to add cuts to the tag side in order to reconstruct a tau process. It could be any process, but as adding cuts inevitably filters out some of them, looking for one track allows us to retain as much events as possible as the tau decays to one charged particle around 85% of the time [24].

```
# Reconstruct electrons
ma.cutAndCopyLists("e-:pid", "e-:all",
    "electronID > 0.5 and -3.0 < dz < 3.0 and dr < 1.0",
    path=main)

# Reconstruct muons
ma.cutAndCopyLists("mu-:pid", "mu-:all",
    "muonID > 0.5 and -3.0 < dz < 3.0 and dr < 1.0",
    path=main)

# Reconstruct pions for the 1prong tag
ma.cutAndCopyLists("pi+:pid", "pi+:all",
    "pionID > 0.5 and -3.0 < dz < 3.0 and dr < 1.0",
    path=main)

# Reconstruct pions for the signal (tau->pi+ pi0 nu)
variables.addAlias("EoverP",
    "formula( ifNANgiveX( clusterE, -1 )/p )")
ma.cutAndCopyLists("pi+:pidsig", "pi+:all",
    "pionID > 0.5 and -3.0 < dz < 3.0 and dr < 1.0
    and EoverP < 0.8 ",
    path=main)

# Count the amount of tracks reconstructed
variables.addAlias("nGoodTracks", "formula(
    countInList(pi+:pid) +
    countInList(e+:pid)  + countInList(mu+:pid))")
```

Listing 6.4: Code snippet with the tracks reconstruction.

Up to this point, we can already apply two useful cuts to filter out a lot of events that are not part of the decay we want to study:

- There must be one track on the signal side corresponding to the pion, and we can limit ourselves to events with one more track on the tag side.
- We can already remove events with no neutral pions.

```
ma.applyEventCuts("nGoodTracks == 2", path=main)
ma.applyEventCuts("nPi0 > 0", path=main)
```

Listing 6.5: Code snippet with the 1prong cut with at least one π^0.

The amount of photons not coming from a π^0 are also counted in a similar fashion as with the "gamma:looseForPi0" and the amount is stored in nGoodPhotons.

```
ma.cutAndCopyLists("gamma:notPi0", "gamma:all", "
    E > 0.2 and clusterNHits > 1.5
    and -0.8660 < cosTheta < 0.9563
    and isDescendantOfList(pi0:fromLooseGammas) == 0
", path=main)

variables.addAlias("nGoodPhotons", "countInList(gamma:notPi0)")
```

Listing 6.6: Code snippet with the photons filtering.

We can compute the thrust found on each side using basf2 and we can now proceed to reconstruct the taus as we have already reconstructed all of its children. Taus from signal side will be stored in:

```
tau+:sig -> pi+:pidsig pi0:fromLooseGammas
```

While taus of the tag side will be stored in

```
tau-:1prong
```

as a combination of all the taus that resulted in a track on the other side. In here, it must be noted that neutrinos are not explicitly required in the reconstruction as they are invisible to the detector.

```
# Signal side
ma.reconstructDecay(
    "tau+:sig -> pi+:pidsig pi0:fromLooseGammas", "", path=main)

# Tag side (1 prong)
ma.reconstructDecay("tau-:e  -> e-:pid",  "", path=main, dmID=11)
ma.reconstructDecay("tau-:pi -> pi-:pid", "", path=main, dmID=211)
ma.reconstructDecay("tau-:mu -> mu-:pid", "", path=main, dmID=13)

ma.copyLists("tau-:1prong", [
    "tau-:e", "tau-:pi", "tau-:mu",
], path=main)
```

Listing 6.7: Code snippet with τ reconstruction.

These two taus all come from a virtual photon produced by the annihilation of the electron and the positron, which will be reconstructed as well.

```
ma.reconstructDecay("vpho -> tau+:sig tau-:1prong", "", path=main)
```

For the taus, a quality cut can be applied to check that both of them are on different sides of the thrust axis. The variable **track_1_x_track_2** will contain the product of the cosines of the angle between the thrust axis and the track on the tag side and on the signal side. The product will be negative when both particles are on different sides of the thrust plane. This is illustrated on figure 6.4.

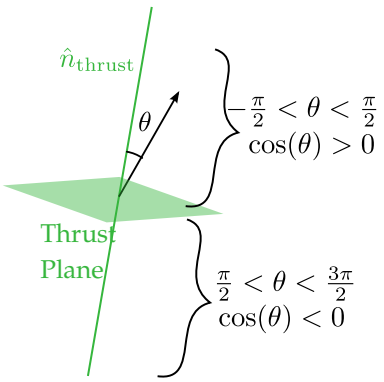


Figure 6.4: Illustration of how the thrust angle and its cosine is measured.

```
variables.addAlias(
    "track_1_x_track_2",
    "formula(
        daughter(0, daughter(0, cosToThrustOfEvent))
        * daughter(1, daughter(0,cosToThrustOfEvent))
    )",
)

ma.applyCuts("vpho", "track_1_x_track_2 < 0", path=main)
```

Listing 6.8: Code snippet with the thrust quality cuts applied to the virtual photon.

Similarly, for each particle we reconstructed (the photons, the pions, etc), we know on what side of the decay they are expected, so we can apply a similar cut to each of them.

Listing 6.9: Code snippet with the thrust quality cuts applied to all the particles.

```
# Create copies of the list of photons not coming from the pi0 to
# operate
ma.copyList("gamma:1prong", "gamma:notPi0", path=main)
ma.copyList("gamma:1prongPi0", "gamma:notPi0", path=main)
ma.copyList("pi0:1prong", "pi0:fromLooseGammas", path=main)
ma.copyList("pi0:1prongPi0", "pi0:fromLooseGammas", path=main)

# Identify particles on the positive side of the thrust
positiveThrust = b2.create_path()
ma.applyCuts("gamma:1prong",    "cosToThrustOfEvent > 0",
    path=positiveThrust)
ma.applyCuts("gamma:1prongPi0", "cosToThrustOfEvent < 0",
    path=positiveThrust)
ma.applyCuts("pi0:1prong",      "cosToThrustOfEvent > 0",
    path=positiveThrust)
ma.applyCuts("pi0:1prongPi0",   "cosToThrustOfEvent < 0",
    path=positiveThrust)

# Identify particles on the negative side of the thrust
negativeThrust = b2.create_path()
ma.applyCuts("gamma:1prong",    "cosToThrustOfEvent < 0",
    path=negativeThrust)
ma.applyCuts("gamma:1prongPi0", "cosToThrustOfEvent > 0",
    path=negativeThrust)
ma.applyCuts("pi0:1prong",      "cosToThrustOfEvent < 0",
    path=negativeThrust)
ma.applyCuts("pi0:1prongPi0",   "cosToThrustOfEvent > 0",
    path=negativeThrust)
```

Finally, for every event that survived the cuts, several variables of every reconstructed particle are stored to be used in the next stages of the experiment. Important variables are:

Event Variables

1. `thrust`: the thrust as calculated with equation (6.1).
2. `visibleEnergyOfEventCMS`: the visible energy in the Center of Mass frame.
3. `tauPlusMCMode`: the decay mode (labeled by the PDG) of the τ^+ as originated by the Monte Carlo generator. This variable is only available on simulations.
4. `tauMinusMCMode`: the decay mode (labeled by the PDG) of the τ^- as originated by the Monte Carlo generator. This variable is only available on simulations.
5. `track_sig_charge`: the charge of the track on the signal side.
6. `tau_sig_InvM`: the invariant mass of the τ on the signal side. [13]
7. `nPi0s_sig`: the amount of π^0 reconstructed on the signal side.

13: As we are not reconstructing resonances (nor neutrinos) directly, the invariant mass of the τ will be equal to the invariant mass of the $\pi\pi^0$, which is the variable of interest for the unfolding.

Variables Stored For All Particles

1. Kinematic Variables (both "measured" and MC in the lab and Center of Mass frames)
   a) $p^\mu = (E, px, py, pz)$: four momentum.
   b) `p`: momentum magnitude.

c) `M`: mass.
d) `InvM`: invariant mass.
e) `pt`: transverse momentum.

2. Thrust Variables
   a) `cosToThrustOfEvent`: cosine of the angle between the particle's trajectory and the thrust axis.
3. Simulation Variables
   a) `mcPDG`: ID (as labeled by the PDG) of the particle generated by the simulator. This variable is only available on simulations.

Variables Stored Only for Charged Particles

1. `charge`: charge of the reconstructed particle.
2. `EoverP`: the Energy (E) divided by the magnitude of the Momentum (P).
3. Probabilities that the track matches a given particle.
   a) `kaonID`
   b) `pionID`
   c) `protonID`
   d) `muonID`
   e) `electronID`
   f) `deuteronID`

Variables Stored Only for the Photons

1. `clusterE`: energy measured by the cluster.

As mentioned in section 6.2, all the resulting events can still have entries that represent background noise instead of signal but that escaped from the cuts. So, before applying the unfolding techniques presented in this work, a subset (called "BDT set") is taken apart to train a BDT so the unfolding is performed on a data set as clean as possible.

Note that in a real-life scenario, a more thorough study could be performed to choose more strict cuts[14] and perform a "better" reconstruction, but (as mentioned in chapter 1) as the decay was chosen arbitrarily and is not the main focus of this work, the cuts presented in here were considered sufficient to set up the unfolding and to move on to the BDT stage.

14: For example, choosing better cuts for the tag, choosing better thresholds for the `particleID` probabilities, etc.

Second Stage: With Boosted Decision Trees

After reconstructing particles and events by cutting data with basf2, we can assess how good the cuts performed by looking at the histograms of the reconstructed invariant mass (as the detector would have measured it) and the true invariant mass (as generated by the Monte Carlo simulations) of the $\pi\pi^0$ for events with one reconstructed π^0, labeled as `measured` and `matchedMC` [15] , respectively, in figure 6.5.

15: These labels are used with the same meaning throughout the rest of the document.

We can notice the following things:

- Both histograms correctly contain the $\rho(770)$ resonance in the 0.770 [GeV] region.

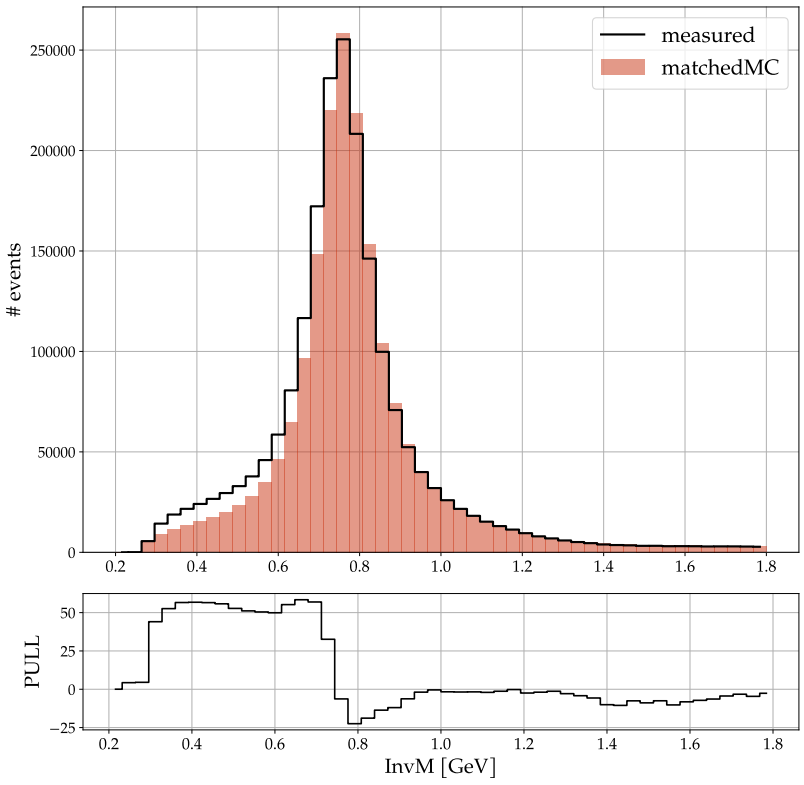


Figure 6.5: Histograms of the reconstructed invariant mass (measured) and the true invariant mass (matchedMC) of the $\pi\pi^0$.

- The measured and reconstructed invariant masses differ mostly in the region below the 1.0 [GeV] as the smearing caused by the detector can be observed (and needs to be unfolded).

To explore possible causes, one approach is to make use of the simulation variables **tauDecayMode** to identify what kind of decays make up the background that we need to deal with.

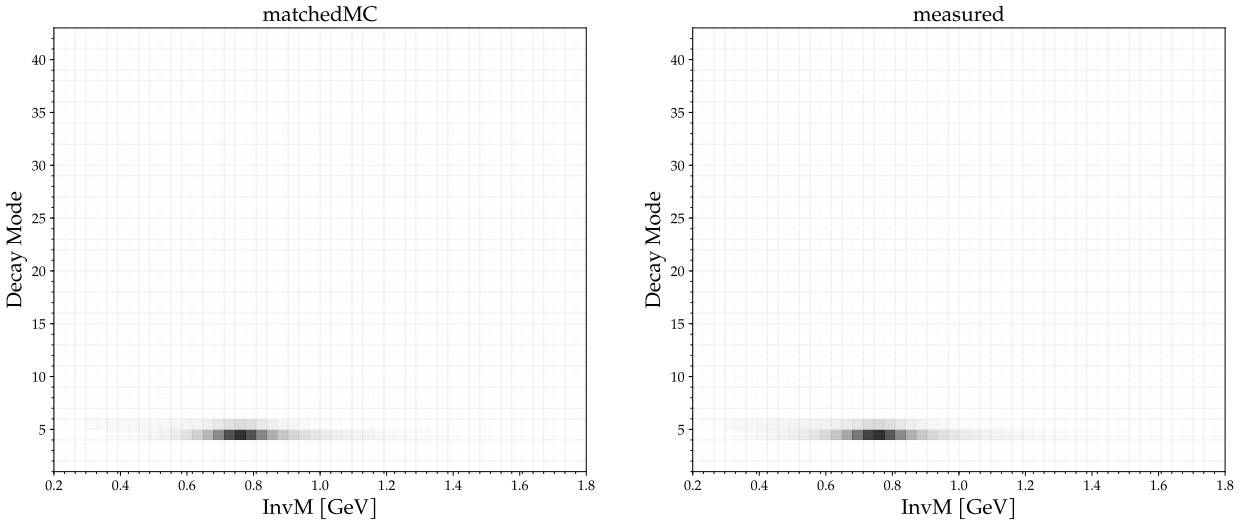


Figure 6.6: Histograms of the reconstructed invariant mass (measured) and the true invariant mass (matchedMC).

