Daniel Silva Pinheiro

Bachelor Degree in Physics Engineering

A Double Crystal Spectrometer for High Precision X-Ray Spectroscopy of Fundamental Physics and Applications

Master Thesis in
Physics Engineering

Adviser: Prof. Doctor Pedro Amaro, Assistant Professor, NOVA University of Lisbon

Co-adviser: Prof. Doctor Mauro Guerra, Assistant Professor, NOVA University of Lisbon

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

December, 2020

A Double Crystal Spectrometer for High Precision X-Ray Spectroscopy of Fundamental Physics and Applications

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Abstract

The European X-ray Spectrometry Association established in 2008 the Fundamental Parameter initiative with the aim of improving the measurements of fundamental X-ray parameters. In detail, the goal of this initiative is to improve the reliability and precision of atomic parameters related with the interaction of X-Ray radiation with matter. Aligned with this initiative, a double crystal spectrometer (DCS) is being built at the LIBPhys-UNL group, which will be able to measure fundamental parameters with competitive precision and reliability.

In preparation for this high-precise data, this thesis is dedicated to the complete simulation of the spectrometer system. With such a complex system and the need to obtain the most precise and reliable results, a simulation with different layers of physics is demanded. These layers would be for example, the physical processes of X-ray interaction, or the geometry of the DCS. To address this task, several simulations were built or modified not only from general particle tracking simulators, like Geant4 (with ROOT) but also more simple home-made simulations (done in FORTRAN). The simulations were tested for the physical processes used, guaranteeing that these simulations are ready to be configured for the final experimental setup. Further investigation on quantum processes that could be observed in experimental spectra was also performed, namely quantum interference and angular distribution, with the aim of further improving the precision of future measurements.

Keywords: Double Crystal Spectrometer, X-Ray Fluorescence Spectroscopy, Fundamental Physics, High Precision, Geant4, Synthetic Spectra.

Resumo

A Associação Europeia de Espetrometria de raios X estabeleceu em 2008 a iniciativa de Parâmetros Fundamentais com o objetivo de melhorar as medições de parâmetros fundamentais de raios X. Em pormenor, o objetivo desta iniciativa é melhorar a fiabilidade e precisão dos parâmetros atómicos relacionados com interações entre raios X e matéria. Alinhado com esta iniciativa, um espetrómetro de duplo cristal (DCS) está a ser construído no grupo LIBPhys-UNL, e terá a capacidade de medir parâmetros fundamentais com precisão e fiabilidade suficiente.

Em preparação para estes dados de alta precisão, esta tese é dedicada à simulação completa do sistema do espetrómetro. Tendo em conta a elevada complexidade do sistema e a necessidade de obter os resultados mais precisos e fiáveis, requer-se uma simulação com várias camadas de física. Estas camadas seriam por exemplo, os processos de interação dos raios X, ou a geometria do DCS. Para abordar esta tarefa, várias simulações foram construídas ou modificadas não só utilizando simulações gerais de rastreamento de partículas, como o Geant4 (utilizando o ROOT) mas também com simulações caseiras mais simples (feitas em FORTRAN). Os processos físicos implementados nestas simulações foram testados, garantindo que as simulações estão prontas para serem configuradas para o sistema experimental final. Investigações adicionais relativamente a processos quânticos que poderão ser observados nos espetros medidos também foram feitas, nomeadamente interferência quântica e distribuição angular, com o objetivo de melhorar a precisão de medições posteriores.

Palavras-chave: Espetrómetro de Duplo Cristal, Espetrometria de Fluorescência de raios-X, Física Fundamental, Alta Precisão, Geant4, Espetros Sintéticos

Contents

List of Figures

List of Tables

Listings

Acronyms

BEB Binary Encounter Bethe.

CPU Central Processing Unit.

CSF Configuration State Functions.

DCS Double Crystal Spectrometer.

FAC Flexible Atomic Code.

MCDF Multi Configuration Dirac Fock.

MCDFGME Multi Configuration Dirac Fock and General Matrix Element.

MRBEB Modified Relativistic Binary Encounter Bethe.

QED Quantum Electrodynamics.

QI Quantum Interference.

XOP X-Ray Oriented Programs.

XRF X-ray Fluorescence Spectroscopy.

CHAPTER 1

Introduction

Since the discovery of X-rays in 1895 by Röntgen [1], this radiation has been heavily employed in both practical and fundamental investigations. One of the most important and distinct properties is its much higher transmission in matter, compared to visible light. Several systems and techniques that employ this radiation were promptly developed, and in 1935 Compton and Allison presented a complete description of X-Ray radiation and its applications in spectroscopy [2].

With the improvement of technology over the decades, the spectroscopy techniques described by Compton and Allison were significantly improved, particularly in respect to their resolution and precision. However, these great improvements often revealed inaccuracies that require more reliable and precise values of several parameters related with the interaction of the X-Ray radiation with matter. With this problem in mind, in 2008 during the European X-Ray Spectrometry conference (EXRS2008) in Cavtat it was pointed out that there was a noticeable lack of recent, reliable values of atomic parameters related to X-ray interactions with matter [3]. Particularly a lack of values with sufficiently low associated uncertainties, responding to the advances in technology. These atomic parameters are often known as fundamental parameters and are required in multiple applications across multiple fields, ranging from medical applications and environmental control to planetary exploration [4]. Although there are many studies regarding this problem, there is still a need for more data, either due to high uncertainties in the values already reported or incomplete data sets. This problem limits the applications, reliability and precision of X-Ray based technology.

1.1 State of the Art

During the last century, various spectroscopy techniques were developed. These techniques can be separated into two methods of measuring the radiation: energy dispersion and wavelength dispersion. The wavelength dispersion method was the first method to be implemented, using Bragg's law of diffraction as basis. In this method the X-ray radiation is geometrically separated according to the wavelength. Instruments using this method were already commercially available in the 1940's [5].

The energy dispersion method was made possible later with the development of the semiconductor or Si(Li) detector in 1965. In this method the energy of the X-ray radiation is absorbed by the semiconductor, inducing a charge proportional to its energy, which is then amplified and measured by electronic devices. The energy dispersive method is more frequently used in laboratories, as all of the energy distribution can be detected faster and simultaneously, usually at the cost of resolution [6].

An early example of a high precision wavelength dispersion spectrometer was described in the book of X-Ray radiation and its applications in spectroscopy written by Compton and Allison in 1935. This spectrometer, named as the double crystal spectrometer, uses two crystals to filter and scan the radiation making use of the wavelength dispersion properties of the crystals' lattice as described by Bragg's law. This spectrometer was presented with many advantages as being able to perform high precision and reference-free measurements, but it also had many drawbacks, mainly due to the limitations of technology at the time. These limitations were primarily due to the angular resolution in rotating the crystals that were overcome with the introduction of high resolution angular encoders [7]. Nowadays, double crystal spectrometers are used for high precision, reference-free measurements of specific electronic transition energies and level widths [8]. The attainable resolution and precision is in the range of Quantum Electrodynamics (QED) corrections for some atomic systems[8, 9].

Recently, several experiments are pointing to imprecisions and discrepancies in the fundamental parameters. For example, one of the most recent problems is the proton radius puzzle, where several experiments in muonic hydrogen point to a discrepancy in the value for the proton radius [10, 11], in comparison to the CODATA value [12]. Additional discrepancies in other fundamental parameters were also described in [4], measured with crystal spectrometers such as the double crystal spectrometer. These parameters were line energies and shapes, fluorescence yields, mass absorption coefficients and others.

Following the last observations mentioned and others described in [4], the Fundamental Parameters initiative was established at the European X-ray Spectrometry conference of 2008, with the aim of improving the measurements of fundamental parameters. In line with this initiative, a high-precision double crystal spectrometer (DCS) is being constructed at the LIBPhys-UNL group [13]. This spectrometer will have enough precision to improve the current values of several fundamental parameters and will allow for reference-free measurements [3, 4]. The geometry of the DCS allows for the self-consistent measurement

of a reference value, without the need for external calibration such as standard reference materials, providing highly reliable values. Additionally, measurements of line energies and shapes made with the DCS will ensure an accuracy of a few parts per million in the soft X-Ray regime [9, 14], where a greater lack of measurements is noted [4].

The purpose of the future spectrometer at LIBPhys-UNL is to measure the scattered X-ray radiation from samples in order to retrieve highly accurate X-Ray fundamental parameters, to a level where chemical shifts can be discerned. Additionally, with its high resolution, the DCS will be capable of characterizing rare earths in geological samples [15], which is difficult with other techniques due to the their typically low concentration and especially due to their complicated line shapes that usually overlap one another. For the purpose of calculating the fundamental parameters, a complete description of the physical processes of X-Ray interactions with matter throughout the spectrometer is needed. Nowadays, there are several packages which provide a framework for simulating these processes, such as Geant4 [16], MCNP [17], XMI-MSIM [18] and PENELOPE [19]. Using these packages, the geometry of systems and the tracking of particle interactions can be easily developed. In this work, the package Geant4 is used for tracking of the X-Rays and their interactions with the material sample.

Regarding the current precision and resolution that can be achieved with a DCS, some additional interesting questions are raised, such as the effect of quantum interference (QI) [20] or the angular distribution of the emitted radiation in the measured spectra. Similar to the case of Raman or Rayleigh scattering [21], it is expected that fluorescent paths sum after ionization can quantum interfere with each other leading to corrections of the (often) used incoherent sum of paths. These questions require a more theoretical approach, namely the calculation of synthetic spectra and selection of the lines that can be involved in the QI phenomenon. In the case of the angular distribution of the emitted radiation, they have to be calculated and applied to the total spectra to assess their impact.

Concerning calculations of atomic structure, there are various atomic structure codes which can provide reliable values to construct the synthetic spectra, such as MCDFGME initially developed by by J. P. Desclaux and P. Indelicato [22], FAC developed by M. F. Gu [23] and Grasp2K developed by P. Jönsson et.al [24].

1.2 Objectives

The main goal is to build an analysis "assembly line" from first principles to the simulation of the final spectrum of the DCS. To achieve this, the first main objective of this work is to modify an already existing Monte Carlo simulation for the DCS, to generate the energy according to an X-ray fluorescence spectrum. The second objective is to test and use a previously built Geant4 simulation to generate the line energy and shape for the DCS Monte Carlo simulation. Finally the last main objective is to perform theoretical calculations, using programs such as MCDFGME [25] or specific calculations from first principles, to better understand the physics behind the obtained measurements, namely

if the influence of quantum interference [20] or polarization can be observed in the DCS measurements.

The thesis is organized as follows:

- The next chapter 2 addresses the simulation of the DCS and the theoretical concepts required for these simulations.
- Chapter 3 addresses the atomic structure calculation codes and theory used in the synthetic spectra calculated in this work.
- In chapter 4 the results from testing and modifying the simulations are presented
- In chapter 5 the calculated synthetic spectra are shown, in addition to other interesting effects described in chapter 3
- Chapter 6 summarises the conclusions of this work and describes the future perspectives in using these simulations

CHAPTER 2

Simulation of the Spectrometer

In this chapter the general layout of the simulations will be explained, in addition to a more detailed explanation of each simulation and packages used. Figure 2.1 is a general layout of the spectrometer, and with the various layers of simulations that are required.

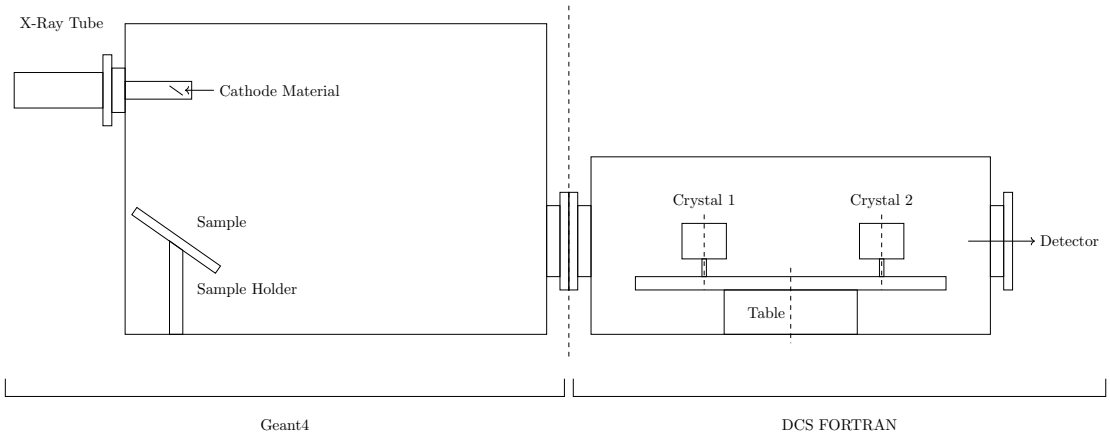


Figure 2.1: General schematic of the double crystal spectrometer setup. The first stage, on the left, will be simulated using the Geant4 package and corresponds to an X-Ray fluorescence spectrometer in planar geometry. The second stage, on the right, is the double crystal spectrometer itself and will be simulated using the DCS code.

This system has two vacuum chambers, corresponding to two stages of the setup. The first stage is the X-Ray fluorescence (XRF) spectrometer in planar geometry, which will be simulated using the Geant4 package (see section 2.3.2), consisting of the X-Ray tube and sample. Both the X-Ray tube's cathode and the sample are placed at a 45° angle, but the sample's angle can be changed. The fluorescence radiation emitted from the sample is then analysed by the DCS in the second stage. This stage will be simulated by the DCS code. In this setup the crystals are mounted on top of a rotating table and each crystal

can also rotate about their own vertical axis. Both the Geant4 package and the DCS code are based on the Monte Carlo method that is briefly described next.

2.1 Monte Carlo Method

The simulations that are developed and tested in this work are based in the Monte Carlo method to model physical phenomena that has a statistical nature [26]. Therefore, this method is often used in physical or mathematical problems, especially in those that an analytical solution is extremely difficult or even impossible to compute. Additionally, Monte Carlo methods can be applied to more general cases, e.g. in cases where the model structure changes, and are more efficient for large-scale tasks when there are a large number of elements to be simulated [26]. This method is extensively used in particle tracking applications [27], such as the ones studied in this work.

A simple Monte Carlo simulation is shown in figure 2.2 for computing the area shown in blue. This method is relatively easy to implement and theoretically easier than an analytical approach, although it requires more time and computational resources.

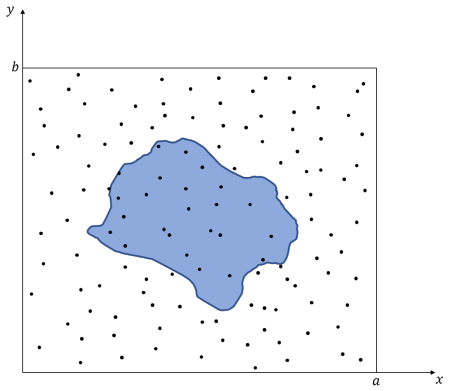


Figure 2.2: Visualisation of the generated points from a Monte Carlo simulation, which is used to estimate the value of the area in blue. a and b are arbitrary values that delimit the area where the points are generated.

By randomly generating points in the square and count the number of points inside the area, the fraction of inside points by the total points gives an approximation of the fraction of the inside area to the total (square) area. The accuracy of this approximation gets better with higher number of events.

2.2 Probability Distributions

Computationally, one can obtain a random variable from an arbitrary underlying distribution by various methods. The modifications that will be described in chapter 4 are based on concepts related to probability distributions.

As the arbitrary input is given by an histogram, the method chosen was rejection sampling. To use rejection sampling in a continuous manner, the histogram is interpolated by cubic splines. An appropriate area for generating the random variable is then chosen. Inside this area, a random point with coordinates (x,y) is generated, using uniform distributions. If this point falls bellow the spline interpolation curve, then the random variable will take the value of x. If the point falls above the spline interpolation, then it is rejected and a new point is generated until it is accepted as a value for the random variable [28]. In this manner a random variable which follows an arbitrary input probability distribution is generated (figure 2.3).

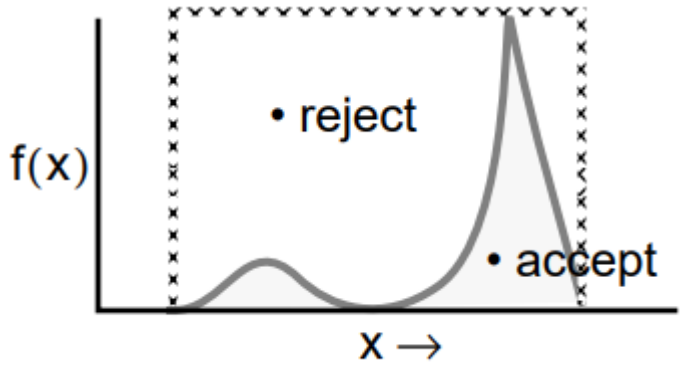


Figure 2.3: Graphical representation of an arbitrary input distribution and the selected area. Examples of one rejected point and one accepted point are also represented [28].

2.3 Planar XRF Simulation: Geant4, ROOT

In this section the Geant4 and ROOT packages will be presented, as well as the necessary theoretical concepts to understand the Geant4 package's operations. Concepts regarding the planar XRF spectrometer geometry and operation will also be presented. These packages are used to simulate the first stage of the setup outlined in figure 2.1.

2.3.1 Planar X-ray Fluorescence Spectrometer

The planar XRF spectrometer is a regular laboratory XRF spectrometer with a planar geometry. This spectrometer is used in the first stage of the DCS (Geant4 stage in figure 2.1). The general layout of an XRF spectrometer is represented in figure 2.4, where the rectangle on the right represents the source from which X-Ray radiation is emitted. This radiation interacts with the sample, from which fluorescence X-Rays are emitted. The detector is placed at a certain angle in relation to the source and detects fluorescence radiation with an energy dependent efficiency. The detector is usually a sensitive volume of silicon. All the geometry is contained in a plane, that in the case of the layout below, is the plane of the sheet itself. The obtained spectrum will contain a Bremsstrahlung background, Compton and Rayleigh peaks and the characteristic peaks of the sample's

elements. Through analysis, the energy and intensity of the characteristic peaks (element specific) the sample's elementary composition can be obtained.

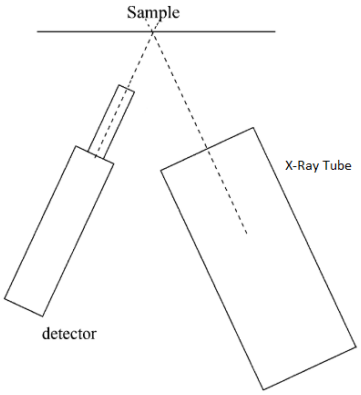


Figure 2.4: General layout of a planar XRF spectrometer. The detector is represented by the rectangle on the left and the X-ray tube is represented by the rectangle on the right. Adapted from [29].

Using an experimental spectrum from a known reference sample, the simulation's predictions of the experiment can be verified. Some discrepancies between simulation and experiment are usually observed due to unknown experimental values, like the exact thickness of the sensitive volume of the detector, or the approximations implemented in the optimization of the simulation. These values need to be optimized to better describe the experimental setup.

2.3.2 Geant4

Geant4 is a Monte Carlo simulation package, which can be used to simulate any geometry involving any radiation-matter interactions. It is better suited for high energy radiation (energies above 1 GeV), however, several recent improvements for lower energies, (particularly fluorescence) have recently been added [30]. In this work, the Geant4 physics lists for photon particles which set the interactions were, `G4LivermorePolarizedPhotoElectricModel` (photoelectric effect), `G4LivermorePolarizedComptonModel` (Compton scatering) and `G4LivermorePolarizedRayleighModel` (Rayleigh scatering). For electrons the `G4eMultipleScattering`, `G4eIonisation` and `G4eBremsstrahlung` interactions were used with default settings. In addition to the interactions used for electrons, in the case of positrons, the `G4eplusAnnihilation` was also included with default settings. The atomic deexcitation considers the fluorescence and auger processes.

This package allows the generation of particles with a certain amount of energy, starting from a specified position and direction, and tracks their path. The path of each particle is divided in steps proportional to the shortest interaction length of all the physical processes included in the simulation. Depending on the particles used, it also checks if any of them decayed before executing each step. The subsequent particles after decay or any created new particles are also tracked, similarly to the parent particle.

To improve the computation time, a cut value for the step length is also used, which means that for any steps with a length smaller than this cut value, the step ends in total absorption of the particle [31]. The respective particle track stops being tracked. If the step length is greater than the cut value, Geant4 checks if there has been an interaction, according to the probability (cross section) of the physical processes considered. According to this interaction, a change in energy or momentum is applied to the particle. If the process generates secondary particles, these particles are placed in the simulation and their tracking is done in the same manner as the primary particles. An example of a tracking path is represented in figure 2.5.

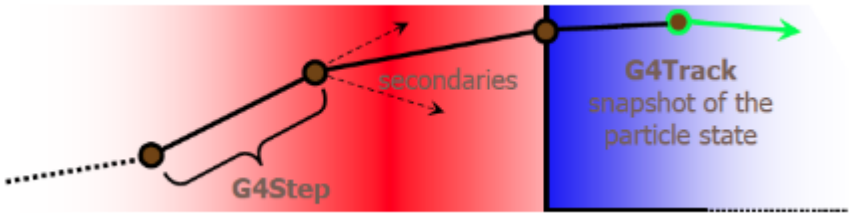


Figure 2.5: Graphical representation of a particle track in Geant4. At the end of each step, the particle can change direction and/or generate secondary particles. G4Step and G4Track are objects that describe the particle's current step and track, respectively [31].

The G4Step and G4Track are objects in the Geant4 package that describe the particle's current step and track, respectively [32, 33]. G4Track also provides a snapshot of the particle's current state. Each step in Geant4 is described by a G4Step object. The main properties that this object possesses are the starting point of the step, the end point of the step, the step length and the particle's deposited energy at the end of the step. The deposited energy is the energy transferred to the volume with which the particle interacts and is differentiated in total and non-ionizing energy deposits [32]. A representation of the step properties is depicted in figure 2.6.

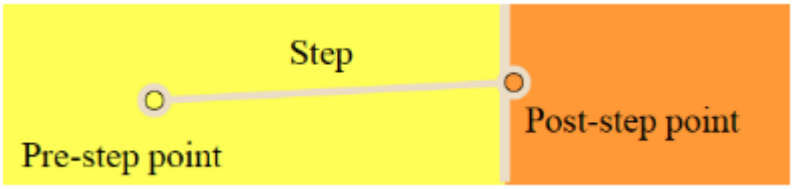


Figure 2.6: Graphical representation of a G4Step object. The step length, step starting point and step ending point are represented [31].

2.3.3 ROOT

The ROOT package is a modular scientific software toolkit, which can perform data processing on large amounts of data, statistical analysis, data visualization and storage

[34]. It is written in C++ but it is also integrated with other languages, specifically Python and R.

Given the data storage and processing capabilities of the ROOT package and the fact that it is also written in C++, it was chosen as a data manager for the Geant4 simulation. More specifically, it is being used to record and store the deposited energy in the various sensitive volumes (detectors) in the simulation. Additionally, the ROOT package is being used to sample an arbitrary input energy spectrum to generate the energy distribution of the experimental X-Ray tube inside the Geant4 simulation.

2.4 Bragg's Law

Bragg's law describes the reflected X-Rays after the irradiation of a crystal. If the reflected X-Rays have a difference in path equal to an integer number of the incident radiation's wavelength, there will be constructive interference, hence there is reflection (figure 2.7).

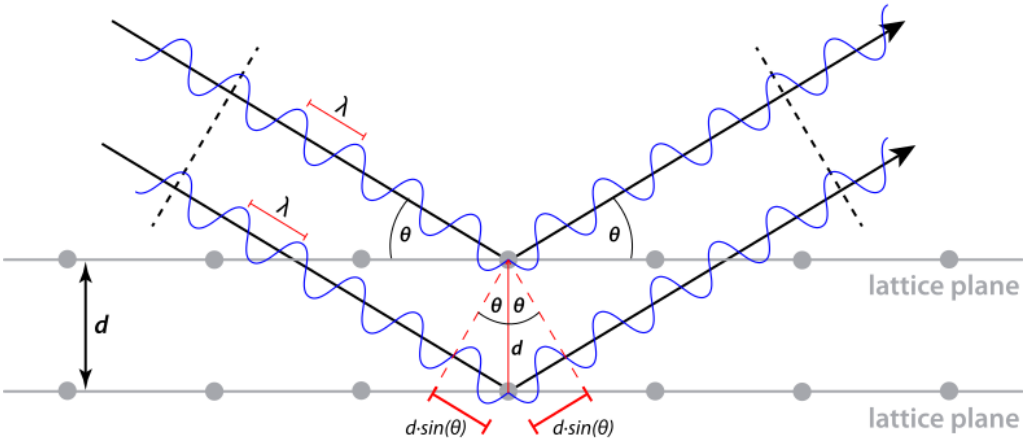


Figure 2.7: Depiction of Bragg's law for two incident radiation waves on two crystal lattice planes. The geometrical meaning of $d\sin(\theta)$ is also depicted [35].

The condition described above is translated in,

$$2d\sin(\theta) = n\lambda, \tag{2.1}$$

where d is the crystal lattice inter-planar distance, λ is the incident radiation wavelength, n is the diffraction order (integer number) and θ is the radiation's angle of incidence.

An important application of Bragg's law is to separate the incident X-Ray radiation by wavelength i.e. different wavelengths will be reflected along different angles. This is the principle behind how the double crystal spectrometer achieves high energy resolution.

This theory only predicts the position of the diffracted radiation as a single point, i.e. a dirac delta. In reality the diffraction response has a certain width and is not centered at the value of the Bragg angle due to the refraction of the incident radiation within the first layers of the crystal, as is shown in figure 2.10. In the case of the DCS, Bragg's theory is

extended (dynamic diffraction theory) to include all of these corrections to calculate the crystal response [36].

2.5 DCS Simulation: Fortran

In this section, the principles of the DCS, and their application in the DCS simulation, will be presented and briefly explained. The base code was built previously [37], but required some modifications to be used for this specific setup (second stage in figure 2.1). The changes made to this simulation will be discussed in chapter 4.

2.5.1 Double Crystal Spectrometer

A DCS has an X-Ray source as input, and through a double monocromatization using two crystals, the input energies can be scanned with high resolution. Due to the geometry of the system, a parallel and an anti-parallel configuration can be produced (figure 2.8) by the relative positions of the first and second crystals.

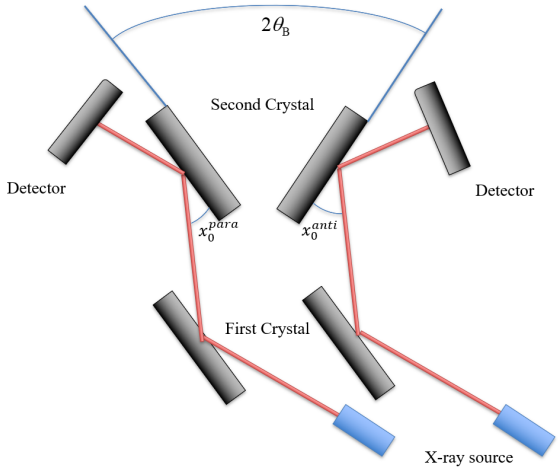


Figure 2.8: Example of a DCS scheme. The two configurations depicted are called parallel and anti-parallel from left to right, respectively [37]. θ_B is the Bragg angle of the specific energy being measured.

The resulting profile from the DCS measurements is represented as the number of events for each crystal angle. The parallel configuration is a non-dispersive configuration, because of the parallel relationship between the first and second crystals. This configuration produces a single peak in the output spectrum that serves as angular reference. On the other hand, the anti-parallel configuration is a dispersive configuration, which separates the incoming radiation by wavelength, producing a spectrum that is a measure of the input radiation's distribution. By using the spectra from both configurations, an experimental absolute value of energy can be obtained from the Bragg angle calculated using the second crystal angles, represented in the figure (equations 2.2 and 2.3).

$$\theta_{\mathrm{B}} = \frac{\left[180 - (x_0^{\mathrm{anti}} - x_0^{\mathrm{para}}) - 2\Delta_{\mathrm{r}} - \left(\xi_1^{+} - \xi_1^{-}\right)\right]}{2}, \tag{2.2}$$

$$E = \frac{hc}{2d\sin(\theta_{\mathrm{B}})}, \tag{2.3}$$

where θ_{B} is the approximate Bragg angle, x_0^{anti} is the angular position of the anti-parallel peak, x_0^{para} is the angular position of the parallel peak, Δ_{r} is a correction due to the index of refraction and the last two terms are related to vertical geometric systematic errors. E is the energy corresponding to the Bragg angle, d is the crystal's inter-planar distance and hc is the plank's constant multiplied by the speed of light to convert wavelength into energy [37]. In the case of converting the anti-parallel profile full angular axis to an energy axis, the x_0^{anti} corresponds to each angular position.

Using this system to acquire spectra and using the previous equations, we can obtain reference-free measurements of input spectra with high resolution due to the double monocromatization.

Although this spectrometer produces very precise and reference-free measurements, its potential was only reached with the introduction of high resolution angular encoders, due to the need of high resolution and stable angular measurements. For example, the original DCS in Laboratoire Kastler Brossel (Paris), for which this simulation was implemented, had a stepping motor to control the rotation of the crystals with a resolution of 0.017" and the encoders for the first and second crystal had a resolution of 0.07" and 0.2" respectively [37].

2.5.2 Simulation Description

The DCS simulation considers a setup as described in figure 2.9. In this setup, the distances between elements (X-Ray source, crystal, detector) are read from an external file, and both dispersive and non-dispersive configurations can be simulated at the same time. The crystals can also be simulated as plane or curved surfaces, which will affect the lattice spacing, depending on the distance of the interaction point to the crystal center.

The simulation generates as many X-Ray events as specified by the user, and for each one it randomly generates an energy around a center line (specified by the user). This random energy is generated using a Gaussian or Lorentzian distribution, with a user given line width. The point and angle at which the ray starts from is also randomly generated, depending on the source type chosen (point source, circular uniform or rectangular uniform). A collimator can also be included after the X-Ray source. The starting angle of the radiation will also depend on the chosen collimator's geometry in order to not generate X-rays with trajectories that would never hit the first crystal, thus saving computational time. Between the elements, the X-Rays travel in a straight line, as the system is in vacuum. When they reach one of the crystals, its experimental reflection response (figure 2.10) is used as the probability of reflection, and geometrical corrections are applied depending on the

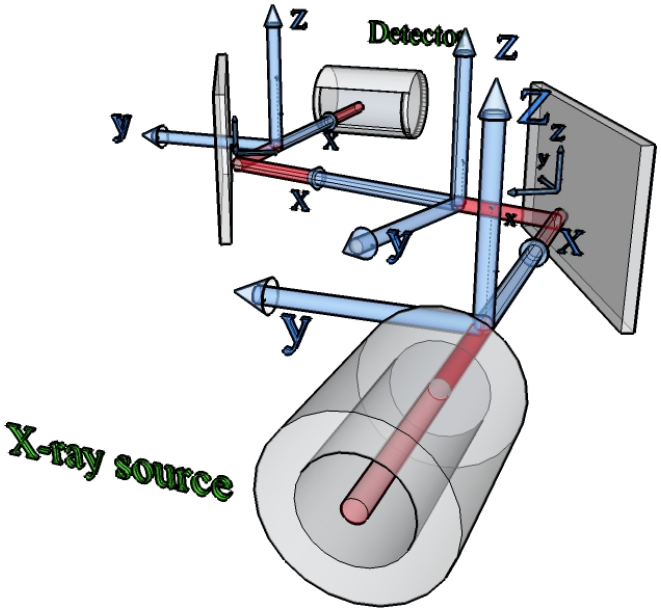


Figure 2.9: Representation of the X-Ray source, crystals and beam used in the simulation. The relative distances can be configured externally and the crystals can be simulated as plane crystals or with a curved surface. Both the X-Ray beam's travel direction and the crystals' surface normals are along the x-axis [37].

crystals' geometry (flat or curved crystal). This probability is given as the probability of reflection of a ray with a given difference of its incident angle with respect to the Bragg angle calculated from the ray's energy. The detector simply counts all X-Rays that reach it for each rotation, in both configurations. A gas filled proportional counter detector is used in the DCS in Laboratoire Kastler Bossel (Paris) [37].

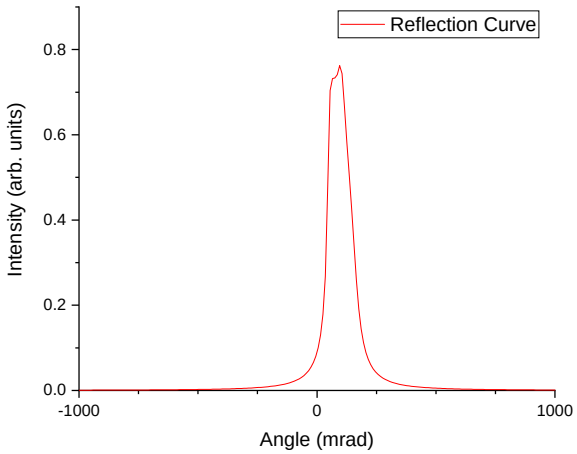


Figure 2.10: Graphical representation of the crystal's reflection response in terms of the difference between the radiation's incident and Bragg angles. Reflection response curves can be experimentally measured, but can also be theoretically calculated with programs as the X-Ray Oriented Programs (XOP) [36].

The crystal reflection response is usually given as a function of the X-ray incidence angle minus the corresponding Bragg angle (section 2.4), which corresponds to the Angle-axis in figure 2.10. This response is usually obtained from dynamic diffraction theory [36], which is an extension of Bragg's law. The prediction for this response using only Bragg's law would be a dirac delta centered at 0.

CHAPTER 3

Atomic Structure Calculations

In this chapter, various theoretical concepts required to understand the atomic structure codes used are described. The atomic structure codes used are the Multi Configuration Dirac Fock and General Matrix Element (MCDFGME) code [22] and the Flexible Atomic Code (FAC) [23].

3.1 Multi Configuration Dirac Fock and General Matrix Element - MCDFGME Code

The MCDFGME code was designed for computing atomic parameters related with matrix elements such as energies, various rates (Auger rates, Radiative rates), cross sections (photoionization cross sections), as well as hyperfine splittings, to name a few [25]. This description will follow mostly the webpage of this code [38] with complementary bibliography sources. Unless explicitly mentioned, all equations are in atomic units.

3.1.1 MCDF Method

This method uses a self-consistent field approach in which each electron constitutes a central-field, one electron wavefunction. An effective Hamiltonian operator is built using the potential fields generated by the wavefunctions from all the electrons. An iterative process is performed on the Hamiltonian operator and the electron wavefunctions, alternating, resulting in an improvement of the Dirac solutions [39]. The resulting energy is then corrected using contributions from QED, such as vacuum polarization and self-energy.

3.1.1.1 Electron Wavefunction

To include relativistic contributions, the wavefunctions used are the Dirac four-component spinors [40]. These spinors represent the full one electron wavefunction with the respective radial, angular and spin components. They are written as [38]:

$$\Phi_{nkm}(r,\theta,\varphi) = \frac{1}{r}\begin{Bmatrix} P_{nk}(r)\chi_{km}(\theta,\varphi) \\ iQ_{nk}(r)\chi_{-km}(\theta,\varphi) \end{Bmatrix}. \tag{3.1}$$

The quantum numbers n, k, m are the principal quantum number, the relativistic quantum number, and the projection of the total angular momentum along the z axis, respectively. The quantum number k includes the parity and the total angular momentum. These spinors are eigenfunctions of the parity operator, total angular momentum operator ($\boldsymbol{j} = \boldsymbol{l} + \boldsymbol{s}$) and its projection along the quantization (z) axis (j_z). The χ_{km} functions in equation (3.1) are two-component Pauli spherical spinors [41].

The Dirac spinors are then used to construct Configuration State Functions (CSF) which describe a specific configuration for the system in question, as well as to assure the Pauli exclusion principle. These CSF's correspond to the atomic wavefunction of a pure state (using the jj coupling scheme as basis). The full MCDF wavefunction is constructed as a superposition of all the CSF's, i.e. a linear combination of all the determined CSF's [38]:

$$\psi = \sum_{\nu=1}^{NCF} W_\nu \phi^\nu(J, M_J). \tag{3.2}$$

Where W_ν is the weight of the ν-th configuration identified by the ψ wavefunction's eigenvalue. Each $\phi^\nu(J, M_J)$ corresponds to the CSF, calculated by the antisymmetric product (Slater determinant) of the one electron wavefunctions, represented by the Dirac four-component spinors in equation 3.1. The weights W_ν are calculated taking into account the value of the $eigv$ parameter [22]. NCF is the total number of configurations.

3.1.1.2 Hamiltonian

The Hamiltonian used is the Dirac Breit Hamiltonian, which is written as [38]:

$$H_N^{DB} = \sum_{i=1}^{N} h_i^D + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} h_{ij}^{C+B}. \tag{3.3}$$

Here the h_i^D operator is the Dirac operator that describes the dynamics of an electron in the nuclear potential, for each one of the one-electron wavefunctions, including relativistic effects. The h_{ij}^{C+B} operator is the Coulomb plus Breit operator, where the latter describes the electron-electron relativistic interaction in the system. N is the number of electrons in the system.

The Dirac operator is written as:

$$h_i^D = c\boldsymbol{\alpha}.\boldsymbol{p} + (\boldsymbol{\beta} - 1)c^2 + V_N(r_i), \tag{3.4}$$

where $V_N(r_i)$ is the electron-nucleus interaction potential, c is the speed of light, $\boldsymbol{p}$ is the linear momentum operator and $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the 4×4 Dirac matrices given by:

$$\boldsymbol{\alpha} = \begin{pmatrix} 0 & \boldsymbol{\sigma} \\ \boldsymbol{\sigma} & 0 \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{3.5}$$

where $\boldsymbol{\sigma}$ are the Pauli matrices, whose components are represented by:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3.6}$$

The Coulomb plus Breit operators are given by:

$$h_{ij}^{C+B} = \frac{1}{r_{ij}} + B(i,j), \tag{3.7}$$

where $1/r_{ij}$ is the Coulomb repulsion between each pair of electrons and $B(i,j)$ is the relativistic Breit operator. This last term includes the relativistic interaction between the electrons, namely the magnetic retardation due to the interaction of spins of a pair of electrons. The Breit operator is written as [42]:

$$B(i,j) = -\frac{1}{2r_{ij}} \left[\boldsymbol{\alpha}(i) \cdot \boldsymbol{\alpha}(j) + \frac{(\boldsymbol{\alpha}(i) r_{ij}) \cdot (\boldsymbol{\alpha}(j) r_{ij})}{r_{ij}^2} \right]. \tag{3.8}$$

This is the first order Breit operator, which is included in the self-consistent field. Higher order, frequency dependent, terms are included as perturbations to the wavefunctions.

3.1.2 Atomic Quantities

Besides the energy of the atomic system, the MCDFGME code calculates many other properties such as: radiative transition probability rates; Auger transition probability rates; photoionization cross sections; hyperfine structure constants; Landé factor; Born electron impact cross sections; Stark effect; parity non-conserving amplitudes; scalar product of total wave functions; Schiff moment; Magnetic part of the *g-2* corrections for antiprotons. MCDFGME also allows for the introduction of exotic particles in the system (e.g. muon), although only one per system is supported. Many of the properties that MCDF calculates are not used further in this work, and are thus not described here. The used properties are the radiative transition probability rates and Auger transition probability rates, used to calculate the emission of an atom and construct its synthetic emission spectrum that is presented in chapter 5.

3.1.3 Calculation of a Theoretical X-Ray Emission Spectrum

When an atomic system is ionized, it will rearrange itself in order to relax to the new lowest possible energy. The rearrangement is done through transitions of the electrons between atomic orbitals. This transition can result in either the emission of X-Rays, called radiative transition, or the ionization of another electron, called non-radiative transition, usually referred to as Auger transitions.

In figure 3.1 it is presented an overview of the notations used to identify atomic transitions. The correspondence between the quantum numbers used in MCDF and the experimental notations will also be helpful in comparing theoretical and experimental lines.

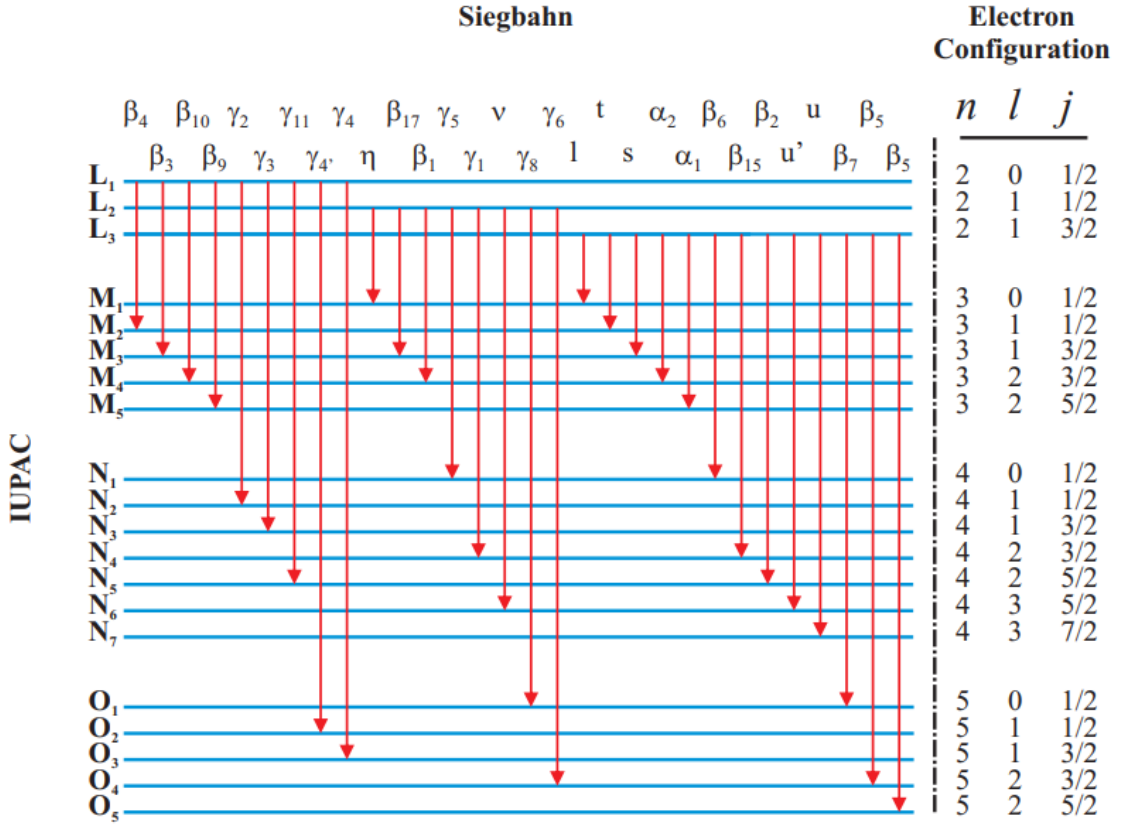


Figure 3.1: Correspondence between Siegbhan, IUPAC and *nlj* electron configuration notations for radiative transitions, where n is the principal quantum number, l is the orbital angular momentum, and j is the total angular momentum quantum number [43].

3.1.3.1 Transition Processes

In figure 3.2, the possible transition processes (radiative and radiationless) are represented, using the nickel electron configuration (neutral state) as an example. In the case of an hole being created in an orbital (initial state), the system can respond in two ways to reach the new lowest energy. By the means of a radiative transition (L_1N_1 in figure 3.2) or a radiationless transition (L_1VV in figure 3.2). For the case of radiationless transitions it

is common to use V to represent the valence orbital (in the example given in figure 3.2 $N_1 = V$) [44].

To calculate the full radiative emission spectrum for an element using MCDF, all possible radiative transitions need to be calculated. In addition to the radiative transitions, all possible non-radiative transitions should also be taken into account. This will improve the accuracy of the emission spectrum, as in an atom this process will occur, decreasing the emission of other radiative transitions.