On the vertical axis we can see the decay modes as labeled in table 6.1 and by checking their position on the horizontal axis, we can find where those decays ended up, from which we can conclude that the

background events present in the invariant mass histograms come from the a_1 resonance (`decayMode=5`).

After checking the decay modes of the a_1 [24], we can see that most of its decays are to three pions like $\rho\pi$, $\pi^+\pi^-\pi^0$, $\pi^0\pi^0\pi^0$, etc, which are very similar to the decay of interest in this work where the ρ decays to $\pi\pi^0$. Particularly, the most resembling and problematic decay is:

$$a_1^{\pm} \to [\rho^{\pm} \to \pi^{\pm}\pi^0]\pi^0 \tag{6.2}$$

The conclusion is that this decay is too similar to the case where the a_1 is not present because if one π^0 is either not detected or not reconstructed correctly, the decays ($\tau \to a_1\nu$) and ($\tau \to \rho\nu$) become indistinguishable. This suggests that applying cuts based on physics principles is not enough and an additional and slightly more sophisticated cleaning stage needs to be performed.

The chosen approach was to use a Gradient Boosted Decision Tree (GBDT) to solve a classification problem where each event can belong to one of two categories: signal and background.

The category can be easily identified for each event by using the Event Variables to check which of the following conditions is satisfied:

- **Signal:** if the τ corresponding to the charge of the track on the signal side decayed to a ρ.

```
( (track_sig_charge ==  1) and (tauPlusMCMode  == 4) )
    or
( (track_sig_charge == -1) and (tauMinusMCMode == 4) )
```

- **Background:** if the τ corresponding to the charge of the track on the signal side decayed to something different than a ρ.

```
( (track_sig_charge ==  1) and (tauPlusMCMode  != 4) )
    or
( (track_sig_charge == -1) and (tauMinusMCMode != 4) )
```

  Note that this is equivalent to just using the negation of the previous condition.

Let us not forget that difficulty arises when trying to classify the events without access to the Event Variables corresponding to the Monte Carlo simulations (in this case, `tauPlusMCMode` and `tauMinusMCMode`) as there is no way of knowing with full certainty what the decay mode was, which is why the GBDT needs to be trained to predict these categories with other meaningful variables accessible on a real data set. The variables chosen to be good for classification based on physics principles are:

- `thrust` and `visibleEnergyOfEventCMS` because we are trying to discriminate between two different types of events, so the event variables are usually a good choice.
- `track_sig_EoverP` because, as mentioned in section 3, the quantity E/P is a good discriminator for charged particles. It works best for electrons, but it was found to be a good parameter for the model.
- `track_sig_pionID` because the probability of the charged particle being a pion is related to the decay mode.

16: The component of momentum transverse (i.e. perpendicular) to the beam line.

- `track_sig_pt` because the transverse momentum [16] is usually different depending on the amount of particles product in the decay and how massive they are. i.e. one particle will have a higher transverse momentum than three.
- `sig_gamma1_E` and `sig_gamma2_E` because it could be possible that photons coming from the background are being incorrectly reconstructed as product of the π^0 decay.
- `track_sig_cosToThrustOfEvent` because it is related to the thrust.
- `neutral_sig_E` and `neutral_sig_M` to identify which π^0s come from a different decay mode.

As Machine Learning models' ability to generalize depends on being trained with samples not part of the data to which it is going to be applied, the **BDT set** (which corresponds to 30% of the taupair dataset) was split in two halves; one to be used for training the model (called the **training set**) and one to evaluate its performance (the **test set**).

Several models were tried with different parameters, and the one with the best score when applied to the test set was an ensemble of 100 Decision Trees with a max depth of 5 levels each and a restriction on the leaf nodes to contain no less than 5% of the data, which correctly classified events on the test set with an accuracy of 76.68% [17] .

17: More on Decision Trees can be found on appendix C.

```
GradientBoostingClassifier(n_estimators=100, max_depth=5,
min_samples_leaf=0.05)
```

The feature importances are shown in figure 6.7, from which we can conclude that the variables mostly used to perform the classification are the visible energy of the event, the energy of the reconstructed π^0, the thrust and the transverse momentum of the track on the signal side.

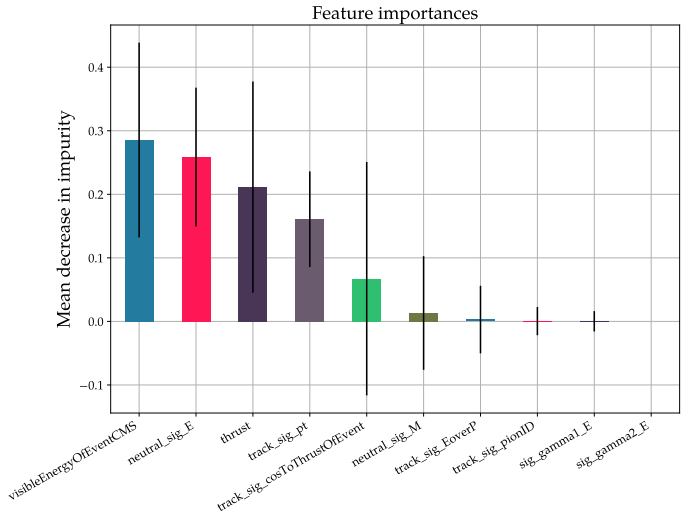


Figure 6.7: Feature importances of the GradientBoostingClassifier.

There are now two different approaches to make use of this model in our signal/background classification problem:

1. Choose the class with the highest output given by the **decision function** [18] . That is, the one with the highest odds or probability.
2. Set a clever threshold to define the minimum probability to assign a given class, say "signal".

18: The "decision function" is just a fancy name given to the output of the model. In this case, it can be given in terms of odds or probabilities.

The usual way is to go with option 1, which will be called **weak classifier**, and the benefits of going with option 2, which will be called **strict classifier**, will be explored in more detail in chapter 7. For now, the decision function (in terms of probabilities as it is more intuitive to explain) for the training set and the test set will be plotted on the left of figure 6.8, both for the signal and the background components in the training and test sets, along with an option for the clever threshold to give an idea of what to expect after using the GBDT.

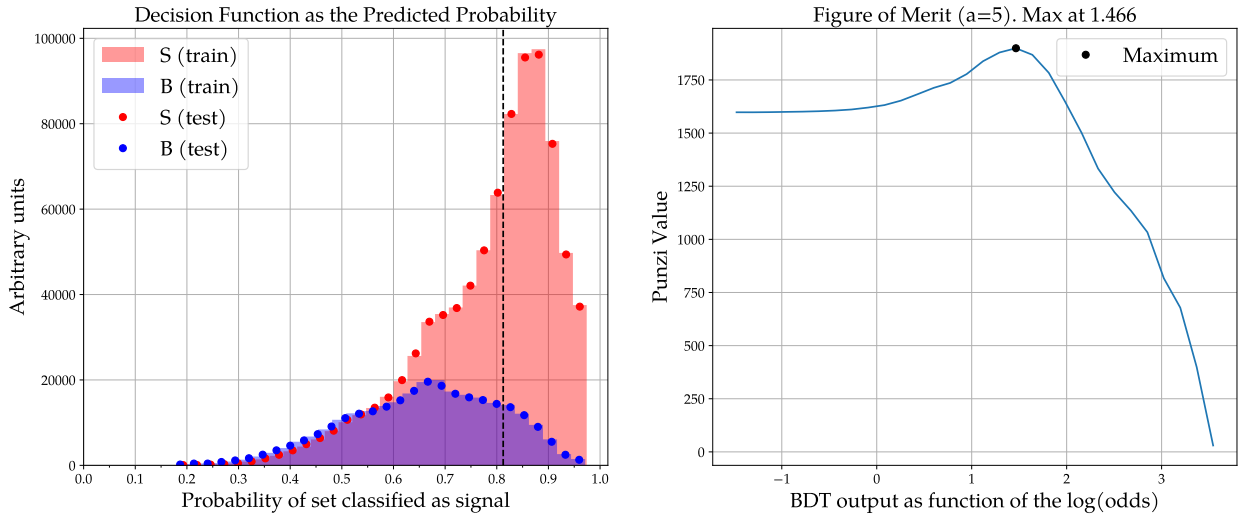


Figure 6.8: Left: Decision Function in terms of the probability with the Punzi threshold where $S(\cdot)$ stands for signal and $B(\cdot)$ stands for background. Right: Output of the BDT in terms of the log(odds) with the Punzi critical value.

The clever threshold is set according to the Punzi Criterion [25] with the following hypotheses:

Null Hypothesis (H_0) The data has no signal and only the background is present

Alternative Hypothesis (H_1) The $\tau \to \pi\pi^0\nu$ decay is present in the data

The significance level chosen to reject H_0 is of five standard deviations ($a = 5$) which consists of choosing only the signal candidates with an output of the decision value greater than the maximum of a so called **Merit Function** (plotted on the right of figure 6.8), given by the integrated efficiency ϵ of signal that passes the cuts and the integrated amount of background B:

$$\text{Punzi Value} = \frac{\epsilon}{a/2 + \sqrt{B}} \tag{6.3}$$

The ideal cut was found to be when log(odds)= 1.466, which, when using equation (C.2), translates to a minimum probability of $P = 81.24\%$ plotted as a vertical dashed line on the left of figure 6.8.

The results (to which the unfolding will then be applied) are shown on figure 6.9 with the signal and background isolated (both for the reconstructed and the Monte Carlo invariant masses) so an assessment on how well the classifiers are working can be done.

It is shown that the weak classifier still leaves a considerable amount of background on the resulting histograms (which can be seen on the left

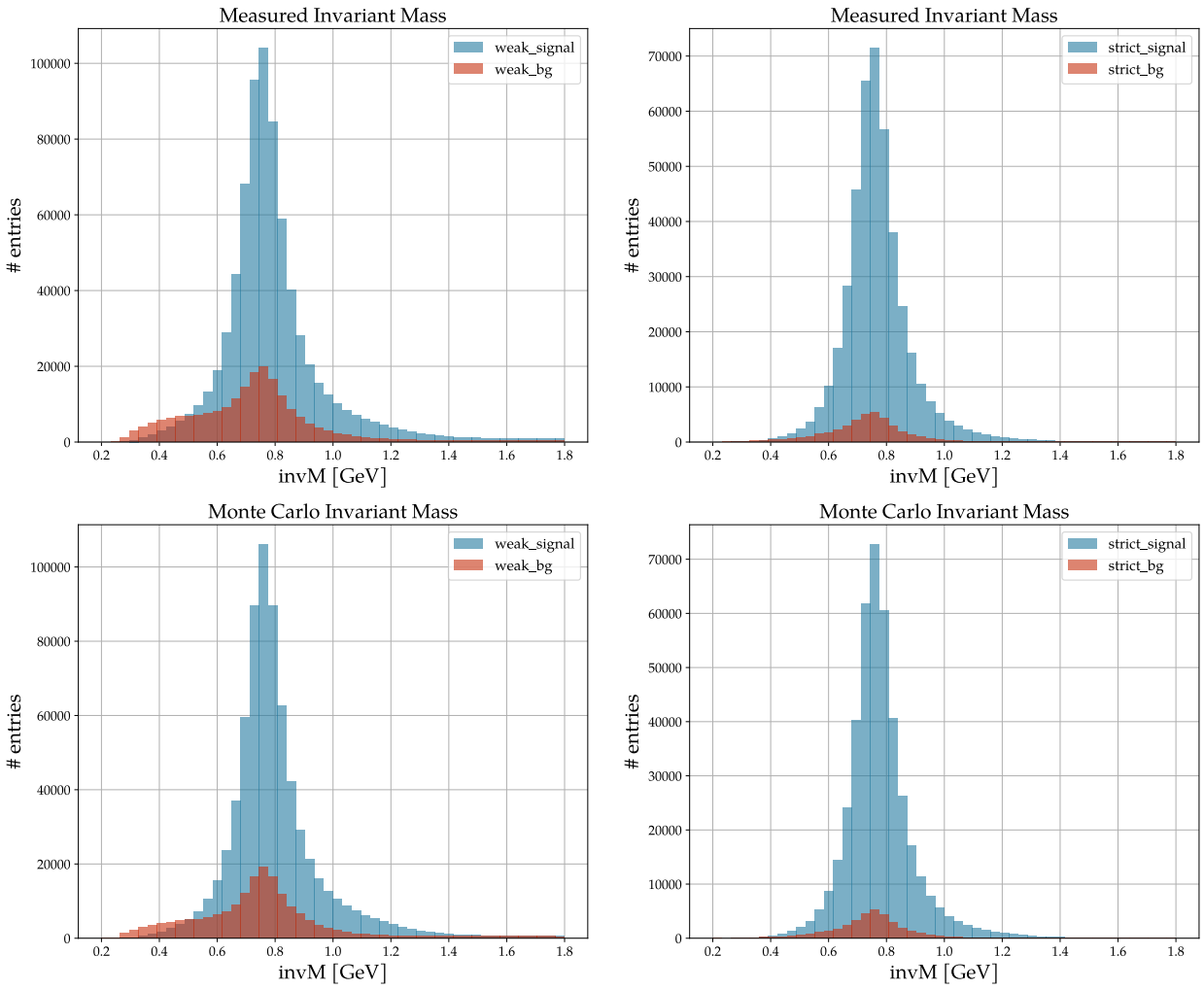


Figure 6.9: Top: reconstructed invariant mass for the output of the weak classifier (left) and the strict classifier (right). Bottom: Monte Carlo invariant mass for the output of the weak classifier (left) and the strict classifier (right).

side), while the strict classifier results in much cleaner histograms (on the right), though still not perfect and with less events. This is expected as no Machine Learning model will have an accuracy of 100% and because the strict classifier is setting a threshold lower than $P = 100\%$.