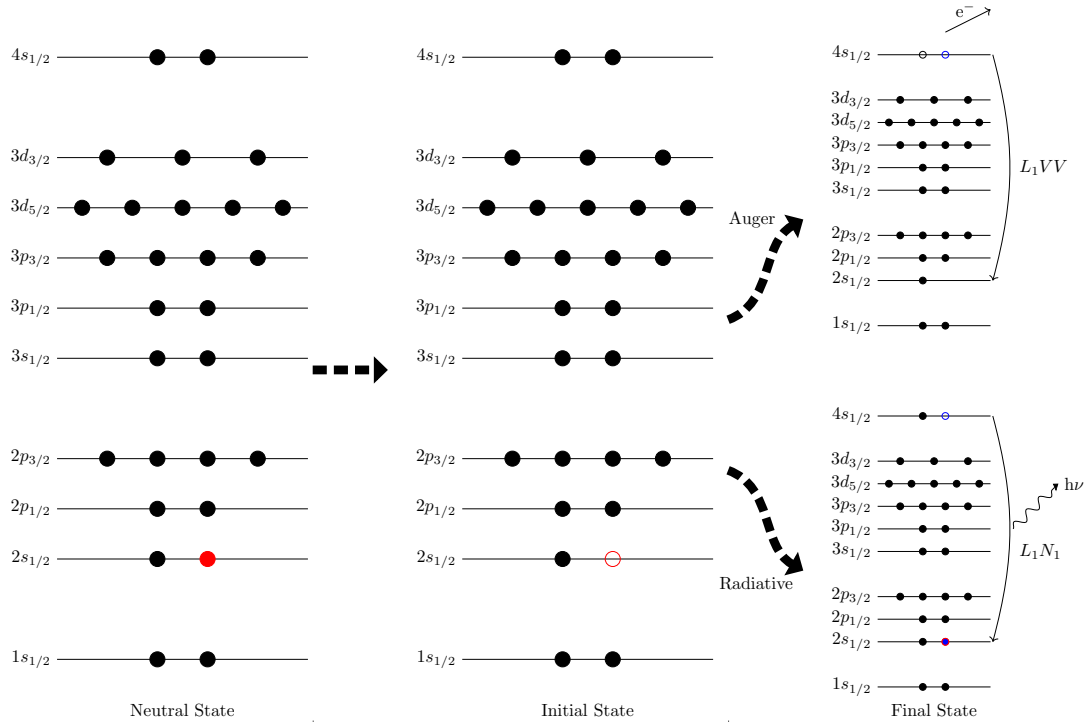


Figure 3.2: Example of a possible transition processes for Ni (Radiative - L_1N_1; Auger - L_1VV).

In the case of radiative emission, the calculation is done using the multipolar decomposition. As the emitted radiation is an electromagnetic wave, it can be decomposed by electric and magnetic multipoles. For each pole a calculation is performed for the respective transition rate and line width. Due to the nature of this decomposition, the electric dipole radiation is the most intense emission in most cases, followed by the magnetic dipole, electric quadrupole, etc.

These processes only take into account one-step transitions, i.e. only one transition is used in the process of filling the vacancy in the ionized orbital. Other processes can be considered, by subdividing the initial one-step transition into multiple transitions i.e. cascades between the initial and final states, improving the calculated spectrum. Moreover, two-electron-one-photon transitions were not included due to their very low probability in the studied systems [45].

3.1.3.2 Unique Transition Identification in MCDF

To individually calculate the properties of each transition, first the identification of the set of values that correspond to a unique transition is needed. In figure 3.3 the necessary values to identify an unique MCDF state are represented. This identification starts with the LS coupled electron configuration, the respective total angular momentum, J and its projection along the quantization axis, M_J. This information is then used to calculate the conversion to the possible jj couplings and generate the final atomic wavefunction, with a mixture of the jj couplings identified by the value of $eigv$. In the case of MCDF, the input is actually double of the angular momentum values, making the values be always integer numbers.

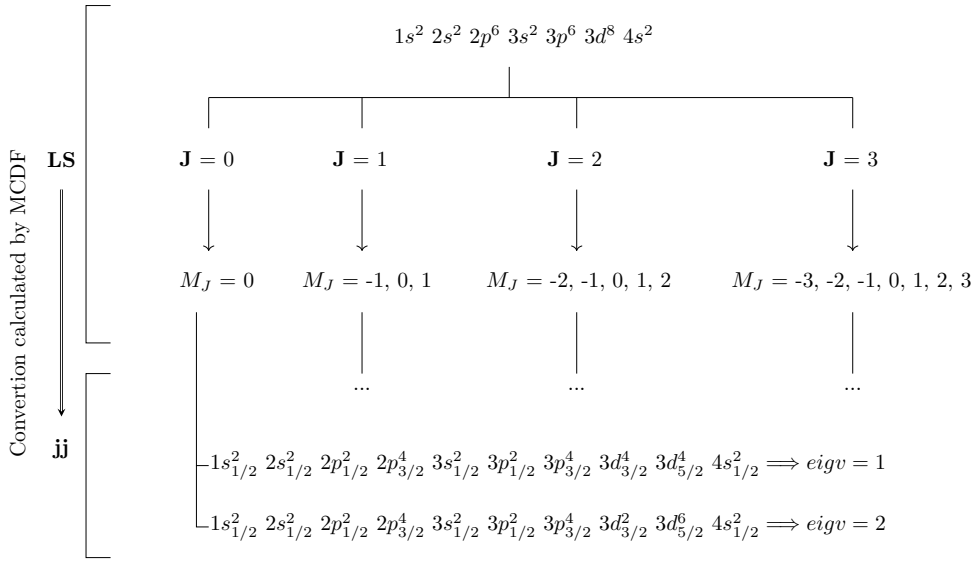


Figure 3.3: Schematic of the identification of the ground state configuration of Ni. Starting from the user input in the LS coupling bracket, MCDF calculates the respective, possible jj coupled configurations. The mixing of these configurations is then specified using the $eigv$ parameter.

In more detail, to generate the total wavefunction used in the MCDF code, the electron wavefunctions are needed. The electron wavefunctions depend on the electron's total angular momentum, j (spin-orbit coupling) and the respective projection (along the z-axis), m_j. To determine these quantum numbers (j and m_j), the jj coupled configuration of each state is needed. However, the MCDF code takes an LS coupled total atomic configuration and total atomic angular momentum J as input (e.g. for nickel the input for the ground electron configuration is $1s^2\ 2s^2\ 2p^6\ 3s^2\ 3p^6\ 3d^8\ 4s^2$), and then generates all the possible jj coupled configurations. To calculate the possible jj coupling from the LS coupling, the code also takes as input double of the associated total angular momentum (J) and double of the respective projection (along the z-axis M_J). The atomic wavefunctions (CSF's) can then be calculated from the determined electron wavefunctions.

The individual electron and total atomic wavefunctions (CSF's) are generated from the jj coupled configurations and the final wavefunction is given as a linear combination of the atomic wavefunctions. Only the configurations associated with a contribution of 0.01 % or higher are taken into account. The linear combination is not unique for a given input set (J and M_J), i.e. each of the calculated jj coupled configurations can be combined with different weights to obtain the input LS coupled configuration with its respective J and M_J. The specification of which jj coupled configuration the convergence of the calculation should be done towards, is also taken as an input parameter. This parameter ($eigv$ in figure 3.3) is yet another value, apart from the system's total angular momentum and its projection, that needs to be considered when generating all the possible initial and final state combinations.

In summary, a unique transition in a given atomic system (LS coupled configuration) is characterized by a unique set of 6 parameters, i.e. the total angular momentum, its corresponding projection and final wavefunction eigenvalue for both the initial and final states of the transition.

3.1.3.3 Line Intensity

From the values obtained from the atomic structure codes, for the transition rates, the intensity of each transition can be calculated using equation 3.9, multiplied with the ionization probability of the ionized subshell.

$$I_i = \frac{\Gamma_i^{\mathrm{R}}}{\Gamma_{S_n}^{\mathrm{R}} + \Gamma_{S_n}^{\mathrm{NR}}}, \tag{3.9}$$

where $\Gamma_{S_n}^{\mathrm{R}}$ and $\Gamma_{S_n}^{\mathrm{NR}}$ are the partial widths of the S_n-subshell corresponding to the radiative and radiationless transitions respectively [43]. Γ_i^{R} is the partial width of the state i belonging to the one-hole electron configuration of the S_n-subshell, i being the set of quantum numbers: $\{J_{\mathrm{initial}}, M_{J\mathrm{initial}}, eigv_{\mathrm{initial}}, J_{\mathrm{final}}, M_{J\mathrm{final}}, eigv_{\mathrm{final}}\}$. The $\Gamma_{S_n}^{\mathrm{R}}$ can be calculated as the sum of all the Γ_i^{R} of the line inside the S_n-subshell one-hole configuration. Considering the Heisenberg uncertainty principle, the partial widths can be replaced by the transition rates (W_i) of the corresponding or line i, $\Gamma_i^{\mathrm{R}} = \hbar W_i$. This rationale also applies to the radiationless quantities.

The output of the calculation performed using MCDF gives the values for the energy, probability and line width of the respective transition, which gives all the information necessary to construct the spectrum. The total intensity of each transition is obtained by multiplying the fluorescence yield by the probability of ionization of the respective initial orbital (this is further specifyed in chapter 5).

3.1.3.4 Quantum Interference Process

The Quantum Interference (QI) between two emitted photons is a process that can occur when certain conditions of angular momentum and parity are met. Although this process

has a small effect on spectra, due to the DCS's high resolution (0.6 eV in energy [37]) it will be observed in this spectrometer and is thus needed for a complete description.

The QI between two emitted photons can occur when both photons have the same angular momentum and parity (j and λ) [46]. Using the conservation of angular momentum and parity, both the initial and final atomic states of each transition must also have matching total angular momentum and parity. Additionally, the frequencies of the photons have to be similar, i.e. $\omega_1 \approx \omega_2$ where ω_1 and ω_2 are the frequencies of two resonances. The meaning of the approximate sign is given by the width of the emission distribution of the corresponding atomic transition. If photon 1 originates from a transition with a full width at half maximum (FWHM) Γ_1 and photon 2 from a transition with a FWHM Γ_2, these transitions can interfere with each other if $(E_2 - \Gamma_1/2 - \Gamma_2/2) \leq E_1 \leq (E_2 + \Gamma_1/2 + \Gamma_2/2)$. In this work the transitions that meet these conditions are identified in a synthetic spectrum, but the intensity of the interference is not calculated.

3.1.3.5 Angular Distribution of Dipolar Radiation

Angular and polarization properties [47] of atomic processes can be followed in the seminal review by Blum and Kleinpoppen done in 1979. There, several theoretical and experimental investigations are described, both performed in light elements like hydrogen, helium and lithium. These investigations show significant differences in terms of the angular distribution of the emission relatively to an isotropic emission. In 1986 [48], further experimental research was performed in Xenon atoms, showing significant differences relatively to isotropic emission. In 2000, Balashov, Grzhimailo and Kabachnik published a book [49] with extensive theory on this subject. Particularly, it was explained how the anisotropy of the angular distribution can be calculated for various cases, depending on the polarization of the radiation in question. In this case we are looking for the polarization independent detection of the emitted radiation, produced by electron impact ionization. This is given in the equation (3.36) of this book:

$$W_{\alpha_{\text{final}} J_{\text{final}}} = \frac{W_0}{4\pi} \left[1 + \alpha_2^{\gamma} A_{20}(\alpha_{\text{initial}} J_{\text{initial}}) P_2(\cos\vartheta)\right], \tag{3.10}$$

where α_{final} and J_{final} are the quantum numbers that identify the final state, α_{final} are the necessary quantum numbers apart from the total angular momentum J_{final}. Similarly, α_{initial} and J_{initial} are the quantum numbers of the initial state, i.e. ionized state before radiative emission. W_0 is the emission intensity of the transition and α_2^{γ} is the anisotropy parameter (α_k^{γ}). $A_{20}(\alpha_{\text{initial}} J_{\text{initial}})$ is the reduced statistical tensor $A_{kq}(\alpha_{\text{initial}} J_{\text{initial}})$ and $P_2(\cos\vartheta)$ is the second Legendre polynomial. This equation is obtained when using the multi-polar decomposition of radiation and considering only the first contribution, i.e. dipolar radiation.

The anisotropy parameter is given by:

$$\alpha_k^\gamma = \sqrt{\frac{3}{2}}\hat{J}(-1)^{J_{\text{initial}}+J_{\text{final}}+k+1}\begin{Bmatrix} J_{\text{initial}} & J_{\text{initial}} & k \\ 1 & 1 & J_{\text{final}} \end{Bmatrix}, \tag{3.11}$$

where the last term is the Wigner 6-j symbol and $\hat{J} = \sqrt{2J_{\text{initial}}+1}$. The reduced statistical tensor can be interpreted as an alignment parameter of the initial state, and is given by [50]:

$$\begin{aligned} A_{20}(\alpha_{\text{initial}}J_{\text{initial}}) = &\left[\frac{5}{(2J_{\text{initial}}+3)J_{\text{initial}}(J_{\text{initial}}+1)(2J_{\text{initial}}-1)}\right]^{1/2}\frac{1}{\sigma(\alpha_{\text{initial}}J_{\text{initial}})} \\ &\times \sum_{M_{\text{initial}}}[3M_{\text{initial}}^2 - J_{\text{initial}}(J_{\text{initial}}+1)]\sigma(\alpha_{\text{initial}}J_{\text{initial}}M_{\text{initial}}). \end{aligned} \tag{3.12}$$

The sum is done over all the possible total angular momentum projections, M_{initial} within the total angular momentum J_{initial} state manifold. $\sigma(\alpha_{\text{initial}}J_{\text{initial}}M_{\text{initial}})$ is the electron impact ionization cross section for the state identified by the quantum numbers α_{initial}, J_{initial} and M_{initial}. $\sigma(\alpha_{\text{initial}}J_{\text{initial}})$ is the total cross section for the state $\alpha_{\text{initial}}J_{\text{initial}}$, i.e. $\sigma(\alpha_{\text{initial}}J_{\text{initial}}) = \sum_{M_{\text{initial}}}\sigma(\alpha_{\text{initial}}J_{\text{initial}}M_{\text{initial}})$.

Equation 3.10 is deduced for a detector with a solid angle of 4π (full sphere), therefore the factor of $\frac{1}{4\pi}$ is introduced. This factor was excluded to calculate the full emission intensity, as often, experimental results are in counts or arbitrary units.

CHAPTER 4

Simulation Results

In this chapter, the tests performed on the simulations that were constructed and modified in this work are presented. Following the outline in figure 2.1, the results from testing the Geant4 simulation will be presented first. The check will consist on the verification of several proportionality relations, namely the density of the sample, elemental composition, as well as the cut value specific of Geant4.

An explanation of the changes made to the DCS code will be given afterwards, including the tests of these modifications. This testing was done by using a synthetic emission spectrum from titanium's $K_{\alpha 1}$ and $K_{\alpha 2}$ lines following the theoretical formalism of chapter 3.

4.1 Planar XRF Simulation Results: Geant4, ROOT

In this section, the results from various tests performed on the first stage simulation will be presented, with an explanation of the respective simulation conditions. All tests were performed using a gold alloy reference sample for which an experimental spectrum was provided. The sample's reference composition can be found in table 4.1.

Table 4.1: Gold alloy pattern's reference elementary composition used in the simulation.

	Au	Ag	Cu
Percentage (%)	70.5 ± 0.1	24.6 ± 0.1	4.9 ± 0.1

In figure 4.1, it is shown an image of the simulation's graphical representation, with and without simulated rays. Each element of this simulation is identified by color: X-Ray tube layer in yellow, sample volume in green and detector volume in red. The simulated rays are shown in green, as they are photons (electrons are featured as red tracks in the

simulation). The yellow points are events, corresponding to physical processes, such as absorption and fluorescence.

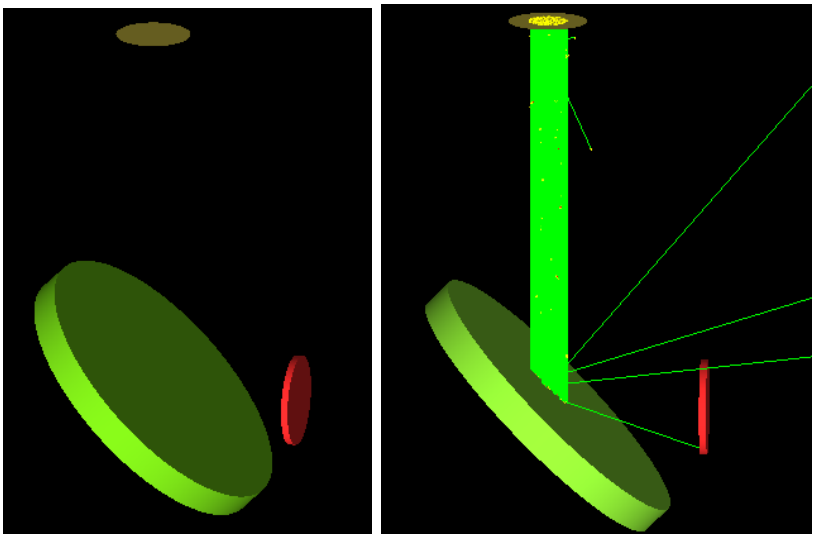


Figure 4.1: Graphical representation of the Geant4 simulation. The simulation without any simulated rays is shown on the left and an example with 1000 simulated rays is shown on the right.

The simulation used in this work reproduces the X-Ray tube emission by generating a photon beam and passing it through a thin layer of the tube's target material. The initial position of the photons is uniformly distributed inside a circle, and the energy distribution of this beam is generated to recreate the Bremsstrahlung background of a typical X-Ray tube's emission spectrum. This energy distribution can be sampled from a histogram or generated from an analytical expression. This is a good approximation of the X-Ray tube's emission spectrum and speeds up the simulation significantly, by producing less particles and interactions/events, compared to simulating the full Bremsstrahlung process. The sampling is performed similarly to the method described in section 2.2. A graphical representation of the sampled distribution is shown in figure 4.2.

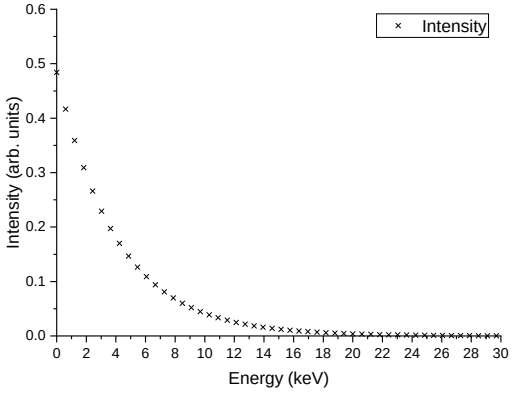


Figure 4.2: Distribution from which the simulated energy is sampled. This distribution is configured for a tube operating voltage of 30 kV.

The analytical distribution is similar to the intensity of a $1/E$ function would produce:

$$E = E_{max}(1 - \sqrt{x}), \tag{4.1}$$

where E is the energy assigned to the generated photon, E_{max} is the maximum energy that can be assigned and corresponds to the electric potential applied to the tube, x is a uniform random variable, between 0 and 1. Using this distribution, a limited energy generation between 0 and E_{max} is calculated.

In the case of this sample the results were better adjusted to experiment by using an analytical distribution (equation 4.1), instead of the sampling process. A graphical representation of a simulation done using the X-ray tube's sampler and one using only the analytical distribution can be seen in figure 4.3, respectively on the right and left. The lines identified correspond to the main lines of the elements present in the sample.

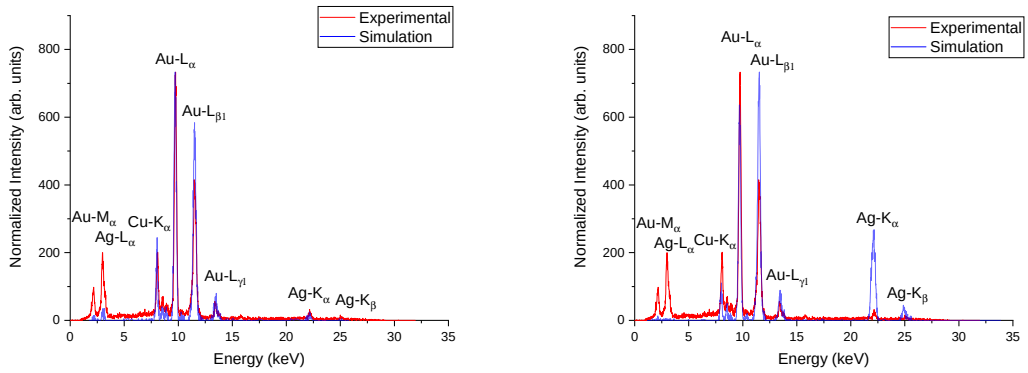


Figure 4.3: Graphical representation of the simulated spectrum using only the analytical distribution on the left and the simulated spectrum using the sampler on the right. Both spectra are superimposed over the experimental spectrum and are normalized to the maximum experimental intensity for comparison. The simulation conditions for both spectra were the same.

The detector used in the experiment is an energy dispersive Si-PIN detector, which was simulated including its energy dependent detection efficiency, energy dependent resolution and sensitive volume geometry indicated by the manufacturer.

While analyzing the simulated spectra, some differences in relation to the experimental spectrum were noticed, specifically much more intense lines than the experimental spectrum for high energies and much less intense lines for low energies. These differences are due to unknown factors, such as the X-ray tube's anode and detector's dead layer thicknesses, which change over time. Additionally, the gold alloy sample was manufactured to recreate old forging methods, resulting in a larger uncertainty in the elemental composition than the one referenced in table 4.1. The influence of the simulation's cut value in these differences has been mostly excluded after further testing, as it remained constant and did not affect the results.

In more detail, the simulation using the sampler presents higher than expected intensities for some lines. Specifically, gold's $L_{\beta 1}$ line (11.440 keV) and silver's K_α and K_β lines (22.101 keV and 24.928 keV). Both spectra present lower than expected intensities for gold's M_α line (2.120 keV) and silver's L_α line (2.984 keV). In table 4.2 the results from fitting a Gauss function to the Cu K_α, Au L_α, Au $L_{\beta 1}$, Au $L_{\gamma 1}$ and Ag K_α lines are shown, providing a quantitative comparison of the spectra.

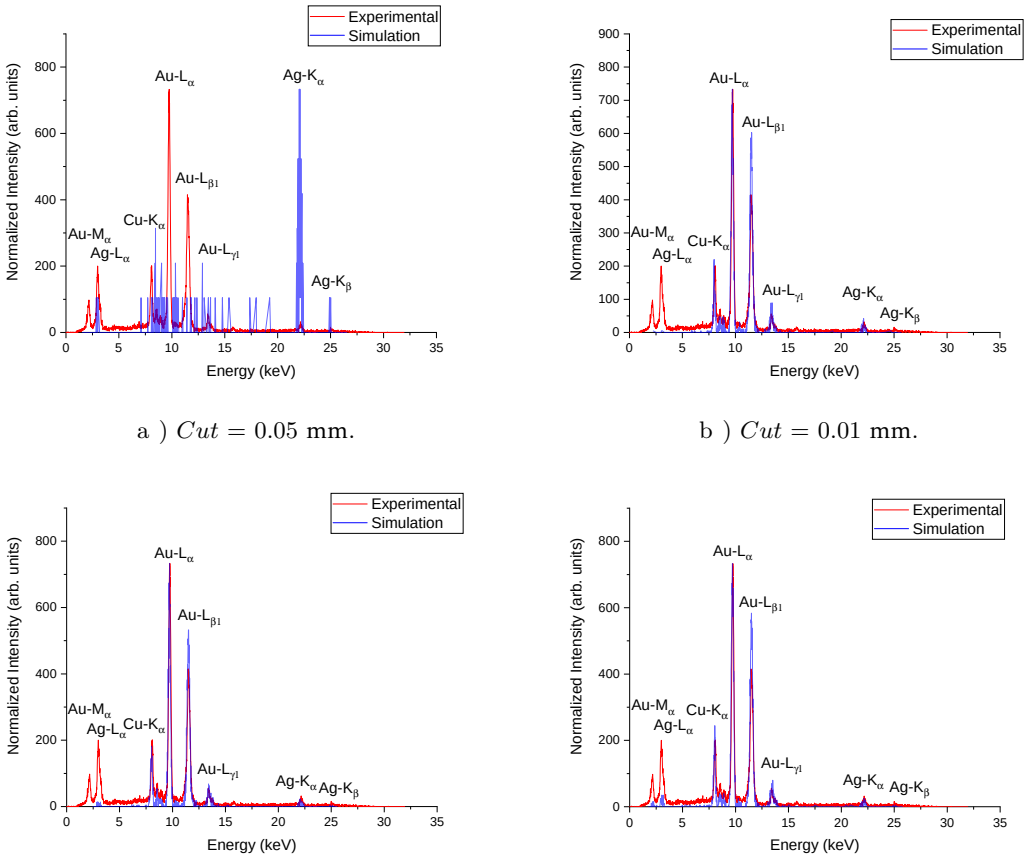
Table 4.2: Summary of the fitting results to the Cu K_α, Au L_α, Au $L_{\beta 1}$, Au $L_{\gamma 1}$ and Ag K_α lines. These lines were chosen due to the consistent fitting results across both simulations. The gaussian function fitted is $y = y0 + Hexp\left(-2\left(\frac{x-xc}{w}\right)^2\right)$. The e, a and s indexes are respective to the parameter of the experimental, analytical method and sampling method spectra. The r^2 values for these fits were all greater than 0.97.

	Cu K_α	Au L_α	Au $L_{\beta 1}$	Au $L_{\gamma 1}$	Ag K_α
$y0_e$	15.6 ± 0.5				
xc_e	8.075 ± 0.002	9.7454 ± 0.0005	11.524 ± 0.001	13.44 ± 0.01	22.18 ± 0.02
w_e	0.200 ± 0.004	0.201 ± 0.001	0.283 ± 0.002	0.18 ± 0.02	0.30 ± 0.05
H_e	172 ± 5	695 ± 5	378 ± 4	36 ± 5	18 ± 4
$y0_a$	4.3 ± 0.7				
xc_a	8.0155 ± 0.003	9.7084 ± 0.0009	11.514 ± 0.001	13.46 ± 0.01	22.15 ± 0.02
w_a	0.158 ± 0.005	0.220 ± 0.002	0.262 ± 0.002	0.29 ± 0.03	0.11 ± 0.05
H_a	188 ± 9	694 ± 7	540 ± 7	47 ± 7	17 ± 10
$y0_s$	6 ± 1				
xc_s	7.97 ± 0.01	9.738 ± 0.002	11.502 ± 0.001	13.48 ± 0.02	22.125 ± 0.005
w_s	0.19 ± 0.03	0.218 ± 0.003	0.246 ± 0.003	0.26 ± 0.03	0.31 ± 0.01
H_s	69 ± 13	610 ± 12	717 ± 12	65 ± 11	245 ± 11

From this table, the simulation using the analytical expression shows a better agreement with the experimental spectrum compared to the sampler distribution. Both simulations were performed with the same number of generated events.

4.1.1 Simulation Cut Value

As explained in section 2.3.2, the Geant4 package uses a cut value (Cut) to remove from the simulation particles whose expected travel path is lower than a certain limit. Ideally, to exactly simulate a particle track, this cut value would be zero, but this would result in very long simulation times, consuming large amounts of memory. Depending on the materials (elements) in the simulation, the cut value can be adjusted, in a way that the simulation is both reliable and time affordable. For metals and heavier elements, a good cut value is around 0.01 mm or lower, but for lower density materials like organic material around 0.1 mm is already a good cut value. To ascertain the effects of the cut value in the simulated spectrum simulations with various cut values were executed. Some of tests are presented here and their respective cut values are: 0.05 mm, 0.01 mm, 0.005 mm, 0.0005 mm. The resulting spectra are represented in figure 4.4.



a) $Cut = 0.05$ mm.

b) $Cut = 0.01$ mm.

c) $Cut = 0.005$ mm.

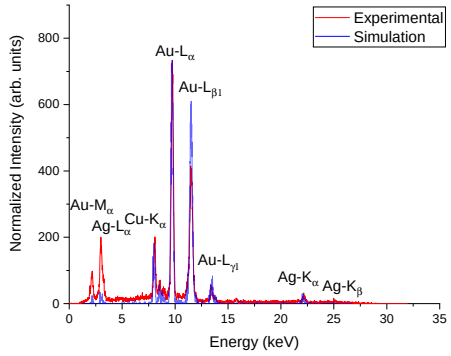
d) $Cut = 0.0005$ mm.

Figure 4.4: Graphical representations of the simulated spectra for different values of the simulation's cut value. All spectra are normalised to the experimental intensity.

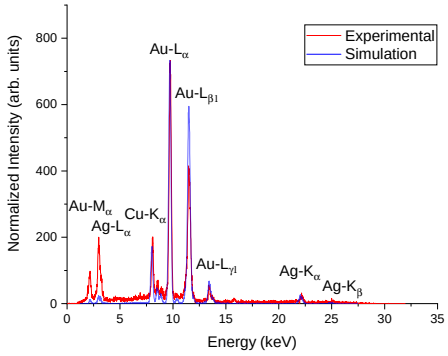
Apart from a few statistical fluctuations, as the cut value decreases the intensity of the lines at lower energies increase and the intensity of the lines at higher energies decrease. Also a too high cut value means that only higher energy radiation will be able to reach the detector, as seen for the case of $Cut = 0.05$ mm, where energies of ≈ 8 keV and bellow were not detected.

4.1.2 Density

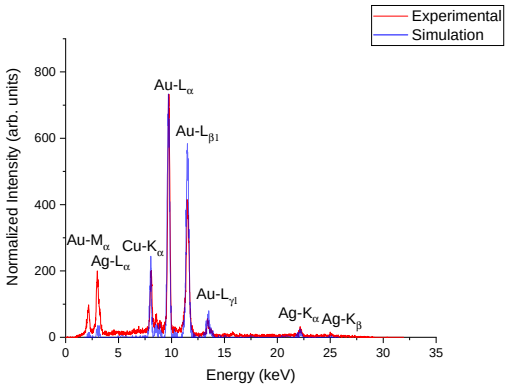
The sample's density can also influence the simulated spectrum. Tests regarding the density of the gold alloy reference material were also performed. These tests were performed using the density from the least dense element, the most dense and the weighted average density (expected density) using the sample's reference composition. Some of these tests can be observed in figure 4.5 and were executed using a cut value of 0.0005 mm.



a) density of 10.49 $\mathrm{g\,cm^{-3}}$.



b) density of 19.30 $\mathrm{g\,cm^{-3}}$.



c) density of 16.64 $\mathrm{g\,cm^{-3}}$.

Figure 4.5: On the top left is the graphical representation of the simulated spectrum for a density of 10.49 $\mathrm{g\,cm^{-3}}$, on the top right a density of 19.30 $\mathrm{g\,cm^{-3}}$ and on the bottom a density of 16.64 $\mathrm{g\,cm^{-3}}$. The density values are the exact ones used in the simulation, and as such do not have associated uncertainties. All spectra are represented over the experimental spectrum and are normalized to the experimental intensity for comparison. The simulation conditions for all spectra were the same apart from the density.

The main changes that appear from changing the density while using a small and constant cut value are the variation of intensity for copper's K_α line and for gold's $L_{\beta 1}$ line.

4.1.3 Intensity and Concentration Linearity

A simple way to test if the physical processes used in the simulation are working properly is to check simple relationships between the input and output of the simulation. In this case, one of these tests is the linear relationship between the input concentration of an element and its respective line intensity. To test this, the sample's composition was changed, various simulations were executed and the lines were fitted with a Gaussian profile using ROOT, obtaining their intensity in a more accurate way. The lines used for this test

were copper's K_α and gold's $L_{\beta 1}$ lines. To illustrate the limits of this linearity, silver lines are not shown as the concentration of silver is in between both copper and gold. These tests can be observed in figure 4.6 and were executed using the weighted average density, 16.64 $\mathrm{g\,cm^{-3}}$. The error bars in the figure represent the fitting error calculated by ROOT.

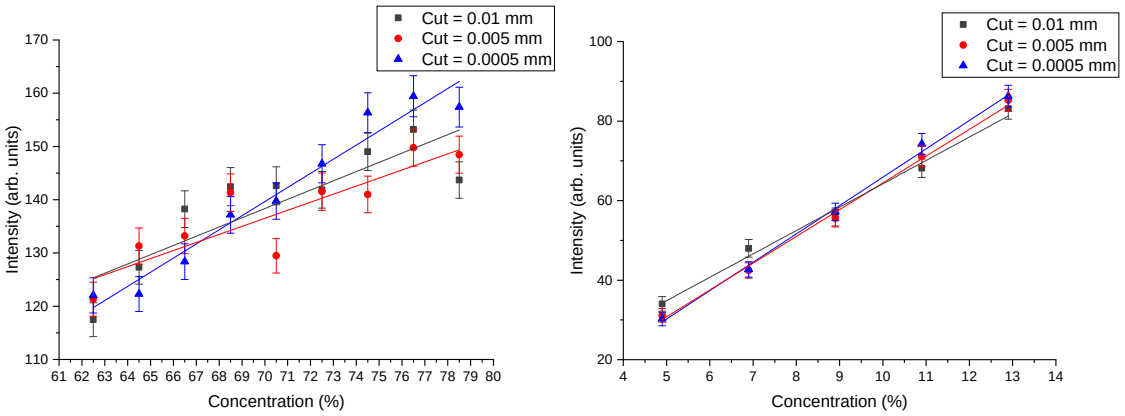


Figure 4.6: Left. Graphical representation of the intensity of gold's $L_{\beta 1}$ line as a function of gold concentration. Right. Graphical representation of the intensity of copper's K_α line as a function of copper concentration. The linear fits performed are represented with the respective data color

The data points in figure 4.6 were fitted using a linear fit. On the left, the intensity of gold's $L_{\beta 1}$ line is represented as a function of gold concentration and on the right, the intensity of copper's K_α line is represented as a function of copper concentration. For comparison the adjusted r squared values for the copper were 0.99617, 0.9949, 0.98876 for the cut values of 0.0005 mm, 0.005 mm and 0.01 mm respectively. For gold, in the same order of cut values, the adjusted r squared values were 0.95476, 0.74085 and 0.69882. Therefore a linear dependency between intensity and concentration is more noticeable for a cut value of 0.0005 mm.

4.1.4 Summary

In this section, various tests performed on the Geant4 simulation of the planar XRF spectrometer were shown. Through these tests, it was shown that the simulation properly implements the necessary physical processes to simulate the target system. However, it still needs to be optimized for specific cases, such as a different X-Ray tube material.

4.2 DCS Simulation Results

In this section, the changes made to the existing DCS code are described. The tests performed to validate the changes made are also presented.

4.2.1 Simulation Changes

The changes made to the simulation were mainly on the X-ray source's energy generation. Previously, the simulation was restricted to read from one to four energy lines and their corresponding natural width from user input. Using these values, the energy distribution was generated according to a Lorentzian distribution. To repurpose this simulation for an arbitrary input XRF spectrum, the X-ray source's energy generation must be modified. As described in section 2.2, the rejection method (section 2.2) was implemented. Taking an arbitrary spectrum as input, an energy distribution is generated for the simulated X-ray source such that it follows the input spectrum's energy distribution.

The implemented code generates an energy value within the limits of the accepted energy window, using a uniform distribution as: $E = rand \times delta + start$. Here, $rand$ is a uniform random value between 0 and 1, $delta$ is the energy window span and $start$ is the minimum value of the energy window. The corresponding intensity value of this energy is calculated from the input spectrum, using a simple spline interpolation, and is then compared to a random, uniformly distributed value of intensity. This intensity value is given by $I = rand \times deltaI + startI$, where $deltaI$ and $startI$ are the intensity span of the full input spectrum and the minimum intensity of the full input spectrum, respectively. If the value calculated from the energy and spline interpolation is less than the random intensity value, then the energy is used. Otherwise another energy value is generated and checked, repeating this process until an energy value is accepted. The respective code can be found in section A.1

4.2.2 Simulation Tests

Firstly, the changes were tested using an input spectrum only with a background and two added peaks. Afterwards, a synthetic spectrum of titanium's $K_{\alpha 1}$ and $K_{\alpha 2}$ lines were used to perform further testing. The results from both tests are respectively represented on the left and right sides of figure 4.7.

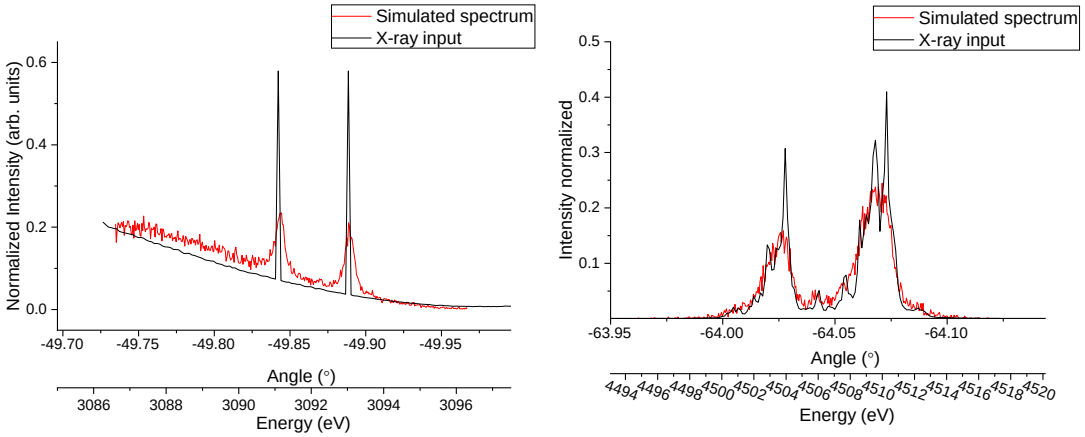


Figure 4.7: Left. The test X-Ray input with background and two peaks is represented in red and the respective DCS simulation output in black. Right. The titanium X-Ray lines spectrum is represented in red and the respective DCS simulation output in black. Both spectra have the crystal angle and energy on the x-axis, and are normalized to their respective area.

In these tests it can be seen that the simulated spectrum follows the respective input spectrum, with the addition of various effects. The statistical effects and the broadening due to the crystals' response function can be clearly seen. The simulation also includes less obvious effect as the geometric effects, systematic and otherwise, such as the vertical and horizontal divergence of each simulated ray. The downside of the changes made is an increased simulation time, as the energy of each X-Ray has to be sampled, compared to using an analytical expression for the energy.

4.3 Summary

In this chapter the results from testing both the Geant4 simulation and the DCS simulation were presented.

Relatively to the Geant4 simulation, the physical processes were tested, guaranteeing that the simulation is representing the target system, i.e. the planar XRF spectrometer. Although the simulation could not be completely adjusted to the system due to various setbacks, the fundamental tests were performed to optimize the simulated X-Ray tube and detector in the future. Both the X-Ray tube's emission spectrum and the detector's properties are dependent on the device used, having to be optimized for the specific case. Additionally, the detector's properties, such as the detection efficiency and dead layer slowly change with time, requiring a new optimization.

In terms of the DCS simulation, the necessary changes were applied and tested. These changes now allow the simulation to take an input spectrum, such as the one simulated with Geant4, from which it can be randomly sampled. As explained in section 4.2.1, an arbitrary spectrum can be used as the energy generator for this simulation, i.e. the energy distribution is replicated inside the simulation and is used as the source's energy

distribution. Due to the input spectra having to be an arbitrary numeric distribution, the method of rejection sampling was implemented. This method leads to significantly larger simulation times (more than 10 times), as multiple energy values can be rejected before one value is accepted, and the corresponding ray is simulated.

The DCS simulation has also been implemented with a graphical interface using the dislin library [51]. Although this library is independent of the operating system, it was very problematic to use with the Windows operating system compared to Ubuntu. As several issues occurred while compiling the FORTRAN code with the dislin library, a C++ version was programmed, using the Qt library. An image of the graphical interface for the C++ version is in figure 4.8.

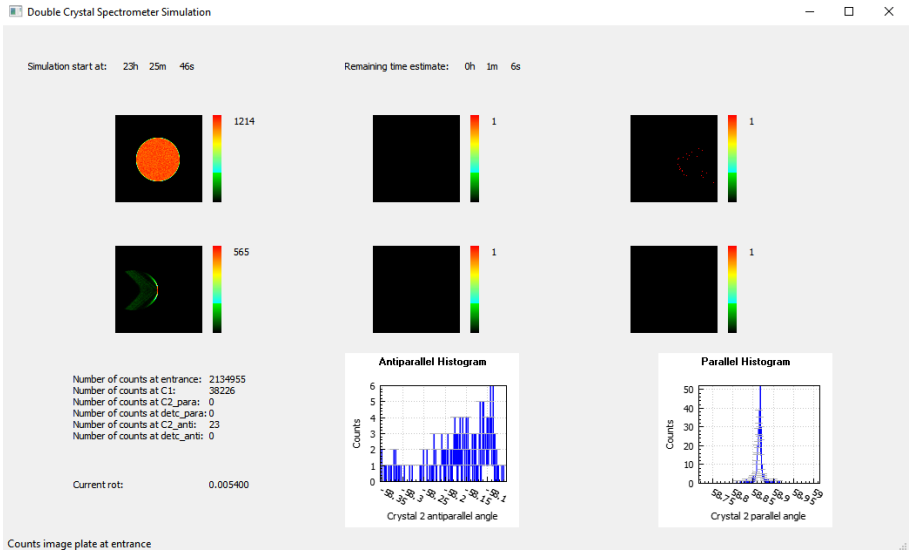


Figure 4.8: Graphical user interface for the C++ version of the DCS simulation.

Although these simulations are not fully optimized in terms of the geometric setting of the simulated elements, detector and X-Ray tube properties, they are ready for this final optimization when these parameters are defined in the final spectrometer setup being built at the LIBPhys-UNL group.

CHAPTER 5

Synthetic Spectra

In section 4.2, it was mentioned that a synthetic spectrum of titanium's $K_{\alpha 1}$ and $K_{\alpha 2}$ lines was used as a test input for the DCS simulation. These type of spectra (synthetic) are another good route to predict the radiative X-Ray emission of an atomic system from first principles.

As explained in section 3.1.3, synthetic spectra are constructed from lines with their energies and intensities calculated theoretically. This theoretical approach brings several advantages over simulations done with Monte Carlo packages like Geant4, mainly for the prediction of intensity and positions of the fluorescence lines. The major advantage, in terms of fundamental studies, is that synthetic spectra allow direct identification of the atomic states and processes responsible for discrepancies between models and experiment. This identification is much harder or impossible to do with simulated where atomic data is either simplified or not accessible for users. This is even more imperative when working with both high resolution and precision X-ray spectrometers, such as the DCS. Identifying the states involved can be used to benchmark the theory in more detail and make possible adjustments that might be needed.

A conjugation of high-resolution spectroscopy and synthetic spectra, also allow the study of interest, but faint, processes such as from chemistry and solid state effects, as well as polarization, and even quantum interference (QI) effects, as we shall see in section 5.2.2 and 5.3.1. For example, the last case of QI needs the identification of the atomic states' total angular momentum and parity that can only be done with a previous theoretical evaluation of the atomic states. Some disadvantages also come with this type of approach, namely the overlook of experimental specificities (e.g. geometry) on the calculated spectrum, and mainly the increased calculation time. Additionally, the ionization and excitation cross sections considered to calculate each shell's (K, L, M, etc) ionization/excitation probability has to be calculated separately. Moreover, some cross sections such as photoionization

cross sections, for incident photons, are more complex to calculate then for (e.g.) electron impact ionization cross sections.

In this chapter, results of different synthetic spectra are presented. As it will be explained, the theoretical approach is very time consuming. Therefore, only some emission lines could be calculated with the time window of this work. Furthermore, we focus on spectra generated by electron impact ionization and thus electron impact ionization cross section was used in the synthetic spectrum calculations. Further considerations, relatively to the QI process and the angular distribution of the emitted radiation, are also presented with their respective spectra. Although the effects from both of these processes can be very subtle, due to the DCS's high resolution, these processes will be observed in this spectrometer and are thus needed for a complete description of the spectra.

5.1 MCDF Calculation Script

The MCDF program described in section 3.1 is written in FORTRAN and compiled in an executable file, it reads the input from a .f05 file, also denominated by unit in FORTRAN, and writes the output to a .f06 file. To automate the calculation of potentially millions of states, a script in python was written (section A.2). This script had several versions, starting with a single-threaded approach and further developed to a multi-threaded calculation, while reporting various useful performance statistics.

Multi-threading is a way of processing where multiple instructions can be executed in a single-central processing unit (CPU) clock cycle. Nowadays, CPU's can have multiple processing cores that can physically process different instructions at the same time. Some CPU's can additionally, within each core, process multiple instructions in the same cycle, through the use of virtual threads. Multi-threading is very advantageous to use for this type of calculations, as multiple states or transitions can be calculated simultaneously. The number of CPU cores and threads is a hardware limitation that depends on the CPU.

To illustrate how time consuming MCDF calculations are, various statistics from the calculations performed are presented in table 5.1. The computer where the script was running allowed for the use of a maximum of 12 threads. According to the explanation in section 3.1.3.2, the table columns identified as "no mixing" and "with mixing" are, respectively, the number of transitions with only $eigv = 1$ and the number of transitions taking into account the all $eigv$'s calculated by MCDF.

The statistics on this table use a 20 s time interval per cycle, which corresponds to ≈ 180 calculations per hour in single-thread (S) and ≈ 2160 in multi-thread (M, 12 threads). Additionally, only one-step processes were considered (section 3.1.3.1), due to the large total calculation time required.

The nickel calculations were also performed before the improvement towards the multi-threaded case, therefore these calculations were significantly slower, as can be seen from the numbers in the table. Some states were excluded from the number of mixed states, either due to the transition rate being zero, or the final state calculation not converging.