7 Unfolding

In this chapter, the main procedure to apply the unfolding techniques revisited on chapters 4 and 5 to the decay proposed on chapter 6 as found in the data set called "the unfolding set" in section 6.2 will be presented.

First, an explanation on how the response matrix was computed will be shown as it is the first step to test both of the unfolding techniques presented in this work. Then, the unfolding will be performed on the result of what was called the weak classifier to illustrate some issues that may occur with noisy data and some peculiarities of each technique. Finally, it will be performed on the result of the strict classifier to study how much these issues can be mitigated.

For most experiments, the error matrix is assumed diagonal, which means its elements can be interpreted as the variation expected to see when repeating the experiment several times and can be conveniently plotted as bars around each measurement, but this is not the case after performing the unfolding.

As mentioned in chapter 4, the SVD technique provides a way to compute the unfolded error matrix, which is not expected to be diagonal [1] , which means no easy interpretation can be given. Following the approach of [12], the elements in the main diagonal are plotted as bars in the results, but it is important to remark that it does not show the full information of the variations as the off-diagonal elements can not be plotted and a simple interpretation can not be given.

1: After all, undoing bin migrations introduces correlation between bins.

For the case of Quantum Annealing, we face the opposite case as there is no technique provided to compute the unfolded error matrix (as was mentioned in chapter 5). In [4], the unfolding was performed several times on a quantum computer and the standard deviation (per bin) was plotted as bars in the results, so a similar approach is considered here and the simulation is performed several times to compute the correlation matrix and the main diagonal is plotted in bars.

So, even though error bars are plotted for the results of both techniques, they are not comparable as they represent different things. Moreover, as neither of the resulting correlation matrices is expected to be diagonal, they are fully shown for all the results.

7.1 Computing the Response Matrix

As a brief reminder, the response matrix represents the effects and distortions (smearing, bin migration, etc) induced by the measurement device (in this case, the Belle II detector) when performing an observation.

The response matrix in terms of amount of events migrated is shown in figure 7.1 and was built using the histograms of figure 7.3. As mentioned

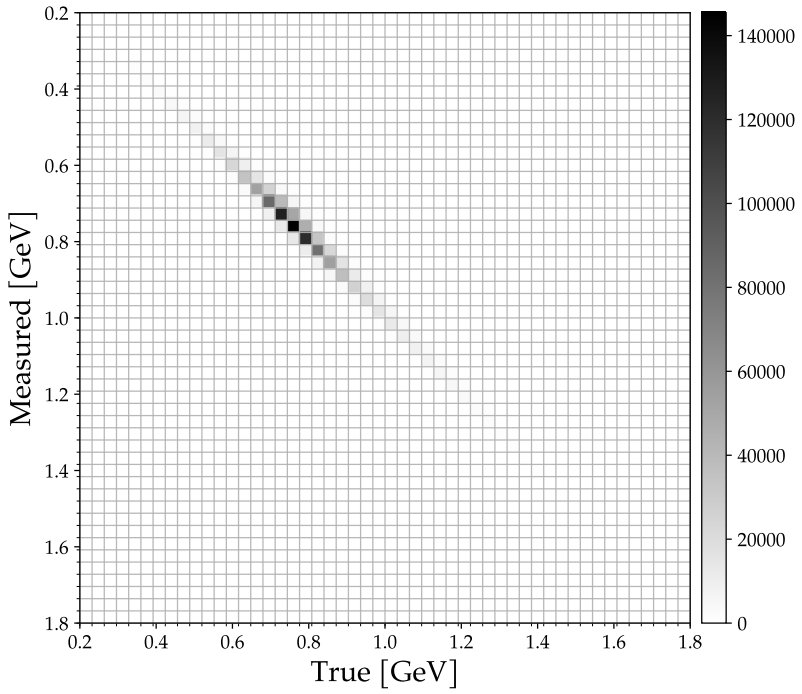


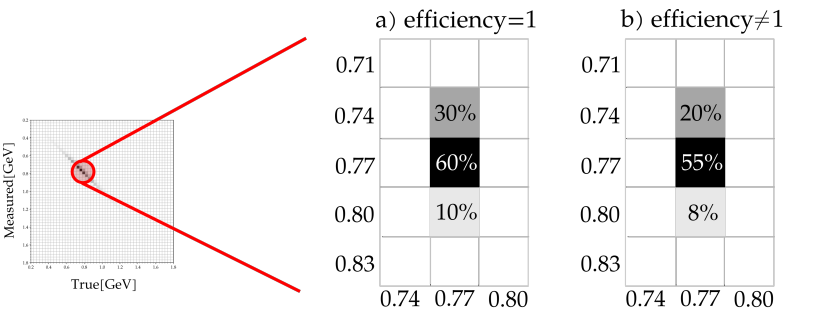
Figure 7.1: Response matrix of the $\tau \rightarrow \pi\pi^0\nu$ decay in terms of amount of events migrated.

in section 2.2, the response matrix can be understood as the conditional probability of finding an event in bin j when it should have been measured on bin i, but the matrix shown is in terms of the amount of events that migrated from one bin to another and not in terms of the probability of migration [2] . As the SVD unfolding technique requires using the response matrix in terms of events and Simulated Quantum Annealing (SQA) requires it in terms of probabilities, this normalization is performed when carrying out the unfolding and it represents no difference in here for the purpose of visualization as the only thing that would differ would be the scale.

2: The response matrix in terms of conditional probabilities can be obtained by normalizing the columns of the response matrix in terms of events considering the total amount of events generated at each energy level.

The events were taken from the **preparing set** and, as they were taken from a simulation campaign of the Belle II experiment, events already contain the effects of the detector, so after reconstructing the decay as explained in section 6.2, it is available the invariant mass (which is the variable of interest) of the τs both as generated by the simulator and as "measured" [3] by the simulated detector.

3: From now on, the reconstructed invariant mass with the effects of the detector will be called the "measured" invariant mass.



1000 events are generated with an invariant mass of 0.77 [GeV]

1000 events are distributed in the bins 0.74, 0.77 and 0.80 as effect of the detector

17% of the events are not found on any bin as they are missed by the detector

Figure 7.2: Illustration where 1000 events were generated with an invariant mass of 0.77 [GeV] and a) 100% of the events are measured so the efficiency is 100%, and b) only 83% are measured, so the efficiency is 83%.

Note that the response matrix in this case does not include the efficiency

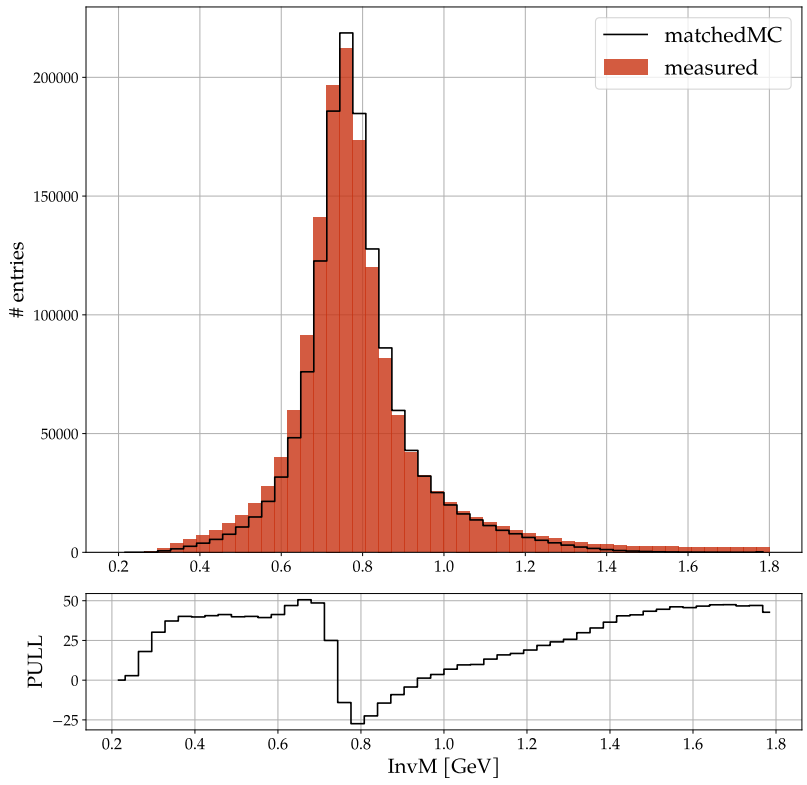


Figure 7.3: Histograms (and its pull plot) used to build the response matrix of the $\tau \rightarrow \pi\pi^0\nu$ decay.

of reconstruction (that is, missing events) because we are considering the "True" histogram as the one built from the selected events and they have a 1:1 correspondence with the "Measured" histogram as the invariant mass of every event that passed the data selection stage is being matched with a measured invariant mass. In other words, we are assuming an unitary efficiency, but this is not the case in a real-life scenario and the efficiencies of the detector need to be considered, which would result in the columns of the response matrix not normalized but with a probability less than 1. This is illustrated on figure 7.2.

As in a real experiment with real data there is no way to characterize the background noise present in the signal, the approach followed in this work is to build the response matrix supposing that a clean signal (that is, free of background noise) was achieved by using the reconstruction techniques explained in section 6.2. Even though in a real-world application a more extensive and thorough study on the background noise needs to be considered (specially as the unfolding techniques based on SVD and Quantum Annealing rely on using a basis model from which variations will be estimated), we can see by comparing the histograms to unfold (figure 6.9, or even the raw histogram on figure 6.5) with this basis model (figure 7.3) that it represents in good degree the effects induced by the detector, making our decision suitable for the purpose of this thesis which is to illustrate how (if) the presented unfolding techniques work.

The true signal was achieved by replicating the same conditions (explained on section 14) expected to be fulfilled by the data after passing through the BDT, but using the Monte Carlo variables directly, while the counting was performed by filling a 2D histogram with the true invariant mass on one axis (chosen to be the x-axis) and the measured invariant

mass on the other (chosen to be the y-axis).

By looking at the response matrix, we can conclude that the detector has an almost diagonal response and we expect to see most of the bin migrations in the region under the 1 [GeV]. This can also be seen by looking at the pull plot of the histograms of figure 7.3 where the skew is around the resonance, though there are events present in the measurements but not in the MC simulations around the 1.6 [GeV] as well.

7.2 On the Weak Set

The purpose of showing the results of the unfolding on the weak set is to illustrate why there is the need to get a cleaner data set and not to explore the unfolding techniques per se, so there will not be much attention on how the hyperparameters were chosen (as opposite to section 7.3).

To assess the quality of the unfolding (that is, the resemblance of the `unfolded` histogram to the `matchedMC` histogram), a two-sample Kolmogorov-Smirnov test [26] is used requiring a confidence level of $\alpha = 0.05$ with the following hypotheses:

Null Hypothesis (H_0) The two histograms are identical.
Alternative Hypothesis (H_1) The histograms are not identical.

Simulated Quantum Annealing was performed 100 times with a regularization parameter of 0.5, 4 bits to encode and a scale of 0.5. The results are shown on the left of figure 7.4 and the full correlation matrix is shown on the right. Qualitatively, it can be seen that the unfolded histogram is similar to the `matchedMC` histogram[4] and, quantitatively, this can be expressed by the probability that a given result (or a more significant result) would occur under the null hypothesis. This probability is called **p-value** and, by applying the Kolmogorov-Smirnov test, results in a value of $p = 0.039$ which is lower than α, indicating that both histograms are different and we can reject the null hypothesis (even though histograms "look" similar).

4: It is important to remember that this comparison can be made on simulations because the `matchedMC` histogram is known. With real data, this is not possible.

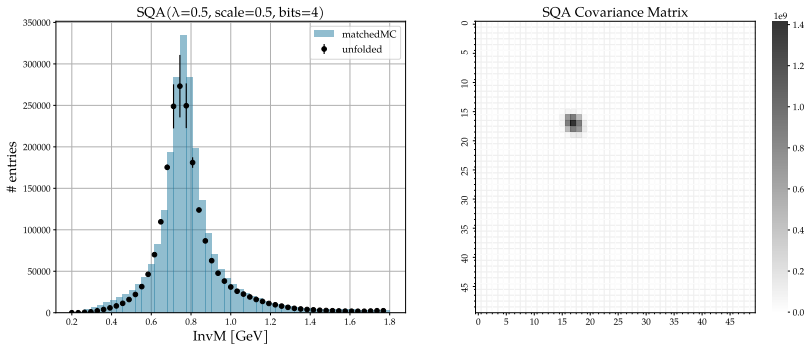


Figure 7.4: Left: result of the SQA unfolding on the weak set. Right: error matrix.