Table 5.1: Summary of the number of calculations to perform for nickel, and respective time cost averages. A good average of time per calculation, is 20 s per cycle, i.e. MCDF running time and input/output reading/writing times combined. This corresponds to ≈ 180 calculations per hour in single-thread (S) and ≈ 2160 in multi-thread (M). The total times in the last 2 columns are respective to the number of transitions in the second column of the table and the respective time averages. N is the number of transitions.

	N no mixing	N with mixing	Total Calculation Time (S)	Total Calculation Time (M)
Ni, Radiative	33092	78408[1]	435.6 h (18.15 d)	36.3 h
Ni, Auger	137559	211347[2]	1174.15 h ($\approx$ 48.92 d)	$\approx$ 97.85 h

[1] Number of transitions for the K and L_1 shells.
[2] Number of transitions for the K shell.

As observed in table 5.1, in terms of time performance, the use of multi-threaded processing significantly reduces the total calculation time. An easy way to code multi-threaded programs is using python, as most of the functions needed to launch and manage parallel processes are already wrapped in various modules. For example in this case the `psutil`, `subprocess`, `signal` and `time` modules were used in the final version of the script to launch, manage and monitor the parallel calculations.

There were some challenges in interfacing with the MCDF executable with the script, primarily because it is not a direct interface, through code, but an indirect console execution of the compiled code. Additionally, as the MCDF code was written in FORTRAN, due to some compatibility problems with more recent systems, the code sometimes produces memory access errors in some calculations, more noticeably in Auger calculations. Sometimes these errors also cause the process to stall, i.e. the process remains alive indefinitely without performing any calculations. A workaround to this was implemented through a timeout of 25 s, i.e. 5 s more than the cycle time average considered in table 5.1. Another difference from single- to multi-threaded processing, due to MCDF being written in FORTRAN, was the need to have multiple copies of the executable and input files, as the input file stays unchangeable while the associated MCDF executable is running.

5.2 Synthetic Emission Spectra

The synthetic emission spectra from nickel are presented in this section. The radiative contributions to these spectra were calculated considering the multi polar decomposition of the emitted radiation, where all possible multipoles were included. As expected, the electric dipole radiation emission is the most intense, followed by the magnetic dipole and electric quadrupole. Additionally, only one-step processes were included.

5.2.1 Nickel Spectrum

In figure 5.1, the graphical representation of a synthetic emission spectrum of nickel is shown. Due to time limitations, only the K series include Auger process contributions to each line's intensity and width. The ionization cross section used in the calculation of this spectrum is the electron impact ionization cross section and the energy considered is 40 keV. This cross section was used to calculate the probability of ionization by electron impact for each atomic subshell. The model used to calculate the electron impact cross section is the Binary Encounter Bethe (BEB) model, with the modifications explained by Guerra *et.al.* in [52]. A non-relativistic version and a relativistic equivalent were also deduced in that work and the relativistic version was used (MRBEB). Although, for 40 keV the difference between the classic and relativistic models is very small (≈5 % for nickel), this difference increases with the impact energy and the relativistic model will give more reliable results.

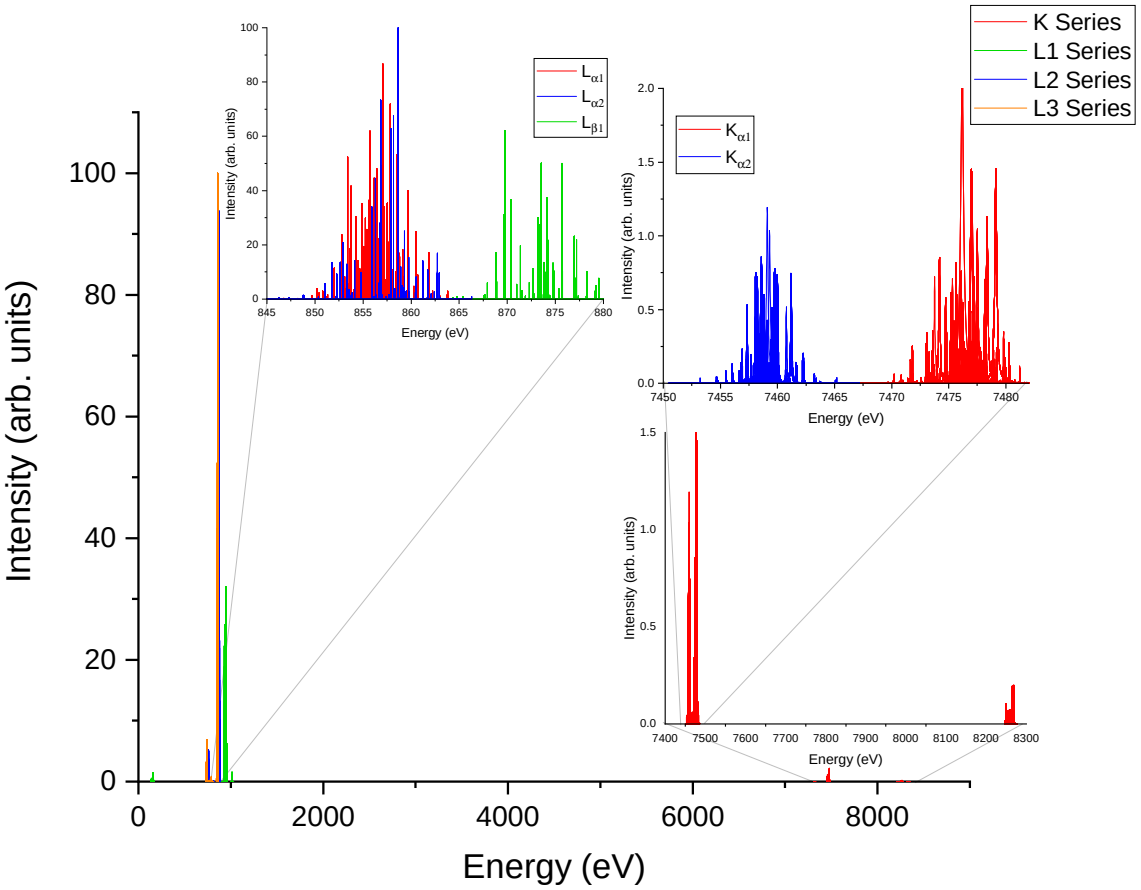


Figure 5.1: Graphical representation of nickel's emission spectrum. Only the K series takes into account the Auger process. The electron impact ionization cross section was considered for an energy of 40 keV.

5.2.2 Nickel Quantum Interference Spectrum

The QI process is a phenomenon that involves two distinct radiative transitions and can be observed when certain conditions are met. In the case of interference for the emitted radiation from an atom, quantum interference can occur when the initial and final states of a transition have the same total angular momentum and parity as another transition. Additionally, the transitions have to be close enough, in terms of energy, so that the interference is not zero, e.g. within half of the full width at half maximum (FWHM) of each transition (section 3.1.3.4). In figure 5.2, the lines that fulfill the conditions described are shown with a different color, over the full spectrum. This identification is performed on the K_α lines of the spectrum shown in figure 5.1, which also include Auger contributions.

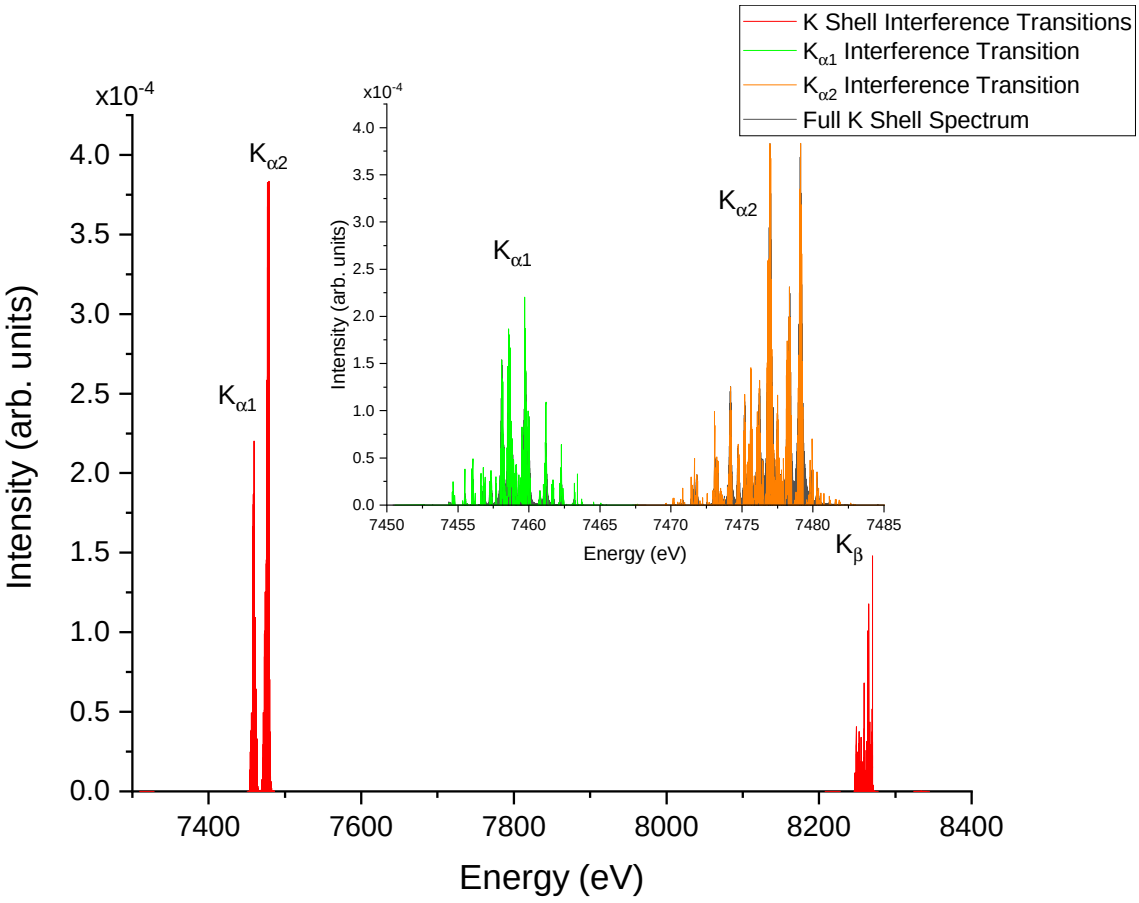


Figure 5.2: Graphical representation of nickel's emission spectrum of the K_α lines that fulfill the QI condition. This spectrum was calculated using the same parameters as the full nickel spectrum. The full spectrum is also shown under the QI lines.

5.3 Angular Distribution Spectra

In this section, the results from the effects described in section 3.1.3.5, on the previously presented spectra are shown. Additional results of the angular distribution for rhodium's K shell are also shown.

5.3.1 Nickel Angular Emission Spectrum

The angular distribution for nickel's K shell is plotted in figure 5.3. These angular distributions were calculated using equation 3.10 with a maximum intensity $W_0 = 4\pi$, giving normalized results.

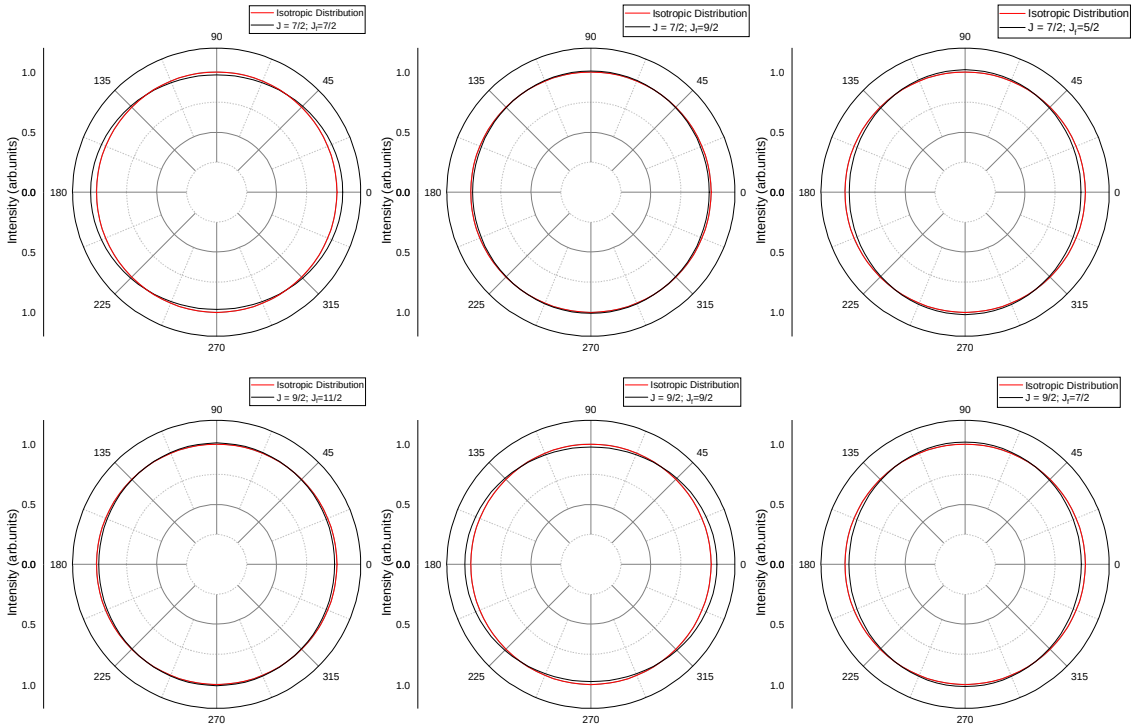


Figure 5.3: Graphical representation of the angular distributions for dipolar radiation emitted from K shell electron impact ionizations in nickel.

The present calculation contains $J = 7/2$ and $J = 9/2$ that within the dipole approximation corresponds a J_f from 5/2 to 11/2. The respective non isotropic combinations are shown in figure 5.3. For combinations where the difference between the initial and final J is greater than 1, the Wigner 6-j in equation 3.11 is equal to 0. Additionally, the distributions corresponding to $J = J_f$ show a greater difference from the isotropic distribution. It was also mentioned in [49] that the symmetry of the orbitals of the states involved in the transition also influences the isotropic nature of the emission. Therefore, a larger anisotropy is expected for the L and M shells in nickel.

In terms of quantitative differences that these considerations might bring to spectra, when looking at the maximum and minimum values of each distribution, there are differences of up to 8.4 % in intensity. At an angle of 0° for the angular distributions with $(J = 9/2,\ J_f = 9/2)$ and $(J = 7/2,\ J_f = 5/2)$, a maximum and minimum are observed, respectively. The value of maximum emission is approximately 1.0485 and minimum emission is approximately 0.9643, which results in a difference of approximately 8.42 %.

The necessary electron impact ionization cross sections for equation 3.12 were calculated using the Flexible Atomic Code (FAC). In this case the cross sections for magnetic sub

levels (total angular momentum projection M_J) were needed, which is a result that MCDF or the MRBEB model do not provide. Using these angular distributions, the previous K shell X-Ray spectrum (figure 5.1) was modified and the spectrum in figure 5.4 was obtained.

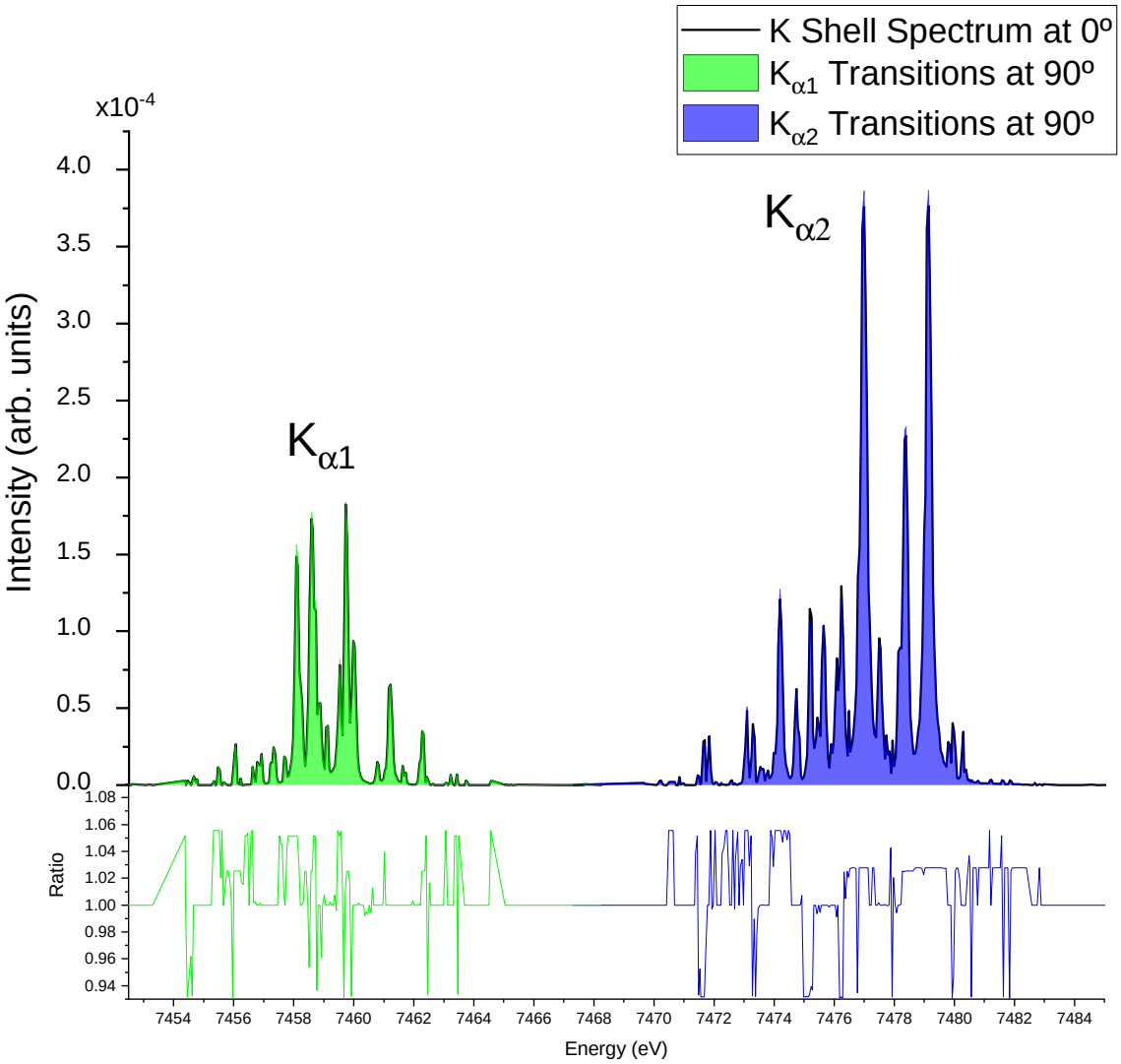


Figure 5.4: Graphical representation of the Kα_1 and Kα_2 X-Ray lines, generated by electron impact ionization in nickel. Both the expected spectra at 0° and 90° are represented. The ratio of the intensity at 90° over the intensity at 0° is represented below, to better visualise the differences. This spectrum is not convoluted with an instrumental function for broadening.

As the angular distribution only depends on the initial and final total angular momentum J, the intensity of all the lines that correspond to the initial and final angular momentum values were multiplied by the respective distribution's value. Below the overlapped spectra, a graphical representation of the intensity ratio is also presented, as the effect of the angular distribution on the final spectrum is very subtle. This is due to all the contributions from different transitions that might be isotropic or have opposite

effects. The segments where the ratio is 1 correspond to regions of the spectrum were the transitions have isotropic emission. The instrumental function (experimental broadening) needs to be considered in the spectra of figure 5.4.

5.3.2 Rhodium Angular Emission Distribution

The angular distribution for rhodium's K shell is plotted in figure 5.5. These angular distributions were calculated with a maximum intensity W_0 of 1, or using equation 3.10 directly, $W_0 = 4\pi$.

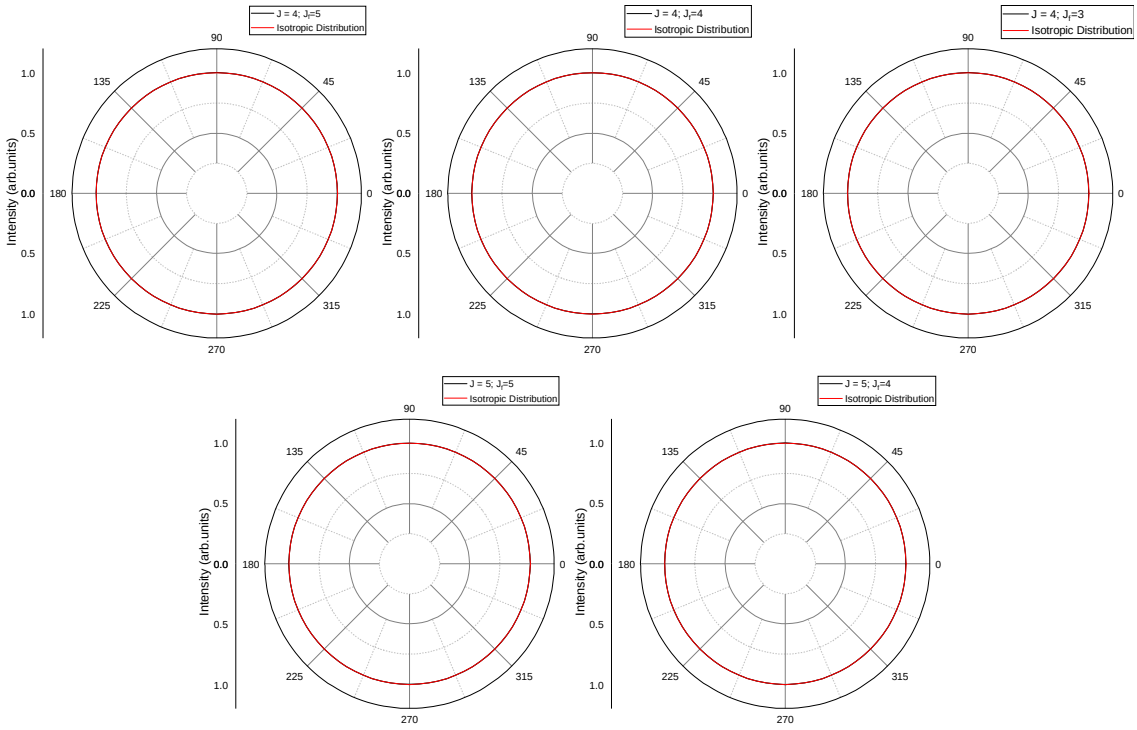


Figure 5.5: Graphical representation of the angular distributions for dipolar radiation emitted from K shell electron impact ionizations in rhodium.

The present calculation contains $J = 4$ and $J = 5$ that within the dipole approximation corresponds a J_f from 3 to 5 (5 is the largest total angular momentum of the system in this case). The respective non isotropic combinations are shown in figure 5.5. For combinations where the difference between the initial and final J is greater than 1, the Wigner 6-j in equation 3.11 is equal to 0. Additionally, the distributions corresponding to $J = J_f$ show a greater difference from the isotropic distribution. These distributions are closer to the isotropic distribution in comparison to the nickel distributions, this is due to various factors. The K shell mean radius in rhodium is smaller than in nickel (from MCDF: Ni K shell mean radius is $\approx 0.054a_0$, Rh K shell mean radius is $\approx 0.032a_0$, a_0 is the Bohr radius), the K shell binding energy for rhodium is larger than the binding energy for nickel (from MCDF: Ni K shell binding energy is ≈ 8397.974 eV, Rh K shell binding energy is ≈ 23317.487 eV). The MRBEB model for the electron impact ionization cross section, predicts a larger cross

section for the same energy in nickel compared to rhodium, as rhodium has a larger K shell effective nuclear charge and smaller mean radius (from MRBEB: Ni K shell cross section is $\approx 6.688 \times 10^{-25}$ m^{-2}, Rh K shell cross section is $\approx 6.205 \times 10^{-26}$ m^{-2}). A larger cross section indicates a larger alignment parameter (equation 3.12), which translates into a larger anisotropy for nickel. For example, the alignment parameter for nickel, with $J = 9/2$ is approximately -0.1100931, and for rhodium, with $J = 4$ is approximately -0.0000011.

In terms of quantitative differences that these considerations might bring to the spectra, when looking at the maximum and minimum values of each distribution, there are differences of up to 0.00015 % in intensity. At an angle of 0° for the angular distributions with $(J = 5,\ J_f = 5)$ and $(J = 5,\ J_f = 4)$, a maximum and minimum are observed, respectively. The value of maximum emission is approximately 1.00000087 and minimum emission is approximately 0.99999942.

Using these angular distributions, the previous K shell X-Ray spectrum (figure 5.1) was modified and the spectrum in figure 5.4 was obtained.

CHAPTER 6

Conclusions and Future Perspectives

In this work, two simulations were tested, to guarantee the accurate representation of the physical processes involved. As mentioned in chapter 2, the system to be simulated is a planar XRF spectrometer, followed by a DCS. This system is being built at the LIBPhys-UNL group, and a general layout of the system is also described there. To simulate this system, it is divided into two stages, the planar XRF spectrometer simulated using the Geant4 and ROOT packages, and the DCS stage simulated using a FORTRAN code.

The planar XRF spectrometer simulation, obtained using the Geant4 and ROOT packages, was tested for the physical processes and parameters being used. For example, the default cut value for the simulation was adjusted to account for the lower mean path of the radiation in materials of heavier elements, such as silver and rhodium. Additionally, the linear relationship between the concentration of an element in the sample and its respective lines' intensities was confirmed. With these tests the simulation can be optimized for a specific planar XRF spectrometer, i.e. the specific system's geometry, X-Ray tube and detector.

The DCS simulation, obtained using a home-made code, was changed to generate the source's energy distribution according to an arbitrary spectrum. This change is necessary to simulate an X-Ray fluorescence spectrum being measured by the DCS, as represented in the system's outline. This change was implemented and tested, confirming that the energy generation is being correctly calculated and the broadening from the crystals' response function is taken into account. This simulation can also be used to simulate the system, after configuring the necessary geometry, i.e. the distance between source, crystals and detector.

To investigate some less intense effects on spectra, that might be visible in the DCS measurements, synthetic spectra were also calculated, similarly to the titanium lines used as a test spectrum in the DCS simulation. The spectra were calculated for nickel, which

due to the large calculation times, were only fully calculated for K shell ionization, i.e. both Auger and radiative, one-step transitions for K shell ionization. The effects investigated using these synthetic spectra were the quantum interference process and the angular distribution of the emitted radiation. In the case of the quantum interference process, the lines that fulfill the necessary conditions were identified and the spectra composed of these lines was generated and compared to the full spectra. Further investigation can be focused on the calculation of these QI effects. For the angular distribution, the electron impact ionization cross sections were calculated with a different code (FAC) that took into consideration the total angular momentum projection in the calculation of the cross sections. Using the results from these calculations, the angular distributions were calculated and the original spectra were multiplied by these angular distributions. The resulting spectra were compared with the original spectra and differences of up to 8.4 % in the intensity of some lines were found. Overall differences of 4% were identified between 0^{o} and 90^{o} angle of observation for Ni. On the other hand, Rh contains an isotropic emission due to the smaller mean radius of the K shell (approximately 0.6 of the radius of the Ni K shell) and higher binding energy (approximately 2.78 times the binging energy for the Ni K shell).

In conclusion, the necessary simulation modifications were implemented and were tested, confirming that the simulations are ready to be configured and used to simulate the final spectrometer system. Additionally some effects, such as the quantum interference and the angular distribution of the emitted radiation, that might be observed in the measured DCS spectra were also investigated.

Bibliography

[1] W. C. Röntgen, "Auf eine neueu Art von Strahlen (On a New Kind of Rays)," *Science*, vol. 3, no. 59, pp. 274–277, 1895, ISSN: 0028-0836. DOI: 10.1126/science.3.59.227.

[2] A. H. Compton and S. K. Allison, *X-Rays in Theory and Experiment*, 2nd ed. D.Van Nostrand Company, Inc, 1935, p. 844.

[3] *European X-ray Spectrometry Association*, 2019. [Online]. Available: https://www.exsa.hu/news/?page_id=13 (Accessed: 01/07/2020).

[4] EXSA, "International initiative on X-ray fundamental parameters Roadmap document on atomic Fundamental Parameters for X-ray methodologies," no. June, 2012.

[5] J. A. Helsen and A. Kuczumow, "Wavelength-dispersive x-ray fluorescence," in *Handbook of X-Ray Spectrometry*, R. E. Van Grieken and A. A. Markowicz, Eds., 2nd ed., New York: Dekker, Nov. 2009, ch. 2, pp. 95–199, ISBN: 978-0824706005.

[6] A. T. Ellis, "Energy-dispersive x-ray fluorescence analysis using x-ray tube excitation," in *Handbook of X-Ray Spectrometry*, R. E. Van Grieken and A. A. Markowicz, Eds., 2nd ed., New York: Dekker, Nov. 2009, ch. 3, pp. 199–239, ISBN: 978-0824706005.

[7] R. D. Deslattes, "Two-crystal, vacuum monochromator," *Review of Scientific Instruments*, vol. 38, no. 5, pp. 616–620, 1967, ISSN: 00346748. DOI: 10.1063/1.1720781.

[8] J. Machado, C. I. Szabo, J. P. Santos, *et al.*, "High-precision measurements of n=2→n=1 transition energies and level widths in He-and Be-like argon ions," *Physical Review A*, vol. 97, no. 3, p. 32 517, 2018, ISSN: 24699934. DOI: 10.1103/PhysRevA.97.032517. [Online]. Available: https://doi.org/10.1103/PhysRevA.97.032517.

[9] P. Amaro, C. I. Szabo, S. Schlesser, *et al.*, "A vacuum double-crystal spectrometer for reference-free X-ray spectroscopy of highly charged ions," *Radiation Physics and Chemistry*, vol. 98, pp. 132–149, 2014, ISSN: 18790895. DOI: 10.1016/j.radphyschem.2014.01.015. [Online]. Available: http://dx.doi.org/10.1016/j.radphyschem.2014.01.015.

[10] A. Antognini, F. Nez, K. Schuhmann, *et al.*, "Proton structure from the measurement of 2S-2P transition frequencies of muonic hydrogen," *Science*, vol. 339, no. 6118, pp. 417–420, 2013, ISSN: 10959203. DOI: 10.1126/science.1230016.

[11] F. Biraben, J. M. R. Cardoso, D. S. Covita, *et al.*, "activation time sensitivity to hyperpolarization (8). The S2-S3 linker is also directed toward the S4/C-linker interface and may play a yet-unknown role in channel gating.," vol. 353, no. 6300, 2016.

[12] P. J. Mohr, D. B. Newell, and B. N. Taylor, "CODATA recommended values of the fundamental physical constants: 2014," *Reviews of Modern Physics*, vol. 88, no. 3, pp. 1–73, 2016, ISSN: 15390756. DOI: 10.1103/RevModPhys.88.035009. arXiv: 1507.07956.

[13] *LIBPhys FCT/UNL*, 2020. [Online]. Available: https://www.libphys.fct.unl.pt (Accessed: 10/09/2020).

[14] P. Amaro, S. Schlesser, M. Guerra, *et al.*, "Absolute measurement of the relativistic magnetic dipole transition energy in heliumlike argon," *Physical Review Letters*, vol. 109, no. 4, pp. 1–5, 2012, ISSN: 10797114. DOI: 10.1103/PhysRevLett.109.043005.

[15] M. Guerra, *Ultra-High-Accuracy X-ray Spectroscopy of Transition Metal Oxides and Rare Earths (SPECTRE) – FCT 31969*, 2019. [Online]. Available: http://libphys.pt/en/ultra-high-accuracy-x-ray-spectroscopy-of-transition-metal-oxides-and-rare-earths (Accessed: 01/14/2020).

[16] P. Amaro, J. P. Santos, A. Samouco, *et al.*, "Validation of the Geant4 Monte Carlo package for X-ray fluorescence spectroscopy in triaxial geometry," *Spectrochimica Acta - Part B Atomic Spectroscopy*, vol. 130, pp. 60–66, 2017, ISSN: 05848547. DOI: 10.1016/j.sab.2017.02.012. [Online]. Available: http://dx.doi.org/10.1016/j.sab.2017.02.012.

[17] T. Trojek and T. Čechák, "Use of MCNP code in energy dispersive X-ray fluorescence," *Nuclear Instruments and Methods in Physics Research, Section B: Beam Interactions with Materials and Atoms*, vol. 263, no. 1 SPEC. ISS. pp. 72–75, 2007, ISSN: 0168583X. DOI: 10.1016/j.nimb.2007.04.063.

[18] T. Schoonjans, L. Vincze, V. A. Solé, *et al.*, "A general Monte Carlo simulation of energy dispersive X-ray fluorescence spectrometers - Part 5: Polarized radiation, stratified samples, cascade effects, M-lines," *Spectrochimica Acta - Part B Atomic Spectroscopy*, vol. 70, pp. 10–23, 2012, ISSN: 05848547. DOI: 10.1016/j.sab.2012.03.011. [Online]. Available: http://dx.doi.org/10.1016/j.sab.2012.03.011.

[19] F. Salvat and M Fern, "PENELOPE – A Code System for Monte Carlo Simulation of Electron and Photon Transport," *PENELOPE, a code system for Monte Carlo simulation of electron and photon transport*, no. July, 2015, ISSN: 0931-041X.

[20] K. E. Dorfman, P. Wei, J. Liu, and R. Li, "Quantum interference and collisional dynamics in excited bounds states revealed by time-resolved pump-high-harmonic-generation-probe spectroscopy," *Optics Express*, vol. 27, no. 5, p. 7147, 2019, ISSN: 1094-4087. DOI: 10.1364/oe.27.007147.

[21] P. Amaro, B. Franke, J. J. Krauth, *et al.*, "Quantum interference effects in laser spectroscopy of muonic hydrogen, deuterium, and helium-3," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 92, no. 2, pp. 1–7, 2015, ISSN: 10941622. DOI: 10.1103/PhysRevA.92.022514.

[22] J. P. Desclaux, "A multiconfiguration relativistic DIRAC-FOCK program," *Computer Physics Communications*, vol. 9, no. 1, pp. 31–45, 1975, ISSN: 00104655. DOI: 10.1016/0010-4655(75)90054-5.

[23] M. F. Gu, "The flexible atomic code," *Canadian Journal of Physics*, vol. 86, no. 5, pp. 675–689, 2008, ISSN: 00084204. DOI: 10.1139/P07-197.

[24] P. Jönsson, G. Gaigalas, J. Bieroń, C. F. Fischer, and I. P. Grant, "New version: Grasp2K relativistic atomic structure package," *Computer Physics Communications*, vol. 184, no. 9, pp. 2197–2203, 2013, ISSN: 00104655. DOI: 10.1016/j.cpc.2013.02.016. [Online]. Available: http://dx.doi.org/10.1016/j.cpc.2013.02.016.

[25] *MCDFGME-Aim*, 2005. [Online]. Available: http://dirac.spectro.jussieu.fr/mcdf/mcdf_code/mcdfgme_aim.html (Accessed: 01/14/2020).

[26] S. Theodoridis, *Monte Carlo Methods*. 2015, pp. 707–744, ISBN: 9780128015223. DOI: 10.1016/b978-0-12-801522-3.00014-8.

[27] G. H. Yeoh and K. K. Yuen, "Additional Considerations in Field Modeling," in *Computational Fluid Dynamics in Fire Engineering*, G. H. Yeoh and K. K. Yuen, Eds., Elsevier, 2009, pp. 135–266. DOI: 10.1016/B978-0-7506-8589-4.00003-X. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B978075068589400003X.

[28] J. L. Vujic, *Monte Carlo - Sampling methods*, 2018. [Online]. Available: http://web.tecnico.ulisboa.pt/{~}mcasquilho/acad/theo/simul/Vujic.pdf (Accessed: 01/18/2020).

[29] A. Raulo, L. Marchini, G. Paternoster, *et al.*, "Ion beam (RBS) and XRF analysis of metal contacts deposited on CdZnTe and CdTe crystals," *IEEE Transactions on Nuclear Science*, vol. 58, no. 4 PART 2, pp. 1964–1971, 2011, ISSN: 00189499. DOI: 10.1109/TNS.2011.2145001.

[30] J. Apostolakis, A. Bagulya, D. Bernard, *et al.*, *Electromagnetic Physics*, 2019. [Online]. Available: https://geant4.web.cern.ch/collaboration/working{_}groups/electromagnetic (Accessed: 11/15/2020).

[31] P. Amaro, *Simulation of a XRF planar spectrometer with Geant4*, 2019. [Online]. Available: https://sites.google.com/fct.unl.pt/quant2019/home-page?authuser=0.

[32] *Geant4: G4Step.hh Source File*, 2020. [Online]. Available: http://www.apc.univ-paris7.fr/~franco/g4doxy/html/G4Step_8hh-source.html (Accessed: 01/14/2020).

[33] *Geant4: G4Track.hh Source File*, 2020. [Online]. Available: http://www.apc.univ-paris7.fr/~franco/g4doxy/html/G4Track_8hh-source.html (Accessed: 01/14/2020).

[34] *ROOT a Data analysis Framework | ROOT a Data analysis Framework*, 2020. [Online]. Available: https://root.cern.ch/ (Accessed: 01/14/2020).

[35] S. Richtberg, *Condition for the constructive interference of waves from a crystal film*, 2019. [Online]. Available: https://www.didaktik.physik.uni-muenchen.de/elektronenbahnen/en/elektronenbeugung/einfuehrung/bragg-bedingung.php (Accessed: 01/14/2020).

[36] M. Sanchez Del Rio, N. Perez-Bocanegra, X. Shi, V. Honkimäki, and L. Zhang, "Simulation of X-ray diffraction profiles for bent anisotropic crystals," *Journal of Applied Crystallography*, vol. 48, no. 1958, pp. 477–491, 2015, ISSN: 16005767. DOI: 10.1107/S1600576715002782. arXiv: 1502.03059.

[37] P. Amaro, "Thèse de doctorat en cotutelle - ÉTUDE DES TRANSITIONS INTERDITES DANS SYSTÈMES ATOMIQUES," Doctoral dissertation, 2011.

[38] *Relativistic Multiconfiguration Dirac-Fock Package*, 2005. [Online]. Available: http://dirac.spectro.jussieu.fr/mcdf/mcdf_theory/Mettec/mettec.html (Accessed: 01/14/2020).

[39] S. M. Blinder, "Basic Concepts of Self-Consistent-Field Theory," *American Journal of Physics*, vol. 33, no. 6, pp. 431–443, 1965, ISSN: 0002-9505. DOI: 10.1119/1.1971665.

[40] I. J. R. Aitchison and A. J. G. Hey, "Relativistic quantum mechanics," *Gauge Theories in Particle Physics*, no. November, 2004. DOI: 10.1887/0750309822/b1130v1c4.

[41] *Pauli Two-Component Formalism*, 2020. [Online]. Available: https://farside.ph.utexas.edu/teaching/qm/Quantum/node53.html (Accessed: 01/14/2020).

[42] G. Breit, "Dirac's equation and the spin-spin interactions of two electrons," *Physical Review*, vol. 39, no. 4, pp. 616–624, 1932, ISSN: 0031899X. DOI: 10.1103/PhysRev.39.616.

[43] M. Guerra, J. M. Sampaio, T. I. Madeira, *et al.*, "Theoretical and experimental determination of L -shell decay rates, line widths, and fluorescence yields in Ge," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 92, no. 2, pp. 1–9, 2015, ISSN: 10941622. DOI: 10.1103/PhysRevA.92.022507.

[44] C. O. Almbladh and A. L. Morales, "Theory of Auger core-valence-valence processes in simple metals. II. Dynamical and surface effects on Auger line shapes," *Physical Review B*, vol. 39, no. 6, pp. 3503–3516, 1989, ISSN: 01631829. DOI: 10.1103/PhysRevB.39.3503.

[45] J. P. Santos, M. C. Martins, A. M. Costa, F. Parente, and P. Indelicato, "Two-electron one-photon transition relativistic calculations for low-Z elements," *Nuclear Instruments and Methods in Physics Research, Section B: Beam Interactions with Materials and Atoms*, vol. 205, pp. 102–105, 2003, ISSN: 0168583X. DOI: 10.1016/S0168-583X(02)01989-4.

[46] L. N. Labzowsky, D. A. Solovyev, G. Plunien, and G. Soff, "Asymmetry of the Natural Line Profile for the Hydrogen Atom," *Physical Review Letters*, vol. 87, no. 14, p. 143 003, Sep. 2001, ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.87.143003. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.87.143003.

[47] K. Blum and H. Kleinpoppen, "Electron-photon angular correlation in atomic physics," *Physics Reports*, vol. 52, no. 4, pp. 203–261, 1979, ISSN: 03701573. DOI: 10.1016/0370-1573(79)90031-0.

[48] H. Nishimura, A. Danjo, and A. Takahashi, "Electron-photon angular correlations for electron impact excitation of Xe," *Journal of Physics B: Atomic and Molecular Physics*, vol. 19, no. 5, 1986, ISSN: 00223700. DOI: 10.1088/0022-3700/19/5/009.

[49] V. V. Balashov, A. N. G. Grzhimailo, and N. M. Kabachnik, *Polarization and Correlation Phenomena in Atomic Collisions: A Practical Theory Course*, 1st ed. 2000, ch. 3, pp. 107–131, ISBN: 9781475732283.

[50] V. V. Balashov, A. N. G. Grzhimailo, and N. M. Kabachnik, *Polarization and Correlation Phenomena in Atomic Collisions: A Practical Theory Course*, 1st ed. 2000, ch. 2, pp. 45–105, ISBN: 9781475732283.

[51] H. Michels, *Dislin*. [Online]. Available: https://www.dislin.de/index.html (Accessed: 11/15/2020).

[52] M. Guerra, F. Parente, P. Indelicato, and J. P. Santos, "Modified binary encounter Bethe model for electron-impact ionization," *International Journal of Mass Spectrometry*, vol. 313, pp. 1–7, 2012, ISSN: 13873806. DOI: 10.1016/j.ijms.2011.12.003. arXiv: 1306.2826. [Online]. Available: http://dx.doi.org/10.1016/j.ijms.2011.12.003.

[53] D. Pinheiro, P. Amaro*, M. Guerra, and J. P. Santos, *Polarization and angular distribution in fluorescence X-ray production*, 2020. [Online]. Available: `https://www.exsa.hu/?inh=6351`.

APPENDIX A

Developed Code

A.1 DCS Sampling Code

The following code was used to implement the rejection method in the already built main loop of the DCS simulation to sample the input spectrum.

Listing A.1: FORTRAN code section with the implementation of the rejection sampling method.

```
1     (...)
2     !Reset variables to check if generated energy is inside
3     !the spectrum. hit represents the y value of the spectrum
4     !and random the y value of the randomly generated point
5     hit = 0
6     random = max_inten
7
8     !check if the input spectrum is valid (intensity range
9     !grater then 0)
10    IF (delta_i .GT. 0) THEN
11      !repeat while the point is outside the spectrum
12      DO WHILE(hit < random)
13        !generate the x value of the random point
14        !ran1(idum) generates a uniform random number between
15        !0 and 1. delta is the window range along the x axis
16        !and start1 is the x value of the window's start
17        energy_t = ran1(idum) * delta + start1
18
19        !obtain the y value of the spectrum at x = energy_t
```

```
20          !from the spline interpolation. The value is saved
21          !in the hit variable
22          CALL splint_te(Energy_spec%lamda, &
23                  Energy_spec%intensity, &
24                  Energy_spec%intensity_two_deriv, &
25                  n_energy_spec, energy_t, hit)
26
27          !generate the y value of the random point
28          random = ran1(idum) * delta_i + min_inten
29        END DO
30      ELSE
31        !if there is no proper spectrum, generate a random
32        !energy, in this case following a uniform distribution.
33        energy_t = ran1(idum) * delta + start1
34      END IF
```

The code is commented for better understanding of this snippet. The energy generation used in the last ELSE clause could be reconfigured to generate a Bremsstrahlung background for example.

A.2 MCDF Calculation Python Script

The following code was used to automate the MCDF calculations for the radiative transitions, which in this case is configured for the rhodium electron configuration.