For the case of the SVD approach, it was performed with a regularization parameter equivalent up to the fifth (k=5) d_i which could not be chosen according to the general approach suggested on [12] as the amplitudes of the coefficients did not decrease below 1 as can be seen on figure 7.5, but was heuristically chosen as it yielded the best results. Qualitatively, the results are similar to SQA's in the sense that we do get an histogram

similar to the matchedMC and, quantitatively, it resulted in the rejection of the null hypothesis with a p-value of $p = 0.039$ as well.

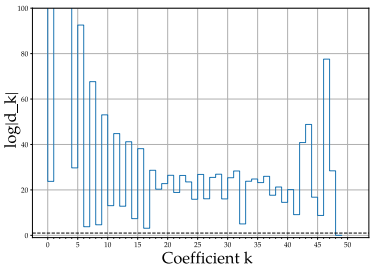


Figure 7.5: Absolute values of d_i when performing SVD on the weak set.

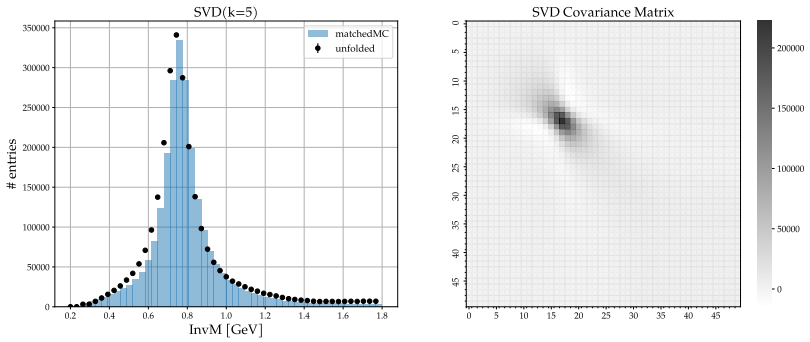


Figure 7.6: Left: result of the SVD unfolding on the weak set. Right: error matrix.

Even though the results suggest that both techniques are comparable in terms of accuracy, neither of them was able to correctly reconstruct the true histogram as produced by the Monte Carlo simulation. To inspect possible causes, the training set was split into signal and background (remembering this is only possible with data coming from simulations) and the unfolding was performed on each slice separately. The result of the split is shown in figure 7.7 for both the Monte Carlo (on the left) and the measured masses (on the right).

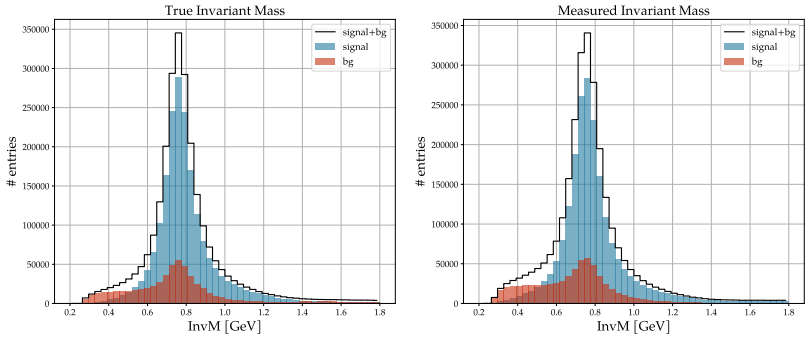


Figure 7.7: Weak set split into signal and background.

After applying the unfolding techniques to the later histograms, it is expected to reconstruct the former histograms and the results are shown on figure 7.8 [5] .

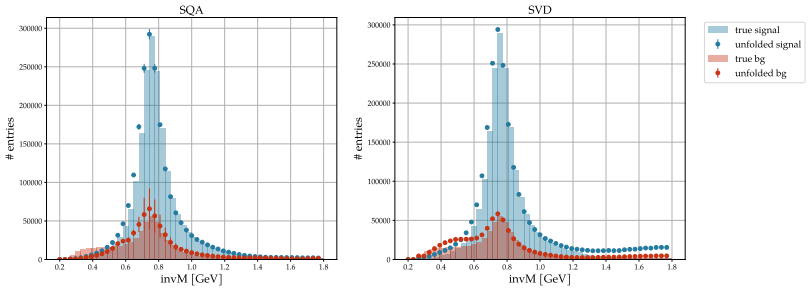


Figure 7.8: Unfolding signal and background splits of the weak set.

5: The error matrix was not shown as it is not the focus point to show the nuances of unfolding noisy histograms.

Again, from a qualitative point of view, both techniques yield similar results, but after comparing its p-values we can notice that SQA performed better for both signal and background with values of $p = 0.997$ and $p = 0.0677$, respectively. These are enough to say that there is chance that both histograms could be the same, which does not happen with SVD as its p-values are given by $p = 0.0002$ and $p = 0.0115$ for signal and background, respectively.

This suggests that the background noise is the main source of trouble that produced the unsatisfactory results of figures 7.4 and 7.6 (specially for the case of SQA), which motivates the following three approaches to improve the results:

1. Build a response matrix that includes this background noise instead of using a response matrix free of background.
2. Keep using the same response matrix, but try to perform the unfolding on a cleaner version of the invariant mass.
3. Keep using the same response matrix, but estimate the background noise well enough to subtract it from the measured signal before performing the unfolding.

In the end, all of these approaches reduce to having a very good knowledge of the background noise that would be present on the data and, additionally, approach 2 would require a very accurate classifier to filter out background so a cleaner version of the signal can be used.

As finding a very good characterization of the background is very far from trivial in any physics process and it would require studies that fall out of the scope of this work (which is to illustrate the unfolding techniques), approach 2 will be chosen and the unfolding will be performed on the strict set which represents a cleaner version of the weak set.

In a real application, simulations are generated as the physicist thinks Nature is going to behave and he decides when the signal and background are reliable enough to build a model, but because there are no weak or strict data sets as there is only one data set measured, it is not possible to know before hand if the simulations were accurate enough.

7.3 On the Strict Set

The results of the SVD unfolding when applied to the strict set are shown in figure 7.10. By looking at the amplitudes of the d_i coefficients on figure 7.9, we can notice that the approach to choose the regularization parameter suggested in [12] will not work either as there is no coefficient with a log amplitude lower than 1 (the dashed black line on the figure), but there is a clear difference in the relative amplitudes between the first coefficient and the rest.

With an heuristic approach, the best result was found to be when using the second coefficient k=2 as it can be seen by visually inspecting the

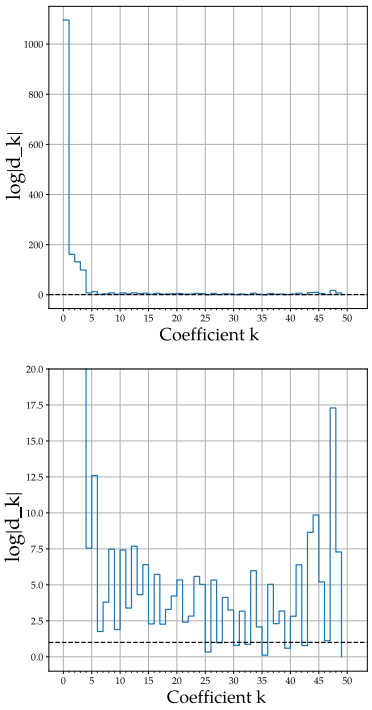


Figure 7.9: Up: Absolute values of d_i when performing SVD on the strict set. Down: a zoomed-in version.

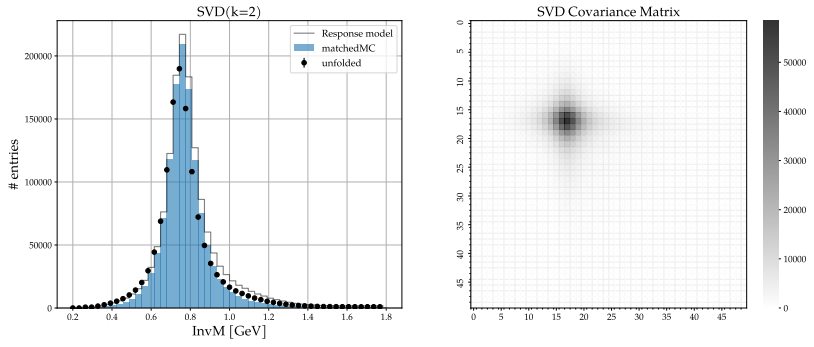


Figure 7.10: Left: result of the SVD unfolding on the strict set. Right: unfolded error matrix.

similarity between the unfolded results and the true MC mass in figure 7.10.

Qualitatively, we can see there are good results to unfold the ρ resonance around 770 [MeV], though the regions before and after the main peak look biased towards the model used to build the response matrix and the background noise around the 1.1 [GeV] could not been unfolded correctly. Quantitatively, the results the p-value is given by $p = 0.0217$ suggesting that both histograms are different.

By looking at the correlation matrix, we can see that the affected area is primarily around the region where the bin migrations are taking place which is surrounding the 770 [MeV].

Following to the results of the SQA, as there is no suggested approach to choose the regularization weight (nor any of the other hyperparameters), a grid search was performed to try different combinations which are shown in figure 7.11.

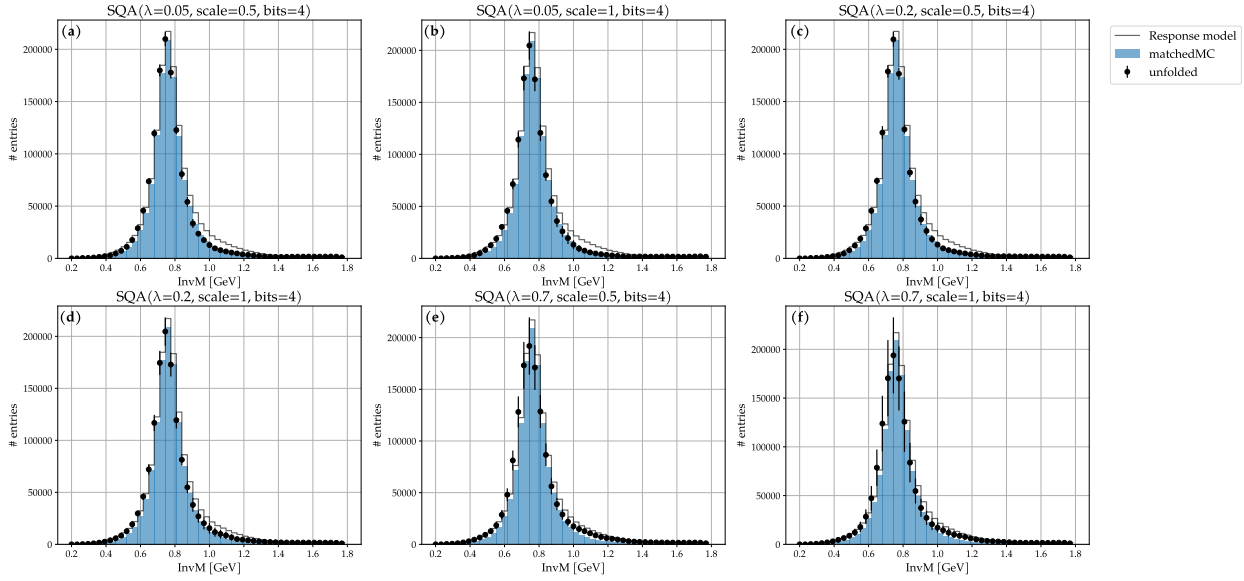


Figure 7.11: Results of the SQA unfolding performed on the strict set with different parameters.

The most important parameters to change are the regularization weight and the scaling used to perform the encoding, so the unfolding was tried with regularization weights $\lambda = \{0.05, 0.2, 0.7\}$ and a scaling factor scale=$\{0.5, 1\}$ from which similar results to that of SVD are observed as the unfolding works better for the main peak while the regions where the noise is present are highly biased towards the response model. The results are shown quantitatively on table 7.1.

Choosing the *best* result is a complication inherent of the unfolding problem. Even though none of them are likely to be equal according to the Kolmogorov-Smirnov tests, if we were forced to choose one, we could rank them by their p-values and the result would have to be "a" or "c", but this test can only be performed with simulated data as we know a priori the `matchedMC` distribution that we are expecting to find. In a real experiment, this histogram is not known and this comparison is not possible.

Table 7.1: Quantitative results of the SQA strict unfolding.

Result	p-value
a	0.02170784069014051
b	0.011511738725894704
c	0.02170784069014051
d	0.011511738725894704
e	0.011511738725894704
f	0.011511738725894704

What we can notice is that increasing the regularization weight and the scale seem to have an impact on the variance as higher weights and higher scales result in higher variance. Until this experiment can be reproduced on a real quantum computer, it is not clear whether this variations are due to the simulation method used in this work or due to the unfolding technique per se.

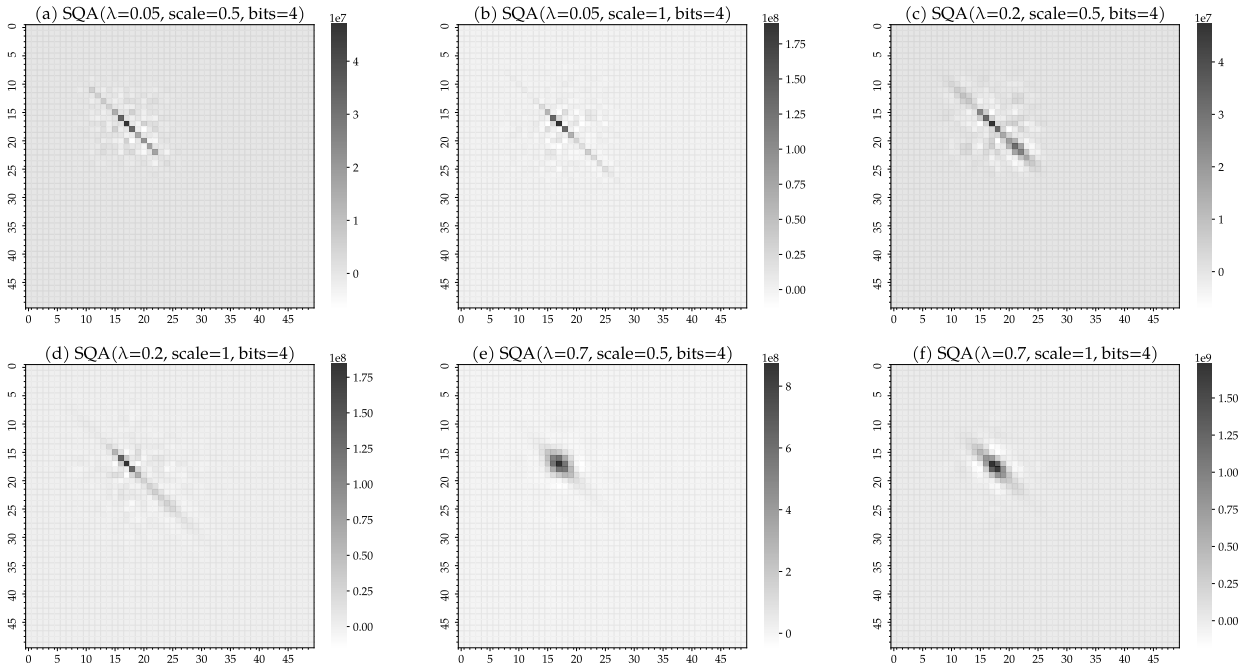


Figure 7.12: Correlation matrices of the SQA results on the strict set.

For completeness, the full correlation matrices are shown on figure 7.12.

Conclusions 8

In this work, a computational experiment was performed to explore how the (Simulated) Quantum Annealing technique would solve the unfolding problem when compared against a more traditional Singular Value Decomposition approach. The experiment used a simulated data set of the $\tau \rightarrow \pi\pi^0\nu$ decay (provided by the Belle II Collaboration) to perform the unfolding on the $\pi\pi^0$ invariant mass.

In this chapter, a direction on how this work can be expanded will be presented and conclusions will be presented as a manner of discussion about the results.

8.1 Future Work

The very first natural next step is to perform this same experiment on a real quantum computer to see how well the simulations approximate the real quantum annealing. By the time of developing this work, access to a real device was not possible as there are not many working quantum annealers and, without any agreement with the developing companies, a significant budget is required. Yet, real quantum annealers are still in an early stage of development and there are other difficulties that would need to be addressed in order to run this work on a real device, such as how noise would affect the measurements and if real devices have enough qubits to map a problem like this (problem known as "embedding"). These points are out of the scope of this work, but more can be found at D-Wave's guide to use their solvers [16].

Another interesting next step would be to perform a similar experiment but choosing a physics process with a steeply-falling spectrum and see how SVD and QA perform. The physics process chosen in this work resulted in a smooth histogram (that is, an histogram without abrupt changes between adjacent bins), but as shown in [4], the results may vary when that is not the case.

As SVD provides a way to compute the unfolded correlation matrix, a contribution to the Quantum Annealing technique would be to see if it is possible to perform a similar calculation and, as mentioned before, it is even of more interest its interpretation and how (if) they relate to a quantity of interest for experimentalists.

Regarding the results, the bias towards the base model is still an issue that needs to be explored and it is not clear whether all techniques and all experiments would have this same issue. For the techniques studied in this work, an alternative scaling technique that is independent of the model would be a possible solution that would need to be investigated more in depth. If the bias would like to be reduced to improve this very same experiment, the other approaches mentioned in section 7.2 could be followed and either 1) a response matrix considering all the background

could be computed (that is, a response matrix based on the strict data set) or 3) see how well the unfolding performs after subtracting a simulated background to the measured data set.

8.2 Discussion and Final Remarks

The results presented in this work confirm that Quantum Annealing (at least in its simulated version) yields comparable results to SVD when used to solve a least squares problem to perform the unfolding, both on the same histogram. Still, solving the unfolding has inherent problems that are independent of the technique (plus the ones that are not) and that still need to be addressed, so this section will focus on highlighting these points and on whether quantum computers introduce any additional benefit or not.

Starting with the hassles of the unfolding, choosing the regularization weight remains an open question that belongs to any linear regression problem. In this case, the method suggested in [12] for the SVD approach did not work exactly as expected as no coefficients were found with an amplitude lower than 1, but, still, lower amplitudes were found which allowed to heuristically choose the best coefficient.

For the case of QA, the open question remains: how to choose the regularization weight? The approach in this work was completely heuristic and, as the main purpose was not to perform the unfolding per se, there was no need to actually "choose" one weight nor one result.

Still, for both cases a quantitative result could be calculated using the Kolmogorov-Smirnov test which, in principle, would allow us to choose the best fit, but it is important to remember that this would not be possible in a real application where the unfolded distribution is not known a priori.

Referring particularly to QA, this technique suffers from the same problem that many Machine Learning techniques, which is how to choose the hyper-parameters in general and not only the regularization weight. That is, the amount of runs, the amount of qubits and the scale weight used in the encoding (and, for the case of SQA, the amount of replicas). This is something that needs to be investigate further if Quantum Annealing is meant to be applied to solve a real problem.

Another point to discuss is how the errors transform after the unfolding. SVD provides a way to compute the unfolded correlation matrix, but its interpretation is not trivial. In contrast, the QA approach does not provide a similar calculation, but **a** correlation matrix can be experimentally measured. In the end, what would be of interest to an experimentalist would be the confidence interval of how well these unfolded results are believed to represent the real true distribution. Neither of these techniques provide a way for its calculation and it is still an open question. For a more detailed discussion on the nuances of the interpretation of the unfolded correlation matrix (for SVD and other "classic" techniques), the reader is referred to [15].

In this particular experiment, it was found that the unfolded results are going to be clearly biased towards the model used to build the response

matrix. This can be due to the scaling technique used in the least squares problems maps unknowns to proportional variations of the base model, so the result is not expected to be much different. As a consequence, either we need to remove the background noise on the histogram to unfold as much as possible to make it similar to the base model, or the base model needs to consider all the possible background effects expected to find in the experiment. Both of these approaches require a good characterisation of the background which is far from trivial and, if it could be achieved, then the benefits of the unfolding may actually not be worth the effort and it would not be necessary.

In the end, one of the most promising aspects of Quantum Computing is to speed up calculations that would be practically impossible to perform on classic computers. This experiment showed that performing the unfolding via quantum annealing has the potential to yield results as good as other classic methods and it showcases one of the many possible ways qubits can be used to perform simulations/calculations, but realistic expectations need to be set when talking about improvements on the performance.

In any unfolding approach, what would benefit the most from a more robust computational device would be the simulations to produce the data and the computation of the response matrix. Once this step is done, performing the unfolding is not a power-demanding task as, for example, computing the SVD factorization of any matrix does not necessarily represent a bottleneck that slows down the experiments. In principle, this last step is already performed with classical computers, so a quantum computer is not strictly required in this procedure, but that does not mean it could not bring any benefit.

Nevertheless, to give an educated and more serious opinion on how much the unfolding could be accelerated with quantum computers, a whole study would need to be carried out considering several things. To begin with, a real quantum computer would be needed and, considering how young this technology still is, the results would probably be tied to the specific architecture of its quantum processing unit. Also, the performance would need to be measured with different tuning parameters (both for the quantum computer setup and for the unfolding technique) and considering different complexities of the problem ranging from a simple unfolding to more complex matrices. This study was out of the scope of this work as the limited access to real quantum annealers makes it costly and impractical.

Until this technique is refined to run flawlessly in real quantum devices, it is difficult to foresee a huge impact on the unfolding as an area of research. Nevertheless, quantum computers are still several years or maybe decades away from becoming a practical technology and finding applications where they truly surpass their classic counterparts with a gap wide enough to justify the effort and investment, is still an active area of research [1] to which this work hopes to have contributed.

1: For this particular case, a discussion on how quantum annealing would perform against simulated quantum annealing, which is its classical counterpart, can be found at [27], [28].

Appendix

A

Simulated Quantum Annealing

Simulated Quantum Annealing is a technique proposed in 2002 [22] to compute the evolution of the Quantum Annealing Hamiltonian (5.5) (written below for reference) using a Monte Carlo approach that takes advantage of the Path Integral formulation.

$$\hat{H} = -\Gamma(t) \sum_i \hat{\sigma}_j^x - \sum_{i=1}^{n} h_i \hat{\sigma}_i^z - \sum_{i<j} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z$$

This Hamiltonian for n spins can be split into a potential energy term (U) and a kinetic energy term (K) that do not commute $[K, U] \neq 0$:

$$U = -\sum_{i=1}^{n} h_i \hat{\sigma}_i^z - \sum_{i<j} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z, \quad K = -\Gamma(t) \sum_i \hat{\sigma}_j^x \tag{A.1}$$

So that, if we wanted to simulate how a system of n particles/spins described by this Hamiltonian evolves with time starting from an initial arbitrary state, Quantum Statistical Mechanics tells us [29] that we need to compute the partition function Z which gives the probability of watching a transition from one configuration to another one:

$$Z = \mathrm{Tr}(e^{-\beta H}) \tag{A.2}$$

Where, as usual, $\beta = 1/kT$ where k is the Boltzmann Constant (which will be set to 1) and T is the temperature of the system.

Developing (A.2) with the Quantum Annealing Hamiltonian by using the Trotter product formula (which introduces an arbitrary scalar P):

$$\begin{aligned} Z =& \mathrm{Tr}(e^{-\beta H}) \\ =& \mathrm{Tr}(e^{-\beta H/P})^P \\ =& \mathrm{Tr}(e^{-\beta (K+U)/P})^P = \sum_{s^1} \dots \sum_{s^P} \langle s^1 | e^{-\beta (K+U)/P} | s^2 \rangle \\ & \times \langle s^2 | e^{-\beta (K+U)/P} \dots | s^P \rangle \langle s^P | e^{-\beta (K+U)/P} | s^1 \rangle \end{aligned} \tag{A.3}$$

To compute this partition function, the Path Integral formula is used to divide the time interval of the whole annealing experiment into small pieces. Once this is done, the Trotter product formula tells us that the noncommutativity of the kinetic and potential energy operators can be ignored $e^{-\beta(K+U)} \approx e^{-\beta K} e^{-\beta U}$, obtaining an approximation Z_P to Z.

The whole procedure can be found in detail in the appendix of [22], but the result is found to be:

$$Z \approx Z_P = C^{n/P} \sum_{s^1} \dots \sum_{s^P} e^{-H_{d+1}/PT} \tag{A.4}$$

Where:

$$H_{d+1} = -\sum_{k=1}^{P} \left(\sum_{i=1}^{n} h_i s_i^k + \sum_{i<j} J_{ij} s_i^k s_j^k + J^{\perp} \sum_i s_i^k s_i^{k+1} \right) \tag{A.5}$$

$$J^{\perp} = -\frac{PT}{2} \ln \tanh \frac{\Gamma}{PT} > 0 \tag{A.6}$$

$$C = \left(\frac{1}{2} \sinh \frac{2\Gamma}{PT} \right)^{1/2} \tag{A.7}$$

Computing the partition function with the Path Integral formulation and the Trotter formula resulted in a convenient interpretation to simulate such a system with a Monte Carlo Metropolis approach because the Trotter scalar P can now be interpreted as the amount of possible paths to consider in the simulation (that is, the number of replicas of the original Ising system to evolve) and the discretization of time is natural as time steps are usually simulated with a loop (i.e. a `for` loop).

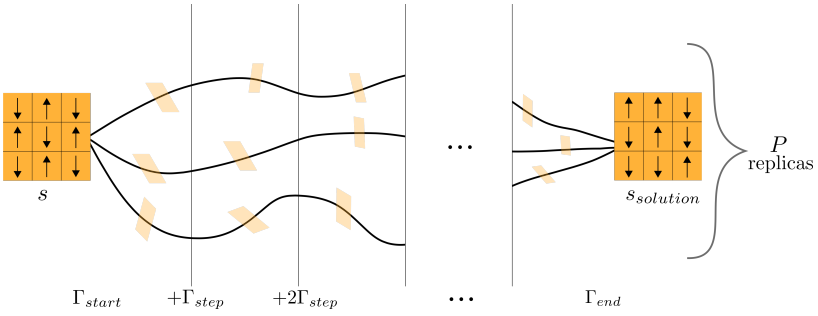


Figure A.1: Illustration of the Simulated Quantum Annealing with n=9 spins in a grid and P replicas.

Implementing the Simulated Quantum Annealing can now be reduced to a couple of steps which are illustrated in figure A.1 and summarized in algorithm 1[1] :

1. Define the initial state **s** which is going to be annealed and an ambient temperature T.
2. Create P copies of the initial state that will simulate the many paths it can take.
3. Set the evolution parameter of the simulation as the strength of the transverse magnetic field Γ, which is going to go from Γ_{start} to Γ_{end} in discrete steps of size Γ_{step}.
4. For each value of Γ, try to flip $n \times P$ random spins by using the following rule:
   - If the spin flip leads to a state with a lower energy in its corresponding replica, commit the spin flip.
   - If the spin flip leads to a state with a higher energy in its corresponding replica, compute the probability of transition using the partition function (A.4) and decide whether to flip

1: The full implementation can be found at [23]. Note that [22] considers only the interaction term for H, but it is straightforward to extend the result to the full Hamiltonian (5.5).

or not using a Monte Carlo approach (i.e. compare with a random number).