Listing A.2: Python script used to automate the MCDF radiative transition calculations. This script is configured for the rhodium electron configuration.

```
1
2  import os, math
3  import numpy as np
4  from astropy.modeling.models import Lorentz1D
5  import matplotlib.pyplot as plt
6  from decimal import Decimal
7  from operator import add
8
9  import psutil, subprocess, signal
10
11 try:
12     from subprocess import DEVNULL # py3k
13 except ImportError:
14     import os
15     DEVNULL = open(os.devnull, 'wb')
```

```
16
17 #Dictionary Containing the Configurations that Identify the
   Initial and Final State#
18 ##   label -> Subshell with the Electron that is Ionized or
   Fills the Hole *space*
19 ##         L1 Represents an Initial State in which the Hole
   is Created in the L1 Subshell. Also Represents a Final
   State in which a Hole is Filled by an
20 ##         Electron from the L1 Subshell.
21 ##   value -> [Electron Configuration, Double of the
   Corresponding Total Maximum Angular Momentum]
22 ##         The Electron Configuration is an LS Coupled
   Configuration in the Syntax Used in MCDF Input File (.f05)
23 ##         i.e. The Syntax Using Letters to Identify the
   Subshell's Orbital Angular Momentum (l) is (nl)e, with n
   Being the Principal Quantum Number, l Being the Subshell's
    Orbital Angular Momentum (s, p, d, f, ...) and e the
   Occupation Number of the Subshell. Check MCDF's Manual for
    the Syntax Using Numbers for the Subshell's Orbital
   Angular Momentum if Needed.
24 #Double of the Corresponding Total Maximum Angular Momentum
   (2*J, i.e. JJ) is the LS Coupled Total Angular Momentum.
   Used by MCDF to Calculate the jj Coupled Configurations.
   ################################################################

25
26 XRL_to_config = {"K":["(1s)1 (2s)2 (2p)6 (3s)2 (3p)6 (3d)10
   (4s)2 (4p)6 (4d)8 (5s)1", "10"],
27                  "L1":["(1s)2 (2s)1 (2p)6 (3s)2 (3p)6 (3d)
                       10 (4s)2 (4p)6 (4d)8 (5s)1", "10"],
28                  "L23":["(1s)2 (2s)2 (2p)5 (3s)2 (3p)6 (3d)
                       10 (4s)2 (4p)6 (4d)8 (5s)1", "12"],
29                  "M1":["(1s)2 (2s)2 (2p)6 (3s)1 (3p)6 (3d)
                       10 (4s)2 (4p)6 (4d)8 (5s)1", "10"],
30                  "M23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)5 (3d)
                       10 (4s)2 (4p)6 (4d)8 (5s)1", "12"],
31                  "M45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                       9 (4s)2 (4p)6 (4d)8 (5s)1", "14"],
32                  "N1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                       10 (4s)1 (4p)6 (4d)8 (5s)1", "10"],
```

```
33                      "N23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                           10 (4s)2 (4p)5 (4d)8 (5s)1", "12"],
34                      "N45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                           10 (4s)2 (4p)6 (4d)7 (5s)1", "10"],
35                      "O1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                           10 (4s)2 (4p)6 (4d)8", "8"]}
36
37
38  ########################################################
       Configuration Variables
       ########################################################
39  ##  procNum -> Number of Parallel Processes Used
                              ##  initialCap -> Initial State
       Subshell Cap, i.e. label from the Aug_to_configI
       Dictionary Where the Calculation Stops.
40  ##                If you Want to Calculte All Initial States
       Leave as Empty String
41  ##  Atom_Z -> Atomic Number Used in the Calculation. The
       Electron Number for the Initial and Final State is
       Calculated from this Value   ##  startLabel -> Full Start
       Label from Where to Start Calculating     ##  trigger ->
       Boolean Flag to Track if the startLabel State has been
       Reached
42  ##  initialStart, finalStart, jjiStart, mjjiStart, jjfStart,
       mjjfStart, eigviStart, eigvfStart -> Start State
       Identifyers Extracted
43  ##                from the startLabel variable
       ########################################################
44
45  configFile = "RadiativeConfig.txt"
46
47  procNum = 6
48  initialCap = "N1"
49  Atom_Z = 45 #Rh
50
51  startLabel = "K M23: jji=8, mjji=0, eigvi=4, jjf=6, mjjf=-6,
       eigvf=16"
52  trigger = False
53
54  initialStart = ""
```

```
55 finalStart = ""
56 jjiStart = 0
57 mjjiStart = 0
58 jjfStart = 0
59 mjjfStart = 0
60 eigviStart = 0
61 eigvfStart = 0
62
63 ######################################################
   Statistic Control Variables
   ######################################################
64 ##  calcStartTime -> Start Time of Calculation Script
                  ##  avgTimePerCycle -> Average Time Taken
    to Complete a Cycle (Configure Input File, Run MCDF, Read
    Output and Write to Output File) ##  avgTimePerCalc ->
   Average Time Taken Per Calculation (MCDF Process Average
   Uptime)
65 ##  totalCalcNum -> Number of Total Calculaions Performed (
   Used to Track the Averages)
66 ##  totalCycleNum -> Number of Total Script Cycles Performed
   (Used to Track the Averages and Multi-Threaded Performance
   )
67 ##  avgCalcPerH -> Average Calculations Per Hour. Calculated
   Using MCDF Process Uptime
68 ##  avgCyclePerH -> Average Script Cycles Per Hour
69 ##  totalSingleCalc -> Number of Single-Threaded Calculations
    Performed (Unused, but Tracked if Needed). These
   Calculations are Needed to Initialize Some State
   Parameters, such as the currEigvF and currEigvI. Also for
   the cases Where There is only one Calculation Left Before
   Changing Initial State Parameters
70 ##  totalMultiCalc -> Number of Multi-Threaded Calculations
   Performed (Unused, but Tracked if Needed)
71 ######################################################
72
73 calcStartTime = time.strftime('%x %X')
74 avgTimePerCycle = 0.0
75 avgTimePerCalc = 0.0
76 totalCalcNum = 0
77 totalCycleNum = 0
```

```
78 avgCalcPerH = 0.0
79 avgCyclePerH = 0.0
80 totalSingleCalc = 0
81 totalMultiCalc = 0
82
83 ##################################################
      Control Variables
      ##################################################
84 ##  file_name -> Input File Name for MCDF Executable
85 ##  currEigvI -> Current Maximum Initial State Eigenvalue (
      Loaded from MCDF Output, but Eigenvalue Equal to 1 is
      Guaranteed)
86 ##  currEigvF -> Current Maximum Final State Eigenvalue (
      Loaded from MCDF Output, but Eigenvalue Equal to 1 is
      Guaranteed)
87 ##################################################
88
89 file_name = '0_J0.f05'
90
91 currEigvI = 1
92 currEigvF = 1
93
94
95 ##################################################
96 ######                                      ######
97 ######           SCRIPT FUNCTIONS           ######
98 ######                                      ######
99 ##################################################
100
101
102 def Setup(verb = 0):
103
104     """
105         Setup the Required Variables and Files.
106
107         verb -> Verbose Level (Print Start State Variables)
108
109         1) Setup the Start State Variables According to
              startLabel Variable
```

```
110         2) Clear Output File (Please Use a Different File to
               Keep the Complete Calculations)
111         3) Setup All the MCDF Input Files for Radiative
               Calculation (One Per Process Used)
112     """
113
114     if(startLabel != ""):
115         initialStart = startLabel.split(":")[0].split(" ")
               [0].strip()
116         finalStart = startLabel.split(":")[0].split(" ")[1].
               strip()
117         jjiStart = int(startLabel.split("jji=")[1].split(",")
               [0])
118         mjjiStart = int(startLabel.split("mjji=")[1].split(",
               ")[0])
119         jjfStart = int(startLabel.split("jjf=")[1].split(",")
               [0])
120         mjjfStart = int(startLabel.split("mjjf=")[1].split(",
               ")[0])
121         eigviStart = int(startLabel.split("eigvi=")[1].split(
               ",")[0])
122         eigvfStart = int(startLabel.split("eigvf=")[1].split(
               ",")[0])
123     else:
124         initialStart = list(XRL_to_config.keys())[0]
125         finalStart = list(XRL_to_config.keys())[0]
126         jjiStart = int(XRL_to_config[initialStart][1])
127         mjjiStart = -jjiStart
128         jjfStart = int(XRL_to_config[finalStart][1])
129         mjjfStart = -jjfStart
130         eigviStart = 1
131         eigvfStart = 1
132
133     if(verb > 0):
134         print(initialStart)
135         print(finalStart)
136         print(jjiStart)
137         print(mjjiStart)
138         print(jjfStart)
139         print(mjjfStart)
```

```
140         print(eigviStart)
141         print(eigvfStart)
142 
143 
144     open("Rh_rawV1.txt", "w").close()
145 
146     for i in range(procNum):
147         replace_line("../core" + str(i) + "/" + file_name,
               10, "    radiative\n")
148         replace_line("../core" + str(i) + "/" + file_name,
               11, "       use_mcdfener=y relax=y y\n")
149 
150 
151     return (initialStart, finalStart, jjiStart, mjjiStart,
           jjfStart, mjjfStart, eigviStart, eigvfStart)
152 
153 def kill_proc_tree(pid, sig=signal.SIGTERM, include_parent=
       True,
154                    timeout=None, on_terminate=None):
155 
156     """
157         Kill a process tree (including grandchildren) with
               signal
158         "sig" and return a (gone, still_alive) tuple.
159         "on_terminate", if specified, is a callabck function
               which is
160         called as soon as a child terminates.
161     """
162 
163     assert pid != os.getpid(), "won't kill myself"
164 
165     try:
166         parent = psutil.Process(pid)
167     except psutil.NoSuchProcess:
168         print("Process Already Terminated.")
169         return (None, None)
170 
171     children = parent.children(recursive=True)
172     if include_parent:
173         children.append(parent)
```

```
174     for p in children:
175         try:
176             p.send_signal(sig)
177         except psutil.NoSuchProcess:
178             print("Already Terminated")
179             return (None, None)
180 
181     gone, alive = psutil.wait_procs(children, timeout=timeout
        ,
182                                     callback=on_terminate)
183     return (gone, alive)
184 
185 def replace_line(file_name, line_num, text):
186 
187     """
188         Replace a Line Inside a File.
189         Reads the File Into a Variable, Replace the Line in
              the Variable,
190         Rewrite the File with the Replaced Line.
191 
192         file_name -> Name of the File
193         line_num -> Number of the Line to Replace
194         text -> Text to Write in the Replaced Line
195     """
196 
197     lines = open(file_name, 'r').readlines()
198     lines[line_num] = text
199     out = open(file_name, 'w')
200     out.writelines(lines)
201     out.close()
202 
203 def StartMultiMCDF(procNum, initial, jji, mjji, eigvi, final,
    jjf, mjjf, eigvf):
204 
205     """
206         Start Multi-Threaded MCDF Calculations.
207 
208         initial -> Initial State Subshell (from
              Aug_to_configI Dictionary)
```

```
209         jji -> Double of the Initial State Total Angular
                Momentum
210         mjji -> Double of the Initial State Total Angular
                Momentum Projection
211         eigvi -> Initial State WaveFunction Eigenvalue (jj
                Coupled Configuration Mixing)
212         final -> Final State Subshells (from Aug_to_configF
                Dictionary)
213         jjf -> Double of the Final State Total Angular
                Momentum
214         mjjf -> Double of the Final State Total Angular
                Momentum Projection
215         eigvf -> Final State WaveFunction Eigenvalue (jj
                Coupled Configuration Mixing)
216
217         Returns an Array with the Processes Launched
218     """
219
220     theproc = []
221
222     for i in range(procNum):
223
224         replace_line("../core" + str(i) + "/" + file_name,
                20, "    nbeli=" + str(Atom_Z - 1) + " jjti=" +
                str(jji) + " mjjti=" + str(mjji) + "\n")
225         replace_line("../core" + str(i) + "/" + file_name,
                23, "c   " + XRL_to_config[initial][0] + " :\n")
226         replace_line("../core" + str(i) + "/" + file_name,
                25, "    neigv=" + str(eigvi) + " icmul=0 iprfgr
                =0\n")
227
228         replace_line("../core" + str(i) + "/" + file_name,
                21, "    nbelf=" + str(Atom_Z - 1) + " jjtf=" +
                str(jjf) + " mjjtf=" + str(mjjf) + "\n")
229         replace_line("../core" + str(i) + "/" + file_name,
                36, "c   " + XRL_to_config[final][0] + " :\n")
230         replace_line("../core" + str(i) + "/" + file_name,
                38, "    neigv=" + str(eigvf + i) + " icmul=0
                iprfgr=0\n")
231
```

```
232         tmp = psutil.Popen("mdfgme2010_1.exe", cwd = "../core
               " + str(i) + "/", shell = True, creationflags =
               subprocess.HIGH_PRIORITY_CLASS, stdout=DEVNULL)
233
234         theproc.append(tmp)
235
236     return theproc
237
238 def CheckTimeout0(theproc, totalCalcNum, totalMultiCalc,
       avgTimePerCalc):
239
240     """
241         Check for Timeout on the First Started Process.
242         The First Process is Used as Control for the
               Remaining Processes in Multi-Threaded Calculations
243
244         theproc -> The Started Processes Array to Manage for
               Timeout
245     """
246
247     startTime = time.time()
248
249     theproc[0].wait(timeout=25)
250
251     totalCalcNum += 1
252     totalMultiCalc += 1
253
254     proc0Time = time.time() - startTime
255
256     avgTimePerCalc += (proc0Time - avgTimePerCalc) /
           totalCalcNum
257
258     return (proc0Time, totalCalcNum, totalMultiCalc,
           avgTimePerCalc)
259
260 def CheckLiveProcs(theproc, procNum):
261
262     """
263         Check Which Processes are Still Running.
264
```

```
265         theproc -> The Started Processes Array to Check
266         procNum -> Maximum Number of Processes That may be
               Alive (Used When Close to the End of Final State
               Eigenvalue Loop)
267
268         Returns a Boolean Array Containing the Status of the
               Process Array (True -> Still Alive; False ->
               Terminated)
269     """
270
271     tmp = []
272
273     for i in range(1, procNum):
274         flag = False
275         try:
276             parent = psutil.Process(theproc[i].pid)
277             flag = True
278         except psutil.NoSuchProcess:
279             flag = False
280
281         tmp.append(flag)
282
283     return tmp
284
285 def UpdateDeadProcsStats(proc0Time, tmp, totalCalcNum,
       totalMultiCalc, avgTimePerCalc):
286
287     """
288         Update the Stats Variables with the Processes that
               Terminated While Waiting for Timeout.
289
290         proc0Time -> Time for the First Process to End (Used
               as Approximation for the Other Processes Time)
291         tmp -> Boolean Array Containing the Status of the
               Process Array (True -> Still Alive; False ->
               Terminated)
292     """
293
294     for i in range(sum([not i for i in tmp])):
295         totalCalcNum += 1
```

```
296         totalMultiCalc += 1
297         avgTimePerCalc += (proc0Time - avgTimePerCalc) /
               totalCalcNum
298
299     return (proc0Time, totalCalcNum, totalMultiCalc,
           avgTimePerCalc)
300
301 def CheckRemainingTimeout(tmp, theproc, totalCalcNum,
       totalMultiCalc, avgTimePerCalc):
302
303     """
304         Check the Remaining Process Array for Timeout (After
               the First Process is Terminated Without Timeout).
305
306         tmp -> Boolean Array Containing the Status of the
               Process Array (True -> Still Alive; False ->
               Terminated)
307         theproc -> The Started Processes Array to Manage for
               Timeout
308     """
309
310     for i in [i + 1 for i, x in enumerate(tmp) if x]:
311         tmpProcTime = time.time() - theproc[i].create_time()
312
313         if(tmpProcTime >= 25000):
314             killFlag, _ = kill_proc_tree(theproc[i].pid)
315
316             if(killFlag is not None):
317                 totalCalcNum += 1
318                 totalMultiCalc += 1
319                 avgTimePerCalc += (tmpProcTime -
                       avgTimePerCalc) / totalCalcNum
320         else:
321             try:
322                 tmpProcTime = time.time() - theproc[i].
                       create_time()
323                 tout = (25 - tmpProcTime)
324                 theproc[i].wait(timeout= max(tout, 0.01))
325
326                 totalCalcNum += 1
```

```
327                 totalMultiCalc += 1
328
329                 tmpProcTime = time.time() - tmpProcTime -
                        theproc[i].create_time()
330
331                 avgTimePerCalc += (tmpProcTime -
                        avgTimePerCalc) / totalCalcNum
332
333             except psutil.TimeoutExpired:
334                 print("TIMEOUT!!")
335                 killFlag, _ = kill_proc_tree(theproc[i].pid)
336
337                 if(killFlag is not None):
338                     totalCalcNum += 1
339                     totalMultiCalc += 1
340                     avgTimePerCalc += (25 - avgTimePerCalc) /
                            totalCalcNum
341
342     return (theproc, totalCalcNum, totalMultiCalc,
            avgTimePerCalc)
343
344 def KillRemainingProcs(theproc, procNum, totalCalcNum,
        totalMultiCalc, avgTimePerCalc):
345
346     """
347         Kill the Remaining Processes that Might Still be
                Alive.
348
349         theproc -> The Started Processes Array to Kill
350         procNum -> Maximum Number of Processes That may be
                Alive (Used When Close to the End of Final State
                Eigenvalue Loop)
351     """
352
353     for i in range(procNum):
354         killFlag, _ = kill_proc_tree(theproc[i].pid)
355
356         if(killFlag is not None):
357             totalCalcNum += 1
358             totalMultiCalc += 1
```

```
359             avgTimePerCalc += (25 - avgTimePerCalc) /
                    totalCalcNum
360
361     return (totalCalcNum, totalMultiCalc, avgTimePerCalc)
362
363 def WriteMultiMCDFOutput(saveFile, procNum, initial, jji,
    mjji, eigvi, final, jjf, mjjf, eigvf, currEigvI, currEigvF
    , trigger):
364
365     """
366         Write the Results of the Calculations from this Cycle
                in the Output File.
367
368         saveFile -> Output File Where to Save the MCDF
                Calculations Output
369         procNum -> Number of Calculation Outputs That Need to
                be Saved (Used When Close to the End of Final
                State Eigenvalue Loop)
370         initial -> Initial State Subshell (from
                Aug_to_configI Dictionary)
371         jji -> Double of the Initial State Total Angular
                Momentum
372         mjji -> Double of the Initial State Total Angular
                Momentum Projection
373         eigvi -> Initial State WaveFunction Eigenvalue (jj
                Coupled Configuration Mixing)
374         final -> Final State Subshells (from Aug_to_configF
                Dictionary)
375         jjf -> Double of the Final State Total Angular
                Momentum
376         mjjf -> Double of the Final State Total Angular
                Momentum Projection
377         eigvf -> Final State WaveFunction Eigenvalue (jj
                Coupled Configuration Mixing)
378     """
379
380     for i in range(procNum):
381         output = open("../core" + str(i) + "/0_j0.f06", 'r')
382
383         configMixi = []
```

```
384         configMixf = []
385         transitionData = []
386         configMixb = False
387         initialMix = True
388         initialCom = ""
389         finalCom = ""
390         commonb = True
391         initialMaxEigv = True
392         printLine = False
393 
394         for line in output:
395             if("The reference LS state for this calculation
                    results" in line):
396                 if(initialMaxEigv):
397                     currEigvI = int(line.strip().split("  ")
                            [1].strip().split(" ")[0].strip())
398                     initialMaxEigv = False
399                 else:
400                     currEigvF = int(line.strip().split("  ")
                            [1].strip().split(" ")[0].strip())
401 
402             if(configMixb):
403                 if("LSJ label:" in line):
404                     configMixb = False
405                 else:
406                     if(initialMix):
407                         configMixi.append(line)
408                     else:
409                         configMixf.append(line)
410 
411             if("List of jj configurations with a weight >=
                    0.01%" in line):
412                 configMixb = True
413 
414                 if(initialMix):
415                     configMixi = []
416                 else:
417                     configMixf = []
418 
```

```
419             if(initialMix and "NP         KE(a.u.)          E(a
                    .u.)           E(eV)          I(a.u.)" in line)
                    :
420                 initialMix = False
421
422             if("Common to all configurations" in line):
423                 if(commonb):
424                     initialCom = line
425                     commonb = False
426                 else:
427                     finalCom = line
428
429             if("transition" in line):
430                 printLine = True
431             if(printLine):
432                 transitionData.append(line.strip())
433             if(printLine and line.strip().split(' ')[0] == "
                    and"):
434                 printLine = False
435
436         output.close()
437
438         if(trigger):
439             if(not initialMix):
440                 print("writing...")
441                 saveFile.write(initial + " " + final + ": jji
                        =" + str(jji) + ", mjji=" + str(mjji) + ",
                         eigvi=" + str(eigvi) + ", jjf=" + str(jjf
                        ) + ", mjjf=" + str(mjjf) + ", eigvf=" +
                        str(eigvf) + "\n")
442                 saveFile.write("Initial state\n")
443                 saveFile.write(initialCom + "\n")
444
445                 for mix in list(filter(None, configMixi)):
446                     saveFile.write(mix + "\n")
447
448                 saveFile.write("Final state\n")
449                 saveFile.write(finalCom + "\n")
450
451                 for mix in list(filter(None, configMixf)):
```

```
452                     saveFile.write(mix + "\n")
453
454                 for transition in list(filter(None,
                        transitionData)):
455                     saveFile.write(transition + "\n")
456
457         eigvf += 1
458
459     return (eigvf, currEigvI, currEigvF)
460
461 def UpdateCycleStats(totalCycleNum, avgTimePerCycle,
        avgTimePerCalc, procNum):
462
463     """
464         Update the Statistic Variables each Cycle and Print
                them.
465     """
466
467     totalCycleNum += 1
468
469     cycleTime = time.time() - cycleStartTime
470
471     avgTimePerCycle += (cycleTime - avgTimePerCycle) /
            totalCycleNum
472
473     avgCalcPerH = 3600 / avgTimePerCalc
474     avgCyclePerH = 3600 / avgTimePerCycle
475
476     os.system('cls')
477
478     print("Total Number of Unique States Calculated: " + str(
            totalCalcNum))
479     print("Total Number of Cycles: " + str(totalCycleNum))
480     print("Average Time Per Unique State Calculation: " + str
            (avgTimePerCalc))
481     print("Average Time Per Cycle: " + str(avgTimePerCycle))
482     print("Average Thread Count: " + str(totalCalcNum /
            totalCycleNum) + " Threads. Current Maximum of " + str
            (procNum))
483     print("Calculation Start Time: " + calcStartTime)
```

```
484     print("Estimated Elapsed Time: " + time.strftime("%H:%M:%
           S", time.gmtime(totalCycleNum * avgTimePerCycle)) + "\
           n")
485     print("
           ##################################################")
486     print("##################### Averages Using Unique State
           Calculation Time #####################")
487     print("
           ##################################################")
488     print("Average Unique States Calculated Per Hour w/Single
            Thread: " + str(avgCalcPerH))
489     print("Average Unique States Calculated Per Hour w/Max
           Thread Count: " + str(procNum * avgCalcPerH))
490     print("Average Unique States Calculated Per Hour w/
           Current Avg Thread Count: " + str(((totalSingleCalc +
           totalMultiCalc) / totalCycleNum) * avgCalcPerH) + "\n"
           )
491     print("
           ##################################################")
492     print("################## Averages Using Script Cycle Time
            (more accurate?) ####################")
493     print("
           ##################################################")
494     print("Average Unique States Calculated Per Hour w/Single
            Thread: " + str(avgCyclePerH))
495     print("Average Unique States Calculated Per Hour w/Max
           Thread Count: " + str(procNum * avgCyclePerH))
496     print("Average Unique States Calculated Per Hour w/
           Current Avg Thread Count: " + str(((totalSingleCalc +
           totalMultiCalc) / totalCycleNum) * avgCyclePerH))
497
498     with open(configFile, "r") as f:
499         for line in f.readlines():
500             if(line[0] != "#" and "procNum = " in line):
501                 procNum = int(line.split("procNum = ")[1])
502
503     return (totalCycleNum, avgTimePerCycle, avgCalcPerH,
           avgCyclePerH, procNum)
504
505
```

```
506
507
508 ########################################################
509 ######                                            ######
510 ######              CORE SCRIPT                   ######
511 ######                                            ######
512 ########################################################
513
514 #Initial Setup
515
516 initialStart, finalStart, jjiStart, mjjiStart, jjfStart,
      mjjfStart, eigviStart, eigvfStart = Setup()
517
518
519
520 #Core Loop
521
522
523 for initial in XRL_to_config:
524  if(initial == initialCap):
525   break
526  for final in XRL_to_config:
527   if(final != initial and ((initial == initialStart and final
        == finalStart) or trigger) and (initial == "K" or (
       initial == "L1" and final != "K") or (initial == "L23"
       and final != "K" and final != "L1") or (initial == "M1"
       and final != "K" and final != "L1" and final != "L23")
       or (initial == "M23" and final != "K" and final != "L1"
       and final != "L23" and final != "M1") or (initial == "
       M45" and final != "K" and final != "L1" and final != "
       L23" and final != "M1" and final != "M23"))):
528    for jji in range(int(XRL_to_config[initial][1]), 0, -2):
529     if(jji == jjiStart or trigger):
530      saveFile = open("Rh_rawV1.txt", "a")
531      saveFile.write("\n")
532      for mjji in range(-jji, jji, 2):
533       if(mjji == mjjiStart or trigger):
534        eigvi = 1
535        while (eigvi <= currEigvI):
```

```
536         for jjf in range(int(XRL_to_config[final][1]), 0, -2)
                :
537          if(jjf == jjfStart or trigger):
538           for mjjf in range(-jjf, jjf, 2):
539            if(mjjf == mjjfStart or trigger):
540             eigvf = 1
541             while (eigvf <= currEigvF):
542              if(initial == initialStart and final ==
                    finalStart and jji == jjiStart and mjji ==
                    mjjiStart and eigvi == eigviStart and jjf ==
                    jjfStart and mjjf == mjjfStart and eigvf ==
                    eigvfStart):
543               trigger = True
544
545              if(trigger or (eigvi == 1 and eigvf == 1)):
546               theproc = []
547
548               if(eigvf <= currEigvF - procNum and procNum >
                     1):
549                theproc = StartMultiMCDF(procNum, initial, jji
                     , mjji, eigvi, final, jjf, mjjf, eigvf)
550
551               try:
552                proc0Time, totalCalcNum, totalMultiCalc,
                     avgTimePerCalc = CheckTimeout0(theproc,
                     totalCalcNum, totalMultiCalc,
                     avgTimePerCalc)
553
554                tmp = CheckLiveProcs(theproc, procNum)
555
556                proc0Time, totalCalcNum, totalMultiCalc,
                     avgTimePerCalc = UpdateDeadProcsStats(
                     proc0Time, tmp, totalCalcNum,
                     totalMultiCalc, avgTimePerCalc)
557
558                if(any(tmp)):
559                    theproc, totalCalcNum, totalMultiCalc,
                        avgTimePerCalc = CheckRemainingTimeout
                        (tmp, theproc, totalCalcNum,
                        totalMultiCalc, avgTimePerCalc)
```

```
560
561                 except psutil.TimeoutExpired:
562                  print("TIMEOUT!")
563
564                  totalCalcNum, totalMultiCalc, avgTimePerCalc
                        = KillRemainingProcs(theproc, procNum,
                        totalCalcNum, totalMultiCalc,
                        avgTimePerCalc)
565
566                  eigvf, currEigvI, currEigvF =
                        WriteMultiMCDFOutput(saveFile, procNum,
                        initial, jji, mjji, eigvi, final, jjf, mjjf
                        , eigvf, currEigvI, currEigvF, trigger)
567
568                 elif(eigvf > currEigvF - procNum and procNum >
                     1 and eigvf > 1):
569
570                  remaining = currEigvF - eigvf + 1
571
572                  theproc = StartMultiMCDF(remaining, initial,
                        jji, mjji, eigvi, final, jjf, mjjf, eigvf)
573
574                  try:
575                   proc0Time, totalCalcNum, totalMultiCalc,
                        avgTimePerCalc = CheckTimeout0(theproc,
                        totalCalcNum, totalMultiCalc,
                        avgTimePerCalc)
576
577                   tmp = CheckLiveProcs(theproc, remaining)
578
579                   proc0Time, totalCalcNum, totalMultiCalc,
                        avgTimePerCalc = UpdateDeadProcsStats(
                        proc0Time, tmp, totalCalcNum,
                        totalMultiCalc, avgTimePerCalc)
580
581                   if(any(tmp)):
582                    theproc, totalCalcNum, totalMultiCalc,
                         avgTimePerCalc = CheckRemainingTimeout(
                         tmp, theproc, totalCalcNum,
                         totalMultiCalc, avgTimePerCalc)
```

```
583
584                 except psutil.TimeoutExpired:
585                  print("TIMEOUT!")
586
587                  totalCalcNum, totalMultiCalc, avgTimePerCalc
                        = KillRemainingProcs(theproc, remaining,
                        totalCalcNum, totalMultiCalc,
                        avgTimePerCalc)
588
589                 eigvf, currEigvI, currEigvF =
                       WriteMultiMCDFOutput(saveFile, remaining,
                       initial, jji, mjji, eigvi, final, jjf, mjjf
                       , eigvf, currEigvI, currEigvF, trigger)
590
591                else:
592                 replace_line("../core0/" + file_name, 20, "
                          nbeli=" + str(Atom_Z - 1) + " jjti=" +
                       str(jji) + " mjjti=" + str(mjji) + "\n")
593                 replace_line("../core0/" + file_name, 23, "c
                        " + XRL_to_config[initial][0] + " :\n")
594                 replace_line("../core0/" + file_name, 25, "
                          neigv=" + str(eigvi) + " icmul=0 iprfgr
                       =0\n")
595
596                 replace_line("../core0/" + file_name, 21, "
                          nbelf=" + str(Atom_Z - 1) + " jjtf=" +
                       str(jjf) + " mjjtf=" + str(mjjf) + "\n")
597                 replace_line("../core0/" + file_name, 36, "c
                        " + XRL_to_config[final][0] + " :\n")
598                 replace_line("../core0/" + file_name, 38, "
                          neigv=" + str(eigvf) + " icmul=0 iprfgr
                       =0\n")
599
600                 tmp = psutil.Popen("mdfgme2010_1.exe", cwd = "
                       ../core0/", shell = True, creationflags =
                       subprocess.HIGH_PRIORITY_CLASS, stdout=
                       DEVNULL)
601
602                 theproc.append(tmp)
603
```

```
604                 try:
605                  startTime = time.time()
606
607                  theproc[0].wait(timeout=25)
608
609                  totalCalcNum += 1
610                  totalSingleCalc += 1
611
612                  proc0Time = time.time() - startTime
613
614                  avgTimePerCalc += (proc0Time - avgTimePerCalc
                        ) / totalCalcNum
615                 except psutil.TimeoutExpired:
616                  print("TIMEOUT!")
617                  kill_proc_tree(theproc[0].pid)
618
619                  totalCalcNum += 1
620                  totalSingleCalc += 1
621
622                  avgTimePerCalc += (25 - avgTimePerCalc) /
                        totalCalcNum
623
624                 eigvf, currEigvI, currEigvF =
                        WriteMultiMCDFOutput(saveFile, 1, initial,
                        jji, mjji, eigvi, final, jjf, mjjf, eigvf,
                        currEigvI, currEigvF, trigger)
625
626                 totalCycleNum, avgTimePerCycle, avgCalcPerH,
                        avgCyclePerH, procNum = UpdateCycleStats(
                        totalCycleNum, avgTimePerCycle,
                        avgTimePerCalc, procNum)
627
628                else:
629                 print(initial + " " + final + ": jji=" + str(
                        jji) + ", mjji=" + str(mjji) + ", eigvi=" +
                        str(eigvi) + ", jjf=" + str(jjf) + ", mjjf="
                         + str(mjjf) + ", eigvf=" + str(eigvf))
630                 eigvf += 1
631
632         eigvi += 1
```

```
633     saveFile.close()
```