After performing these steps for every step of Γ, the replica with the lowest energy is taken as the result and that is the output of the algorithm.

Algorithm 1: Pseudo-code of the Simulated Quantum Annealing

```
1  n ← number of spins in the Ising system
2  P ← number of Trotter replicas
3  T ← ambient temperature
4  Γ_start, Γ_end ← starting and ending transverse magnetic field strengths
5  Γ_step ← amount to decrease the transverse magnetic field at each time step
6  s ← {s_1, s_2, ..., s_n} initial state of n spins
7
8  function Metropolis(s, s_i, γ) is
9      s' ← configuration after spin s_i is flipped on s
10     Δ ← E(s, γ) − E(s', γ)            /* Calculate the transition energy */
11     if Δ > 0 or e^(Δ/T) > rand() then
12         return s'                                  /* Move to new state */
13     else
14         return s                              /* Stay on the same state */
15     end
16 end
17
18 S ← {s_1, s_2, ..., s_P}             /* Create P copies of the initial state */
19
20 N ← n × P                 /* Compute the amount of spins in all replicas */
21 for γ ∈ [Γ_start : Γ_step : Γ_end] do
22     for N times do                                 /* Try to flip N spins */
23         s_i, s_j ← pick a random spin s_j and its replica s_i
24         s_i ← Metropolis (s_i, s_j, γ)
25     end
26 end
27
28 s_solution ← s with lowest energy in S
```

B

Steering File for the $\tau \longrightarrow \pi\pi^0\nu$ reconstruction

```
# ######################################################################
# Selection of 3%1 prong taupair decays with loose cuts
#
#                 l(h) nu nu
#                  |
#        e+ e- -> tau+ tau-
#                        |
#                     pi+ l- l- nu
#
# Contributors:
#  Eduard De La Cruz Burelo
#  Carlos Pegueros
#
# last modified: June 2021
# ######################################################################

# !/usr/bin/env python3
# -*- coding: utf-8 -*-

include_ks_modes = False
include_HLT = True
include_L1 = False

import basf2 as b2
import modularAnalysis as ma
import vertex as vx
import sys
import variables.collections as vc
import variables.utils as vu
from variables import variables

default_output = "tau2pipi0"

b2.conditions.reset()
b2.conditions.prepend_globaltag("Legacy_IP_Information")
b2.conditions.append_globaltag("leptonid_Moriond2021_Official_v3")

######################################################
# Specify path to data sample and database
######################################################

main = b2.create_path()

# read input
arg_dataORmc = str(sys.argv[1])
arg_input = str(sys.argv[2])
arg_output = str(sys.argv[3])
```

```
if arg_dataORmc:
    arg_output = default_output + ".root"

b2.B2INFO("CHECK: analysing MC, is this correct?")
b2.conditions.append_globaltag("leptonid_ICHEP2020_Official_v0")
ma.inputMdst("default", "%s" % arg_input, path=main)

######################################################
# create and fill the ParticleLists
######################################################
ma.fillParticleList("e-:all", "", path=main)
ma.fillParticleList("mu-:all", "", path=main)
ma.fillParticleList("pi+:all", "", path=main)
ma.fillParticleList("gamma:all", "", path=main)

gammaForPi0Cuts = "E > 0.1"
gammaForPi0Cuts += " and -0.8660 < cosTheta < 0.9563"
gammaForPi0Cuts += " and clusterNHits > 1.5"
ma.cutAndCopyLists("gamma:looseForPi0", "gamma:all", gammaForPi0Cuts, path=main)

#################################################################################
# pi0 reconstruction with loose photons
#################################################################################
### photons from pi0s and pi0s
ma.reconstructDecay(
    "pi0:fromLooseGammas -> gamma:looseForPi0 gamma:looseForPi0",
    "0.115 < M < 0.152",
    path=main,
)
ma.cutAndCopyLists(
    "gamma:pi0",
    "gamma:looseForPi0",
    "isDescendantOfList(pi0:fromLooseGammas) == 1",
    path=main,
)

variables.addAlias("nPhotonsFromPi0", "countInList(gamma:pi0)")
variables.addAlias("nPi0", "countInList(pi0:fromLooseGammas)")

#########################################################################################################
# track (not used for reconstruction of K_0S) and photon (not used for reconstruction of pi0) cuts
#########################################################################################################
trackCuts = "-3.0 < dz < 3.0 and dr < 1.0"

# reconstruct electrons
eIDCuts = "electronID > 0.5"
ma.cutAndCopyLists("e-:pid", "e-:all", f"{trackCuts} and {eIDCuts}", path=main)

# reconstruct muons
muIDCuts = "muonID > 0.5"
ma.cutAndCopyLists("mu-:pid", "mu-:all", f"{trackCuts} and {muIDCuts}", path=main)

# reconstuct pions for the 1prong tag
piIDCuts_tag = "pionID > 0.5"
ma.cutAndCopyLists("pi+:pid", "pi+:all", f"{trackCuts} and {piIDCuts_tag}", path=main)

# reconstruct pions for the signal (tau->pi+ pi0 nu)
variables.addAlias("EoverP", "formula( ifNANgiveX( clusterE, -1 )/p )")
```

```
piIDCuts_signal = "EoverP < 0.8 and pionID > 0.5"
ma.cutAndCopyLists("pi+:pidsig", "pi+:all", f"{trackCuts} and {piIDCuts_signal}", path=main)
variables.addAlias("nGoodTracks", "formula(countInList(pi+:pid) + countInList(e+:pid) + countInList
    (mu+:pid))")

######################################################
# event based cut - 2 tracks in event and at least one pi0
######################################################
ma.applyEventCuts("nGoodTracks == 2", path=main)
ma.applyEventCuts("nPi0 > 0", path=main)

gammaCuts = "E > 0.2"
gammaCuts += " and -0.8660 < cosTheta < 0.9563"
gammaCuts += " and clusterNHits > 1.5"
gammaCuts += " and isDescendantOfList(pi0:fromLooseGammas) == 0"
ma.cutAndCopyLists("gamma:notPi0", "gamma:all", gammaCuts, path=main)
variables.addAlias("nGoodPhotons", "countInList(gamma:notPi0)")

######################################################
# EventShape and EventKinematics modules
######################################################
particleList = ["pi+:pid", "pi0:fromLooseGammas", "gamma:notPi0"]

ma.buildEventShape(
    particleList,
    foxWolfram=False,
    cleoCones=False,
    jets=False,
    harmonicMoments=False,
    allMoments=False,
    collisionAxis=False,
    sphericity=False,
    thrust=True,
    path=main,
)
ma.buildEventKinematics(particleList, path=main)

#############################################################################################
####
####    Tau -> pi pi0 nu_tau
####
#############################################################################################

######################################################
# Signal and tag sides
######################################################

# --- signal side
ma.reconstructDecay("tau+:sig -> pi+:pidsig pi0:fromLooseGammas", "", path=main)

# -- 1 prong side (tag)
ma.reconstructDecay("tau-:e -> e-:pid", "", path=main, dmID=11)
ma.reconstructDecay("tau-:pi -> pi-:pid", "", path=main, dmID=211)
ma.reconstructDecay("tau-:mu -> mu-:pid", "", path=main, dmID=13)

tau_1prongLists = [
    "tau-:e",
    "tau-:pi",
```

```
    "tau-:mu",
]

ma.copyLists("tau-:1prong", tau_1prongLists, path=main)

ma.reconstructDecay("vpho -> tau+:sig tau-:1prong", "", path=main)

variables.addAlias(
    "dmID_1prong", "daughter(1, extraInfo(decayModeID))"
)  # reconstructed 1-prong decay mode

######################################################
# 2 tracks on the opposide sides
######################################################
variables.addAlias(
    "track_1_x_track_2",
    "formula(daughter(0, daughter(0, cosToThrustOfEvent))*daughter(1, daughter(0,cosToThrustOfEvent
    )))",
)

ma.applyCuts("vpho", "track_1_x_track_2 < 0", path=main)

#############################################################################################
# number of photons and pi0 on 1-prong signal side and 1-prong tag side
# put requirements on the number of pi0s on 1-prong to choose the decay mode
#############################################################################################

ma.copyList("gamma:1prong", "gamma:notPi0", path=main)
ma.copyList("gamma:1prongPi0", "gamma:notPi0", path=main)

ma.copyList("pi0:1prong", "pi0:fromLooseGammas", path=main)
ma.copyList("pi0:1prongPi0", "pi0:fromLooseGammas", path=main)

# 1-prong on positive or negative side of thrust axis
variables.addAlias(
    "1prongInPosThrust",
    "countInList(vpho, daughter(1, daughter(0,cosToThrustOfEvent)) > 0)",
)
variables.addAlias(
    "1prongInNegThrust",
    "countInList(vpho, daughter(1, daughter(0,cosToThrustOfEvent)) < 0)",
)

positiveThrust = b2.create_path()
negativeThrust = b2.create_path()

ma.applyCuts("gamma:1prong", "cosToThrustOfEvent > 0", path=positiveThrust)
ma.applyCuts("gamma:1prongPi0", "cosToThrustOfEvent < 0", path=positiveThrust)

ma.applyCuts("gamma:1prong", "cosToThrustOfEvent < 0", path=negativeThrust)
ma.applyCuts("gamma:1prongPi0", "cosToThrustOfEvent > 0", path=negativeThrust)

ma.applyCuts("pi0:1prong", "cosToThrustOfEvent > 0", path=positiveThrust)
ma.applyCuts("pi0:1prongPi0", "cosToThrustOfEvent < 0", path=positiveThrust)

ma.applyCuts("pi0:1prong", "cosToThrustOfEvent < 0", path=negativeThrust)
ma.applyCuts("pi0:1prongPi0", "cosToThrustOfEvent > 0", path=negativeThrust)

```

```
# take different paths if 1-prong in cosToThrustOfEvent > or < 0
sigThrustModule = main.add_module("VariableToReturnValue", variable="1prongInPosThrust")
sigThrustModule.if_value("> 0", positiveThrust, b2.AfterConditionPath.CONTINUE)
sigThrustModule = main.add_module("VariableToReturnValue", variable="1prongInNegThrust")
sigThrustModule.if_value("> 0", negativeThrust, b2.AfterConditionPath.CONTINUE)

# the number of photons and pi0s in 3prong and 1prong hemispheres
variables.addAlias("nPhotons_1prong", "nParticlesInList(gamma:1prong)")
variables.addAlias("nPhotons_sig", "nParticlesInList(gamma:1prongPi0)")

variables.addAlias("nPi0s_1prong", "nParticlesInList(pi0:1prong)")
variables.addAlias("nPi0s_sig", "nParticlesInList(pi0:1prongPi0)")

######################################################
# kinematics of 1-prong+Pi0 and 1-prong
######################################################
variables.addAlias("photonE_sig", "totalEnergyOfParticlesInList(gamma:1prongPi0)")
variables.addAlias("photonE_1prong", "totalEnergyOfParticlesInList(gamma:1prong)")

variables.addAlias(
    "photonECMS_sig", "useCMSFrame(totalEnergyOfParticlesInList(gamma:1prongPi0))"
)
variables.addAlias(
    "photonECMS_1prong", "useCMSFrame(totalEnergyOfParticlesInList(gamma:1prong))"
)

ma.cutAndCopyList("tau+:sigFromVpho", "tau+:sig", "isDescendantOfList(vpho)", path=main)
ma.cutAndCopyLists(
    "tau-:1prongFromVpho", tau_1prongLists, "isDescendantOfList(vpho)", path=main
)

variables.addAlias(
    "ECMS_sig", "useCMSFrame(totalEnergyOfParticlesInList(tau+:sigFromVpho))"
)
variables.addAlias(
    "ECMS_1prong", "useCMSFrame(totalEnergyOfParticlesInList(tau-:1prongFromVpho))"
)

variables.addAlias(
    "ECMS_sig_photons",
    "formula(useCMSFrame(totalEnergyOfParticlesInList(tau+:sigFromVpho)) + useCMSFrame(
    totalEnergyOfParticlesInList(gamma:1prongPi0)) + useCMSFrame(totalEnergyOfParticlesInList(pi0:1
    prongPi0)) )",
)

variables.addAlias(
    "ECMS_sig_photons_pi0s",
    "formula(useCMSFrame(totalEnergyOfParticlesInList(tau+:sigFromVpho)) + useCMSFrame(
    totalEnergyOfParticlesInList(gamma:1prongPi0)))",
)

variables.addAlias(
    "ECMS_1prong_photons",
    "formula(useCMSFrame(totalEnergyOfParticlesInList(tau-:1prongFromVpho)) + useCMSFrame(
    totalEnergyOfParticlesInList(gamma:1prong)))",
)

variables.addAlias("M_sig", "invMassInLists(tau+:sigFromVpho)")
```

```
variables.addAlias("M_1prong", "invMassInLists(tau-:1prongFromVpho)")

variables.addAlias("M_sig_photons", "invMassInLists(tau+:sigFromVpho, gamma:1prongPi0)")
variables.addAlias(
    "M_1prong_photons", "invMassInLists(tau-:1prongFromVpho, gamma:1prong)"
)

######################################################
# perform MC matching for MC samples
######################################################
if arg_dataORmc == "mc":
    ma.matchMCTruth("vpho", path=main)
    ma.labelTauPairMC(path=main)

###############################################################################
# PID corrections
###############################################################################

payload_elID_eff = "ParticleReweighting:electronID_eff_combination_09"
variables.addAlias(
    "weight_elID_eff_tmp", f"extraInfo({payload_elID_eff}_data_MC_ratio)"
)
variables.addAlias(
    "weight_elID_eff_stat_up_tmp",
    f"extraInfo({payload_elID_eff}_data_MC_uncertainty_stat_up)",
)
variables.addAlias(
    "weight_elID_eff_stat_dn_tmp",
    f"extraInfo({payload_elID_eff}_data_MC_uncertainty_stat_dn)",
)
variables.addAlias(
    "weight_elID_eff_sys_up_tmp",
    f"extraInfo({payload_elID_eff}_data_MC_uncertainty_sys_up)",
)
variables.addAlias(
    "weight_elID_eff_sys_dn_tmp",
    f"extraInfo({payload_elID_eff}_data_MC_uncertainty_sys_dn)",
)

payload_muID_eff = "ParticleReweighting:muonID_eff_combination_09"
variables.addAlias(
    "weight_muID_eff_tmp", f"extraInfo({payload_muID_eff}_data_MC_ratio)"
)
variables.addAlias(
    "weight_muID_eff_stat_up_tmp",
    f"extraInfo({payload_muID_eff}_data_MC_uncertainty_stat_up)",
)
variables.addAlias(
    "weight_muID_eff_stat_dn_tmp",
    f"extraInfo({payload_muID_eff}_data_MC_uncertainty_stat_dn)",
)
variables.addAlias(
    "weight_muID_eff_sys_up_tmp",
    f"extraInfo({payload_muID_eff}_data_MC_uncertainty_sys_up)",
)
variables.addAlias(
    "weight_muID_eff_sys_dn_tmp",
    f"extraInfo({payload_muID_eff}_data_MC_uncertainty_sys_dn)",
```

```
)

payload_elID_fake = "ParticleReweighting:electronID_misid_pi_combination_09"
variables.addAlias(
    "weight_elID_fake_tmp", f"extraInfo({payload_elID_fake}_data_MC_ratio)"
)
variables.addAlias(
    "weight_elID_fake_stat_up_tmp",
    f"extraInfo({payload_elID_fake}_data_MC_uncertainty_stat_up)",
)
variables.addAlias(
    "weight_elID_fake_stat_dn_tmp",
    f"extraInfo({payload_elID_fake}_data_MC_uncertainty_stat_dn)",
)
variables.addAlias(
    "weight_elID_fake_sys_up_tmp",
    f"extraInfo({payload_elID_fake}_data_MC_uncertainty_sys_up)",
)
variables.addAlias(
    "weight_elID_fake_sys_dn_tmp",
    f"extraInfo({payload_elID_fake}_data_MC_uncertainty_sys_dn)",
)

payload_muID_fake = "ParticleReweighting:muonID_misid_pi_combination_09"
variables.addAlias(
    "weight_muID_fake_tmp", f"extraInfo({payload_muID_fake}_data_MC_ratio)"
)
variables.addAlias(
    "weight_muID_fake_stat_up_tmp",
    f"extraInfo({payload_muID_fake}_data_MC_uncertainty_stat_up)",
)
variables.addAlias(
    "weight_muID_fake_stat_dn_tmp",
    f"extraInfo({payload_muID_fake}_data_MC_uncertainty_stat_dn)",
)
variables.addAlias(
    "weight_muID_fake_sys_up_tmp",
    f"extraInfo({payload_muID_fake}_data_MC_uncertainty_sys_up)",
)
variables.addAlias(
    "weight_muID_fake_sys_dn_tmp",
    f"extraInfo({payload_muID_fake}_data_MC_uncertainty_sys_dn)",
)

reweighter_el_eff = b2.register_module("ParticleWeighting")
reweighter_el_eff.set_name(f"ParticleWeighting_e+:tag_eff")
reweighter_el_eff.param("tableName", payload_elID_eff)
reweighter_el_eff.param("particleList", "e-:pid")
main.add_module(reweighter_el_eff)

reweighter_mu_eff = b2.register_module("ParticleWeighting")
reweighter_mu_eff.set_name(f"ParticleWeighting_mu+:tag_eff")
reweighter_mu_eff.param("tableName", payload_muID_eff)
reweighter_mu_eff.param("particleList", "mu-:pid")
main.add_module(reweighter_mu_eff)

reweighter_el_fake = b2.register_module("ParticleWeighting")
reweighter_el_fake.set_name(f"ParticleWeighting_e+:tag_fake")
```

```
reweighter_el_fake.param("tableName", payload_elID_fake)
reweighter_el_fake.param("particleList", "e-:pid")
main.add_module(reweighter_el_fake)

reweighter_mu_fake = b2.register_module("ParticleWeighting")
reweighter_mu_fake.set_name(f"ParticleWeighting_mu+:tag_fake")
reweighter_mu_fake.param("tableName", payload_muID_fake)
reweighter_mu_fake.param("particleList", "mu-:pid")
main.add_module(reweighter_mu_fake)

######################################################
# select the variables to be stored in the ntuple
######################################################
# CMS variables
variables.addAlias("E_CMS", "useCMSFrame(E)")
variables.addAlias("p_CMS", "useCMSFrame(p)")
variables.addAlias("px_CMS", "useCMSFrame(px)")
variables.addAlias("py_CMS", "useCMSFrame(py)")
variables.addAlias("pz_CMS", "useCMSFrame(pz)")
variables.addAlias("pt_CMS", "useCMSFrame(pt)")
variables.addAlias("theta_CMS", "useCMSFrame(theta)")
variables.addAlias("phi_CMS", "useCMSFrame(phi)")

# -- event based variables
eventVariables = [
    "dmID_1prong",
    "nGoodTracks",
    "nGoodPhotons",
    "nPhotons_sig",
    "nPhotons_1prong",
    "nPhotonsFromPi0",
    "nPi0",
    "nPi0s_sig",
    "nPi0s_1prong",
    "1prongInPosThrust",
    "1prongInNegThrust",
    "Ecms", "beamE",
    "thrust",
    "thrustAxisCosTheta",
    "visibleEnergyOfEventCMS",
    "totalPhotonsEnergyOfEvent",
    "photonE_sig",
    "photonECMS_sig",
    "photonE_1prong",
    "photonECMS_1prong",
    "ECMS_sig",
    "ECMS_sig_photons",
    "ECMS_sig_photons_pi0s",
    "ECMS_1prong",
    "ECMS_1prong_photons",
    "M_sig",
    "M_sig_photons",
    "M_1prong",
    "M_1prong_photons",
]

commonVariables = vc.kinematics + vc.inv_mass
commonVariables += ["theta", "cosTheta", "phi"]
```

```
commonVariables += [
    "E_CMS",
    "p_CMS",
    "px_CMS",
    "py_CMS",
    "pz_CMS",
    "pt_CMS",
    "theta_CMS",
    "phi_CMS",
]
commonVariables += ["charge", "cosToThrustOfEvent"]

# -- tau candidate variables
tauVariables = vc.inv_mass  # la podemos calcular, pero igual la pongo
# -- track level variables
##
variables.addAlias("hasAncTau_P", "hasAncestor(-15,1)")
variables.addAlias("hasAncTau_M", "hasAncestor(15,1)")
variables.addAlias("isPDGEl", "matchedMCHasPDG(11)")
variables.addAlias("isPDGPi", "matchedMCHasPDG(211)")
trackVariables = ["clusterE", "EoverP"] + vc.pid + vc.track_hits + ["dz", "dr"]

# -- photon and pi0 variables
pi0Variables = vc.inv_mass  # + ['rank']

# -- MC specific info
if arg_dataORmc == "mc":
    # -- event variables
    eventVariables += [
        "tauMinusMCMode",
        "tauPlusMCMode",
        "tauMinusMCProng",
        "tauPlusMCProng",
    ]

    # -- common variables
    commonVariables += vc.mc_variables + vc.mc_truth

    variables.addAlias("mcE_CMS", "matchedMC(useCMSFrame(E))")
    variables.addAlias("mcP_CMS", "matchedMC(useCMSFrame(p))")
    variables.addAlias("mcPX_CMS", "matchedMC(useCMSFrame(px))")
    variables.addAlias("mcPY_CMS", "matchedMC(useCMSFrame(py))")
    variables.addAlias("mcPZ_CMS", "matchedMC(useCMSFrame(pz))")
    variables.addAlias("mcPT_CMS", "matchedMC(useCMSFrame(pt))")
    cms_mc_kinematics = [
        "mcE_CMS",
        "mcP_CMS",
        "mcPX_CMS",
        "mcPY_CMS",
        "mcPZ_CMS",
        "mcPT_CMS",
    ]

    tauVariables += vc.mc_variables
    tauVariables += cms_mc_kinematics
    tauVariables += [
        "matchedMC(InvM)"
    ]
```

```
    trackVariables += cms_mc_kinematics

# (real) photon variables
gammaVariables = vc.kinematics + ["clusterE", "EoverP"] + ["dz", "dr"]

def get_unique_list(l):
    return list(dict.fromkeys(l))

vphoVariableList = (
    vu.create_aliases_for_selected(
        list_of_variables=get_unique_list(eventVariables + ["cosTheta"]), decay_string="^vpho"
    )
    + vu.create_aliases_for_selected(
        list_of_variables=get_unique_list(commonVariables + tauVariables),
        decay_string="vpho -> ^tau+ ^tau-",
        prefix=["tau_sig", "tau_1prong"],
    )
    + vu.create_aliases_for_selected(
        list_of_variables=get_unique_list(commonVariables + ["dz", "dr", "
    isSignalAcceptMissingNeutrino"]),
        decay_string="vpho -> [tau+ -> pi- [^pi0 -> gamma gamma]] [tau- -> e-]",
        prefix=["neutral_sig"],
    )
    + vu.create_aliases_for_selected(
        list_of_variables=get_unique_list(commonVariables + trackVariables + ["dz", "dr", "
    isSignalAcceptMissingNeutrino"] ),
        decay_string="vpho -> [tau+ -> ^pi- [pi0 -> gamma gamma]] [tau- -> ^e-]",
        prefix=["track_sig", "track_1prong"],
    )
    + vu.create_aliases_for_selected(
        list_of_variables=get_unique_list(gammaVariables),
        decay_string="vpho -> [tau+ -> pi- [pi0 -> ^gamma ^gamma]] [tau- -> e-]",
        prefix=["sig_gamma1", "sig_gamma2"],
    )
)

######################################################
# Write flat ntuples
######################################################

ma.variablesToNtuple(
    decayString="vpho",
    variables=vphoVariableList,
    filename=arg_output,
    treename="tau1x1",
    path=main,
)

# Process the events
b2.process(main)
```

C

Decision Trees and Ensembles

The purpose of this appendix is to give intuition on Decision Trees and an ensemble technique called Gradient Boosted Decision Trees. The content is based on [30] and a more formal description can be found at [31].

C.1 Decision Trees

Decision trees are built as a set of rules in the form of **threshold conditions** for both classification and regression problems. These rules are stored in **nodes** that are then combined in a hierarchical manner to make up a **binary tree**. The first node in the structure is called the **root node** and each subsequent node is called a **leaf node**.

When they are used to solve classification problems (as is the case of this work), they try to look for the set of rules on a set of features [1] $\mathbf{X} = \{x_0, x_1, ...\}$ that can split best the data into the amount of classes desired.

1: "Feature" is the name given to variables in the machine learning argot.

To illustrate the concept, a simple example is shown in figure C.1 with only two features and two classes that resembles a particle physics example: **signal** and **background**.

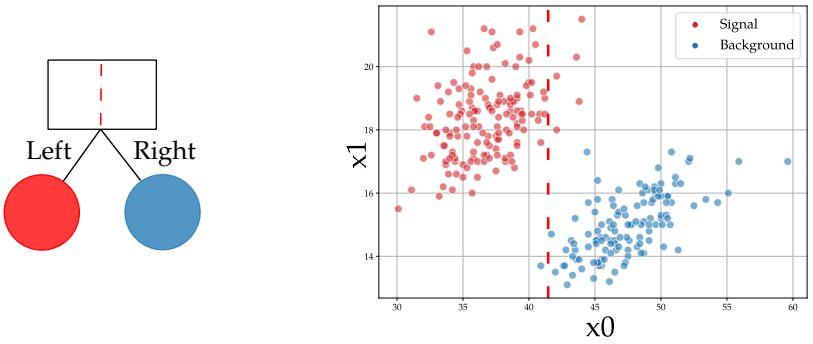


Figure C.1: An classification problem with a Decision Tree with a max depth of 1.

In this case, the rule on the root node is defined as a threshold on the x_0 variable which is represented as a vertical line.

If the event to classify has a value for x_0 that is lower than the threshold, then it is classified as signal by moving one step forward in the left direction. If the value is greater, it is classified as background by moving one step forward in the right direction.

Whenever we add a new split node, the selected feature and the optimal threshold are found by maximizing the improvement in a quantity named entropy or gini index that measures how mixed are the classes in the

nodes. The goal is to find splits such that both child nodes are as pure as possible

The tree predicts the fraction of training points of each class that reached that node, divided by the total number of training points that reached that node. Those values can be interpreted as class assignment probabilities.

We can incrementally expand any leaf to refine the decision function. At each step, the leaf focuses on a smaller sub-region of the space as shown in figure C.2.