The Auger transitions were calculated with a similar script, where the configuration dictionary (line 26 in listing A.2) and changing some of the lines that configure the input file in the `StartMultiMCDF` function (line 203 in listing A.2). The modified code is in listing A.3.

Listing A.3: Modifications to the Python script used to automate the MCDF Auger transition calculations. This script is configured for the rhodium electron configuration.

```
1
2  ######################################     Dictionary
       Containing the Configurations that Identify the Final
       State     ######################################
3  ##   label -> Subshell with the Electron that Fills the Hole
       *space* Subshell with the Electron that is Ejected Instead
        of Emiting a Photon ##         L1 L1 Represents a Final
       State in which the Hole is Filled with an Electron from
       the L1 Subshell and Another Electron
4  ##         from the L1 Subshell is Ejected as the Auger
       Electron.
5  ##   value -> [Electron Configuration, Double of the
       Corresponding Total Maximum Angular Momentum]
6  ##         The Electron Configuration is an LS Coupled
       Configuration in the Syntax Used in MCDF Input File (.f05)
        i.e. The Syntax Using Letters to Identify the Subshell's
       Orbital Angular Momentum (l) is (nl)e, with n Being the
       Principal Quantum Number, l Being the Subshell's Orbital
       Angular Momentum (s, p, d, f, ...) and e the Occupation
       Number of the Subshell. Check MCDF's Manual for the Syntax
        Using Numbers for the Subshell's Orbital Angular Momentum
        if Needed.
7  ##         Double of the Corresponding Total Maximum Angular
       Momentum (2*J, i.e. JJ) is the LS Coupled Total Angular
       Momentum. Used by MCDF to Calculate
8  ##         The jj Coupled Configurations.
       ######################################################
9
10 Aug_to_configF = {"L1 L1":["(1s)2 (2p)6 (3s)2 (3p)6 (3d)10 (4
       s)2 (4p)6 (4d)8 (5s)1","9"],
11                      "L1 L23":["(1s)2 (2s)1 (2p)5 (3s)2 (3p)6
                            (3d)10 (4s)2 (4p)6 (4d)8 (5s)1","13"
                          ],
```

```
12                  "L1 M1":["(1s)2 (2s)1 (2p)6 (3s)1 (3p)6
                        (3d)10 (4s)2 (4p)6 (4d)8 (5s)1","11"
                        ],
13                  "L1 M23":["(1s)2 (2s)1 (2p)6 (3s)2 (3p)5
                         (3d)10 (4s)2 (4p)6 (4d)8 (5s)1","13"
                        ],
14                  "L1 M45":["(1s)2 (2s)1 (2p)6 (3s)2 (3p)6
                         (3d)9 (4s)2 (4p)6 (4d)8 (5s)1","15"
                        ],
15                  "L1 N1":["(1s)2 (2s)1 (2p)6 (3s)2 (3p)6
                        (3d)10 (4s)1 (4p)6 (4d)8 (5s)1","11"
                        ],
16                  "L1 N23":["(1s)2 (2s)1 (2p)6 (3s)2 (3p)6
                         (3d)10 (4s)2 (4p)5 (4d)8 (5s)1","13"
                        ],
17                  "L1 N45":["(1s)2 (2s)1 (2p)6 (3s)2 (3p)6
                         (3d)10 (4s)2 (4p)6 (4d)7 (5s)1","11"
                        ],
18                  "L1 O1":["(1s)2 (2s)1 (2p)6 (3s)2 (3p)6
                        (3d)10 (4s)2 (4p)6 (4d)8","9"],
19
20                  "L23 L23":["(1s)2 (2s)2 (2p)4 (3s)2 (3p)
                        6 (3d)10 (4s)2 (4p)6 (4d)8 (5s)1","13
                        "],
21                  "L23 M1":["(1s)2 (2s)2 (2p)5 (3s)1 (3p)6
                         (3d)10 (4s)2 (4p)6 (4d)8 (5s)1","13"
                        ],
22                  "L23 M23":["(1s)2 (2s)2 (2p)5 (3s)2 (3p)
                        5 (3d)10 (4s)2 (4p)6 (4d)8 (5s)1","15
                        "],
23                  "L23 M45":["(1s)2 (2s)2 (2p)5 (3s)2 (3p)
                        6 (3d)9 (4s)2 (4p)6 (4d)8 (5s)1","17"
                        ],
24                  "L23 N1":["(1s)2 (2s)2 (2p)5 (3s)2 (3p)6
                         (3d)10 (4s)1 (4p)6 (4d)8 (5s)1","13"
                        ],
25                  "L23 N23":["(1s)2 (2s)2 (2p)5 (3s)2 (3p)
                        6 (3d)10 (4s)2 (4p)5 (4d)8 (5s)1","15
                        "],
```

```
26                  "L23 N45":["(1s)2 (2s)2 (2p)5 (3s)2 (3p)
                       6 (3d)10 (4s)2 (4p)6 (4d)7 (5s)1","13
                       "],
27                  "L23 O1":["(1s)2 (2s)2 (2p)5 (3s)2 (3p)6
                        (3d)10 (4s)2 (4p)6 (4d)8","11"],
28
29                  "M1 M1":["(1s)2 (2s)2 (2p)6 (3p)6 (3d)10
                        (4s)2 (4p)6 (4d)8 (5s)1","9"],
30                  "M1 M23":["(1s)2 (2s)2 (2p)6 (3s)1 (3p)5
                        (3d)10 (4s)2 (4p)6 (4d)8 (5s)1","13"
                       ],
31                  "M1 M45":["(1s)2 (2s)2 (2p)6 (3s)1 (3p)6
                        (3d)9 (4s)2 (4p)6 (4d)8 (5s)1","15"
                       ],
32                  "M1 N1":["(1s)2 (2s)2 (2p)6 (3s)1 (3p)6
                       (3d)10 (4s)1 (4p)6 (4d)8 (5s)1","11"
                       ],
33                  "M1 N23":["(1s)2 (2s)2 (2p)6 (3s)1 (3p)6
                        (3d)10 (4s)2 (4p)5 (4d)8 (5s)1","13"
                       ],
34                  "M1 N45":["(1s)2 (2s)2 (2p)6 (3s)1 (3p)6
                        (3d)10 (4s)2 (4p)6 (4d)7 (5s)1","11"
                       ],
35                  "M1 O1":["(1s)2 (2s)2 (2p)6 (3s)1 (3p)6
                       (3d)10 (4s)2 (4p)6 (4d)8","9"],
36
37                  "M23 M23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       4 (3d)10 (4s)2 (4p)6 (4d)8 (5s)1","13
                       "],
38                  "M23 M45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       5 (3d)9 (4s)2 (4p)6 (4d)8 (5s)1","17"
                       ],
39                  "M23 N1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)5
                        (3d)10 (4s)1 (4p)6 (4d)8 (5s)1","13"
                       ],
40                  "M23 N23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       5 (3d)10 (4s)2 (4p)5 (4d)8 (5s)1","15
                       "],
```

```
41                  "M23 N45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       5 (3d)10 (4s)2 (4p)6 (4d)7 (5s)1","13
                       "],
42                  "M23 O1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)5
                        (3d)10 (4s)2 (4p)6 (4d)8","11"],
43
44                  "M45 M45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       6 (3d)8 (4s)2 (4p)6 (4d)8 (5s)1","17"
                       ],
45                  "M45 N1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6
                        (3d)9 (4s)1 (4p)6 (4d)8 (5s)1","15"
                       ],
46                  "M45 N23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       6 (3d)9 (4s)2 (4p)5 (4d)8 (5s)1","17"
                       ],
47                  "M45 N45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       6 (3d)9 (4s)2 (4p)6 (4d)7 (5s)1","15"
                       ],
48                  "M45 O1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6
                        (3d)9 (4s)2 (4p)6 (4d)8","13"],
49
50                  "N1 N1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6
                       (3d)10 (4p)6 (4d)8 (5s)1","9"],
51                  "N1 N23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6
                        (3d)10 (4s)1 (4p)5 (4d)8 (5s)1","13"
                       ],
52                  "N1 N45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6
                        (3d)10 (4s)1 (4p)6 (4d)7 (5s)1","11"
                       ],
53                  "N1 O1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6
                       (3d)10 (4s)1 (4p)6 (4d)8","9"],
54
55                  "N23 N23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       6 (3d)10 (4s)2 (4p)4 (4d)8 (5s)1","13
                       "],
56                  "N23 N45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                       6 (3d)10 (4s)2 (4p)5 (4d)7 (5s)1","13
                       "],
57                  "N23 O1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6
                        (3d)10 (4s)2 (4p)5 (4d)8","11"],
```

```
58
59                      "N45 N45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)
                           6 (3d)10 (4s)2 (4p)6 (4d)6 (5s)1","9"
                           ],
60                      "N45 O1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6
                            (3d)10 (4s)2 (4p)6 (4d)7","9"]}
61
62 ####################################     Dictionary
   Containing the Configurations that Identify the Initial
   State     ####################################
63 ##   label -> Subshell with the electron that was Ionized
                ##         L1 Represents an Initial State in
   which the Hole is Created by Ionization in the L1 Subshell
   . The Process of Ionization is not Taken Into Account. To
   Take that Into Account Choose the Aplicable States from
   the Results Output File.
64 ##   value -> [Electron Configuration, Double of the
   Corresponding Total Maximum Angular Momentum]
65 ##         The Electron Configuration is an LS Coupled
   Configuration in the Syntax Used in MCDF Input File (.f05)
    i.e. The Syntax Using Letters to Identify the Subshell's
   Orbital Angular Momentum (l) is (nl)e, with n Being the
   Principal Quantum Number, l Being the Subshell's Orbital
   Angular Momentum (s, p, d, f, ...) and e the Occupation
   Number of the Subshell. Check MCDF's Manual for the Syntax
    Using Numbers for the Subshell's Orbital Angular Momentum
    if Needed.
66 ##         Double of the Corresponding Total Maximum Angular
   Momentum (2*J, i.e. JJ) is the LS Coupled Total Angular
   Momentum. Used by MCDF to Calculate
67 ##         The jj Coupled Configurations.
68 ######################################################
69
70 Aug_to_configI = {"K":["(1s)1 (2s)2 (2p)6 (3s)2 (3p)6 (3d)10
   (4s)2 (4p)6 (4d)8 (5s)1", "10"],
71                   "L1":["(1s)2 (2s)1 (2p)6 (3s)2 (3p)6 (3d)
                        10 (4s)2 (4p)6 (4d)8 (5s)1", "10"],
72                   "L23":["(1s)2 (2s)2 (2p)5 (3s)2 (3p)6 (3d)
                        10 (4s)2 (4p)6 (4d)8 (5s)1", "12"],
```

```
73                      "M1":["(1s)2 (2s)2 (2p)6 (3s)1 (3p)6 (3d)
                           10 (4s)2 (4p)6 (4d)8 (5s)1", "10"],
74                      "M23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)5 (3d)
                           10 (4s)2 (4p)6 (4d)8 (5s)1", "12"],
75                      "M45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                           9 (4s)2 (4p)6 (4d)8 (5s)1", "14"],
76                      "N1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                           10 (4s)1 (4p)6 (4d)8 (5s)1", "10"],
77                      "N23":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                           10 (4s)2 (4p)5 (4d)8 (5s)1", "12"],
78                      "N45":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                           10 (4s)2 (4p)6 (4d)7 (5s)1", "10"],
79                      "O1":["(1s)2 (2s)2 (2p)6 (3s)2 (3p)6 (3d)
                           10 (4s)2 (4p)6 (4d)8", "8"]}
80
81  (...)
82
83  def StartMultiMCDF(procNum, initial, jji, mjji, eigvi, final,
        jjf, mjjf, eigvf):
84
85      """
86          Start Multi-Threaded MCDF Calculations.
87
88          initial -> Initial State Subshell (from
                Aug_to_configI Dictionary)
89          jji -> Double of the Initial State Total Angular
                Momentum
90          mjji -> Double of the Initial State Total Angular
                Momentum Projection
91          eigvi -> Initial State WaveFunction Eigenvalue (jj
                Coupled Configuration Mixing)
92          final -> Final State Subshells (from Aug_to_configF
                Dictionary)
93          jjf -> Double of the Final State Total Angular
                Momentum
94          mjjf -> Double of the Final State Total Angular
                Momentum Projection
95          eigvf -> Final State WaveFunction Eigenvalue (jj
                Coupled Configuration Mixing)
96
```

```
97          Returns an Array with the Processes Launched
98      """
99      os.system("taskkill /f /im  mdfgme2010_1.exe")
100
101     theproc = []
102
103     for i in range(procNum):
104
105         replace_line("../core" + str(i) + "/" + file_name,
               21, "    nbeli=" + str(Atom_Z - 1) + " jjti=" +
               str(jji) + " mjjti=" + str(mjji) + "\n")
106         replace_line("../core" + str(i) + "/" + file_name,
               24, "c   " + Aug_to_configI[initial][0] + " :\n")
107         replace_line("../core" + str(i) + "/" + file_name,
               26, "    neigv=" + str(eigvi) + " icmul=0 iprfgr
               =0\n")
108
109         replace_line("../core" + str(i) + "/" + file_name,
               22, "    nbelf=" + str(Atom_Z - 2) + " jjtf=" +
               str(jjf) + " mjjtf=" + str(mjjf) + "\n")
110         replace_line("../core" + str(i) + "/" + file_name,
               37, "c   " + Aug_to_configF[final][0] + " :\n")
111         replace_line("../core" + str(i) + "/" + file_name,
               39, "    neigv=" + str(eigvf + i) + " icmul=0
               iprfgr=0\n")
112
113         tmp = psutil.Popen("mdfgme2010_1.exe", cwd = "../core
               " + str(i) + "/", shell = True, creationflags =
               subprocess.HIGH_PRIORITY_CLASS, stdout=DEVNULL)
114
115         theproc.append(tmp)
116
117     return theproc
```

ANNEX I

Scientific Conferences

The results shown in section 5.3.1 were also presented by Professor Doctor Pedro Amaro in the International Initiative on X-ray Fundamental Parameters 2020 meeting with the title "Polarization and Angular Distribution in Fluorescence X-ray Production" [53].