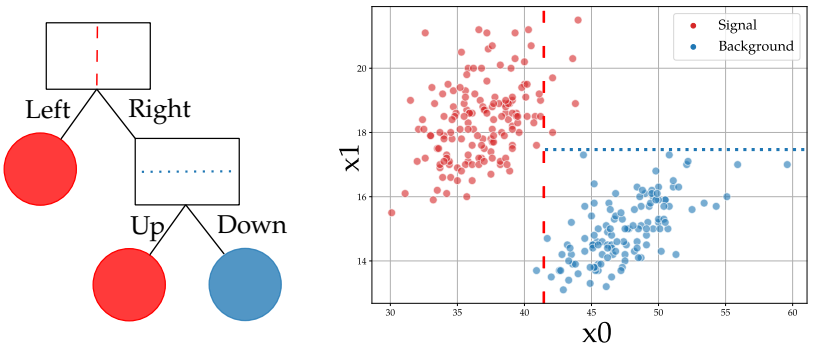


Figure C.2: An classification problem with a Decision Tree with a max depth of 2.

In this example, after two splits, we obtain **pure leaves**. i.e. in each leaf, there is only one class, so the max depth here is equal to 2 and we do not need to go deeper.

C.2 Boosting

Ensemble methods in Machine Learning consist of combining many simple base models to produce one optimal predictive model and Boosting (from which Gradient Boosted Decision Trees get their name) is one of them.

In Boosting, one often considers many homogeneous weak learners and one's output is fed as the input to the next one in a sequential way.

When used with trees, the weak learners are Decision Trees with a very small max depth (i.e. 3, 5 or 8) where mistakes done by the first tree model are corrected by the second tree model, and so on. The final model is a weighted sum of all the trees.

In traditional boosting (AdaBoost), errors are corrected by re-weighting samples at each step so that misclassified events are then given a higher importance, but in Gradient Boosting, errors of one model are predicted by the next model and subtracted from the prediction in order to reduce the error to zero as much as possible.

One advantage of Boosting techniques is that every learner is expected to under-fit individually as a result of it being shallow, but adding each one of them results in an under-fitting reduction of the whole model.

Gradient Boosted Decision Trees

When using Gradient Boosted Decision Trees to solve a classification problem (that is, to predict a target value $y(\boldsymbol{x})$ that represents the probability of classification), it is more convenient to work with the logarithm of the **odds** instead of the probabilities directly.

Odds are defined as the number of "successes" per "failure" and can be translated to probabilities with a simple transformation:

$$\text{odds} = \frac{\#\,\text{successes}}{\#\,\text{failures}} = \frac{p}{1-p} \tag{C.1}$$

So, as the odds are defined in the interval $(0, \infty)$, they map to log-odds on $(-\infty, \infty)$ so that they can be used on regression equations without worrying about additional restrictions:

$$\log(\text{odds}) = \log\left(\frac{p}{1-p}\right) \in (-\infty, \infty) \tag{C.2}$$

With this in mind, Gradient Boosted Trees can be described in the following steps:

First, an initial prediction $\hat{y}$ is made by a naive probabilistic prediction (the output will be the most frequent class).

$$\hat{y} = \hat{y_0} = \log(\text{odds}) \tag{C.3}$$

The prediction errors for this first step R_1 which will be called **residuals**, are then computed as a simple subtraction:

$$\hat{R_1} = y - \hat{y_0} \tag{C.4}$$

where y is given in terms of the real $\log(odds)$ and then it becomes the target to predict for the first weak classifier:

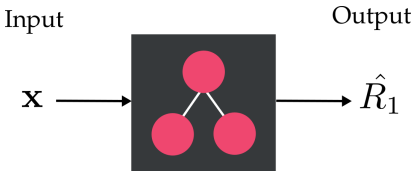


Figure C.3: Prediction of the first Decision Tree in the GBDT.

And this error is then subtracted to the original prediction (with a multiplier λ that works as a learning rate for the algorithm) to create a new estimator:

$$\hat{y} = \hat{y_0} + \lambda\hat{R_1} \equiv \hat{y_1} \tag{C.5}$$

The residual is estimated again by a new tree in the chain.

$$\hat{R_2} = y - \hat{y_1} \tag{C.6}$$

And is added to the prediction.

$$\hat{y} = \hat{y_0} + \lambda \hat{R_1} + \lambda \hat{R_2} \equiv \hat{y_2} \tag{C.7}$$

And passed as the target of the next classifier and so on, so the final prediction $\hat{y}$ for n estimators (as illustrated in figure C.4) becomes:

$$\hat{y} = \hat{y_0} + \lambda \hat{R_1} + \lambda \hat{R_2} + ... + \lambda \hat{R_n} \tag{C.8}$$

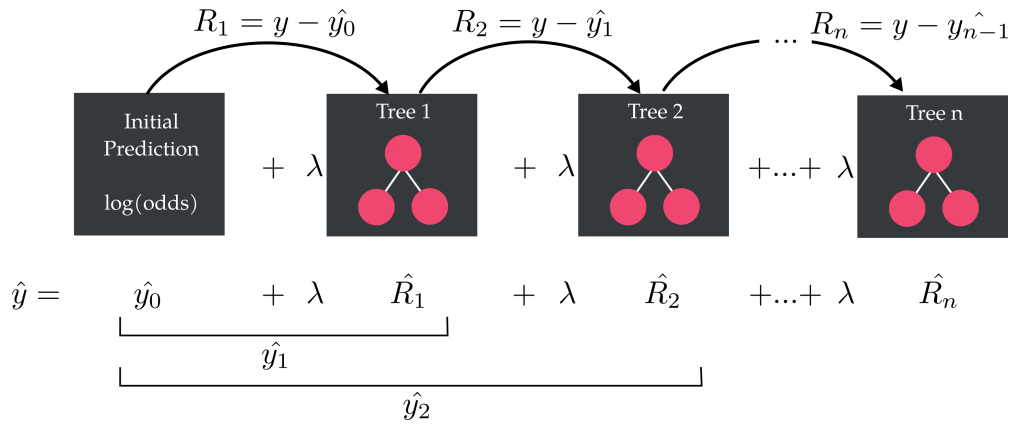


Figure C.4: Illustration of the Gradient Boosted Decision Trees algorithm.

This can be casted to an optimization/regression algorithm if we define a **Loss Function** L to be minimized as the negative log likelihood of the data y as a function of the predicted log(odds) [2] :

$$L(y, \log(\text{odds})) = -y \log(\text{odds}) + \log(1 + e^{\log(\text{odds})}) \tag{C.9}$$

2: The Loss Function is usually defined as a function of the predict probabilities, but in this case we can make use of equation (C.2) to go between one and the other.

So that the gradient (or the derivative) to compute by each model can be written as the residuals needed to improve the prediction yielded by the previous model $\gamma = \log(\text{odds})$:

$$\begin{aligned} \frac{dL(y,\gamma)}{d\gamma} &= \frac{d}{d\log(\text{odds})}\left(-y\log(\text{odds}) + \log(1 + e^{\log(\text{odds})})\right) \\ &= -y + \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} \\ &= -y + \hat{y} \\ &= -\text{real} + \text{predicted} \equiv -R \end{aligned} \tag{C.10}$$

In the end, predictions are given in terms of the log(odds) and can be easily translated to predicted probabilities $\hat{p}$ by solving equation (C.2) for p, that results in:

$$\hat{p} = \frac{1}{1 + e^{-\hat{y}}}, \quad \text{where} \quad \hat{y} = \text{predicted log(odds)} \tag{C.11}$$

And the predicted class can now be the one with the highest probability (or the highest log(odds)) predicted by the model.

Bibliography

Here are the references in citation order.

[1] Steven Weinberg. 'A Model of Leptons'. In: *Phys. Rev. Lett.* 19 (1967), pp. 1264–1266. DOI: 10.1103/PhysRevLett.19.1264 (cited on page 1).

[2] A. Pais and S. B. Treiman. 'How Many Charm Quantum Numbers are There?' In: *Phys. Rev. Lett.* 35 (23 Dec. 1975), pp. 1556–1559. DOI: 10.1103/PhysRevLett.35.1556 (cited on page 1).

[3] Richard P Feynman. 'Simulating physics with computers'. In: *International journal of theoretical physics* 21.6/7 (1982), pp. 467–488 (cited on page 3).

[4] Kyle Cormier, Riccardo Di Sipio, and Peter Wittek. 'Unfolding measurement distributions via quantum annealing'. In: *Journal of High Energy Physics* 2019.11 (Nov. 2019). DOI: 10.1007/jhep11(2019)128 (cited on pages 3, 15, 22, 25, 26, 42, 50).

[5] E Kou et al. 'The Belle II Physics Book'. In: *Progress of Theoretical and Experimental Physics* 2019.12 (Dec. 2019). 123C01. DOI: 10.1093/ptep/ptz106 (cited on pages 3, 28, 30, 32).

[6] T. Abe et al. *Belle II Technical Design Report*. 2010 (cited on pages 3, 32).

[7] T. Kuhr et al. 'The Belle II Core Software'. In: *Computing and Software for Big Science* 3.1 (Nov. 2018). DOI: 10.1007/s41781-018-0017-9 (cited on page 4).

[8] The Belle II Collaboration. *Belle II Analysis Software Framework (basf2)*. https://github.com/belle2/basf2/. 2021 (cited on page 4).

[9] Carlos Pegueros. *Unfolding Benchmark*. https://github.com/peguerosdc/unfolding_benchmark. 2021 (cited on page 4).

[10] G. Cowan. *Statistical data analysis*. Oxford University Press, USA, 1998 (cited on pages 6, 13).

[11] Per Christian Hansen. *Discrete Inverse Problems*. Society for Industrial and Applied Mathematics, 2010 (cited on page 13).

[12] Andreas Höcker and Vakhtang Kartvelishvili. 'SVD approach to data unfolding'. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 372.3 (1996), pp. 469–481. DOI: https://doi.org/10.1016/0168-9002(95)01478-0 (cited on pages 15, 17, 19–21, 42, 45, 47, 51).

[13] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering*. Cambridge University Press, Jan. 2019 (cited on page 17).

[14] '25. Practical Analysis of Least Squares Problems'. In: *Solving Least Squares Problems*, pp. 180–198. DOI: 10.1137/1.9781611971217.ch25 (cited on page 20).

[15] Mikael Kuusela. 'Statistical Issues in Unfolding Methods for High Energy Physics'. In: 2012 (cited on pages 21, 51).

[16] D-Wave. *Solving Problems with D-Wave Solvers*. https://docs.dwavesys.com/docs/latest/c_gs_3.html. 2021 (cited on pages 22, 50).

[17] Tadashi Kadowaki and Hidetoshi Nishimori. 'Quantum annealing in the transverse Ising model'. In: *Phys. Rev. E* 58 (5 Nov. 1998), pp. 5355–5363. DOI: 10.1103/PhysRevE.58.5355 (cited on page 22).

[18] Sorin Istrail. 'Statistical Mechanics, Three-Dimensionality and NP-Completeness: I. Universality of Intractability of the Partition Functions of the Ising Model Across Non-Planar Lattices'. In: *32nd ACM Symposium on the Theory of Computing (STOC00)*. Portland, Oregon: ACM Press, 2000, pp. 87–96 (cited on page 23).

[19] M. Born and V. Fock. 'Beweis des Adiabatensatzes'. In: *Zeitschrift für Physik* 51.3-4 (Mar. 1928), pp. 165–180. DOI: 10.1007/bf01343193 (cited on page 23).

[20] Andrew Lucas. 'Ising formulations of many NP problems'. In: *Frontiers in Physics* 2 (2014). DOI: 10.3389/fphy.2014.00005 (cited on page 25).

[21] Kyle Cormier, Riccardo Di Sipio, and Peter Wittek. *Unfolding as quantum annealing*. https://github.com/rdisipio/quantum_unfolding. 2019 (cited on page 25).

[22] Roman Marto ňák, Giuseppe E. Santoro, and Erio Tosatti. 'Quantum annealing by the path-integral Monte Carlo method: The two-dimensional random Ising model'. In: *Phys. Rev. B* 66 (9 Sept. 2002), p. 094203. DOI: 10.1103/PhysRevB.66.094203 (cited on pages 26, 54, 55).

[23] Shinya Morino. *SQAOD*. https://github.com/shinmorino/sqaod/. 2019 (cited on pages 26, 55).

[24] P.A. Zyla et al. 'Review of Particle Physics'. In: *PTEP* 2020.8 (2020), p. 083C01. DOI: 10.1093/ptep/ptaa104 (cited on pages 28, 29, 32, 33, 38).

[25] Giovanni Punzi. 'Sensitivity of searches for new signals and its optimization'. In: *Proceedings of PHYSTAT2003: Statistical Problems in Particle Physics, Astrophysics, and Cosmology* (Sept. 2003) (cited on page 40).

[26] Springer Verlag GmbH, European Mathematical Society. *Kolmogorov–Smirnov test. Encyclopedia of Mathematics.* Website. URL: http://encyclopediaofmath.org/index.php?title=Kolmogorov%E2%80%93Smirnov_test&oldid=22660. Accessed on 2021-10-27 (cited on page 45).

[27] Matthias Troyer. 'Simulated annealing versus quantum annealing'. In: *APS March Meeting Abstracts*. Vol. 2016. APS Meeting Abstracts. Jan. 2016, B13.001 (cited on page 52).

[28] Sei Suzuki. 'A comparison of classical and quantum annealing dynamics'. In: *Journal of Physics: Conference Series* 143 (Jan. 2009), p. 012002. DOI: 10.1088/1742-6596/143/1/012002 (cited on page 52).

[29] W. Greiner et al. *Thermodynamics and Statistical Mechanics*. Classical theoretical physics. Springer-Verlag, 1995 (cited on page 54).

[30] Inria. *Machine learning in Python with scikit-learn [MOOC]*. https://www.fun-mooc.fr/en/courses/machine-learning-python-scikit-learn/. 2021 (cited on page 67).

[31] Jerome H. Friedman. 'Greedy function approximation: A gradient boosting machine.' In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451 (cited on page 67).