POLITECNICO DI MILANO

&

ÉCOLE CENTRALESUPÉLEC

Radiation damage in mono-atomic and bi-atomic materials: Simulation of neutron irradiation by ion irradiation

Julien Bouix

Laurea Magistrale

Materials Engineering and Nanotechnology

DEPARTMENT OF MATERIALS ENGINEERING

September 2021

Abstract

The direct simulation of the neutron irradiation damage during the whole lifespan of the in-core components in a nuclear facility can be time-consuming. The literature being quite rare in this field, we investigated the possibility to simulate this neutron irradiation damage with ion irradiation that can cause similar damage in a reduced duration. Conditions for their possible equivalence will be defined through the projectile energy loss distribution between phonon generation, ionization dissipation and future stored energy. Python codes working in symbiosis with the renown TRIM software were developed in order to simulate the neutron path inside any mono and bi-atomic materials. The interest of this master thesis is mainly focused on the creation of the defects (vacancies and interstitials), their spacial distribution and the energy loss of the neutron or ion projectiles. The recovery processes will not be considered and their modelization and programming with molecular dynamic codes can constitute an excellent second step to couple with the codes created in the present.

1 Estratto

Al fine di valutare il danno arrecato a un materiale da una data fluenza di irradiazione di neutroni in un impianto nucleare, gli esperimenti di laboratorio possono richiedere anni per fornire informazioni pertinenti. L'idea alla base di questa tesi è di prevedere se l'irradiazione ionica in laboratorio può causare danni simili a quelli dell'irradiazione di neutroni. In effetti, il tempo tipico impiegato per gli esperimenti sugli ioni tende ad essere molto inferiore a quello per i neutroni (da anni a settimane). Uno dei miei obiettivi era quantificare la riduzione del tempo o, in un approccio leggermente diverso, determinare la fluenza dell'irradiazione ionica richiesta, con ioni a una data energia e massa, al fine di ottenere una generazione di danno simile a una data fluenza di neutroni irradiazione.

Ho utilizzato il rinomato programma TRIM sviluppato da James F.Ziegler per effettuare il calcolo del percorso ionico eseguito all'interno del materiale e per determinare i difetti creati come i vuoti. Tuttavia, TRIM non può valutare il percorso in caso di irradiazione di neutroni. Un codice di neutroni non era prontamente disponibile e mi servivano solo le geometrie più semplici, quindi decisi di sviluppare da solo un codice di neutroni, sul quale avrei potuto avere il controllo completo. Affinché i lettori possano comprendere il mio lavoro nel modo più efficiente, descriverò prima i presupposti fisici e i calcoli utilizzati in questo lavoro e solo dopo descriverò in modo approfondito cosa sta facendo il mio codice e come ho cercato di raggiungere i miei obiettivi utilizzando principalmente il calcolatore TRIM e i file di output da esso generati. Sono stati creati quattro codici per utilizzare TRIM in modo automatizzato. Sono stati creati due codici distinti per l'irradiazione di ioni e neutroni e questi due codici sono ulteriormente suddivisi per analizzare separatamente i materiali bersaglio monoatomici e biatomici.

Le analisi dell'output di TRIM sono state effettuate, principalmente, leggendo il file COLLISION.txt generato da TRIM. Questo file è l'output più completo e dettagliato di TRIM. Ecco perché ci ho dedicato del tempo cercando di districare il suo prezioso ma complesso contenuto. Questo file può contenere centinaia di migliaia di righe che riportano tutte le collisioni avvenute nel materiale per uno e un solo proiettile. È suddiviso, per comprensibili ragioni, in tabelle. Ogni "Primary Knock-on Atom", definito più avanti nella tesi, ha una tabella corrispondente nella quale sono riportate tutte le collisioni che si sono verificate a causa dell'attuale Primary Knock-on Atom (PKA) e delle sue sotto cascate. La suddivisione non va oltre quel punto. È stato fatto un grande sforzo per cercare di analizzare correttamente il file COLLISION.txt, non sono ancora riuscito a trovare un algoritmo di validità del tutto generale in grado di districare, dal file COLLISION, l'intero, spesso molto ramificato, albero delle collisioni. Sono stati sviluppati e testati un totale di tre algoritmi, basati su metodi diversi. Sono chiamati algoritmo per lunghezza, per intero ordinamento a cascata e per fononi.

Uno sforzo particolare è stato posto nel tentativo di ricostruire la struttura dell'albero delle collisioni, attraverso l'"algoritmo per ordinamento a cascata intero", utilizzando il file COLLISION.txt per poi ricalcolare le diverse perdite energetiche di interesse. Avrebbe dato risultati molto accurati, ma nonostante la persistenza e la determinazione che ho dovuto estrarre questa architettura di sistema di ramificazione dal file COLLISION.txt, purtroppo non sono riuscito a riuscirci. Il problema

era principalmente il modo in cui TRIM sceglieva di visualizzare le diverse righe nel file. Le analisi ei test sono comunque riportati sia nella sezione 6 che in 7. La sezione 6 offre addirittura un possibile inquadramento teorico, con notazioni sicuramente perfettibili, che potrebbe essere utilizzato per affrontare un complesso sistema di ramificazione con architettura ad albero.

Poiché l'algoritmo quasi perfetto non sarebbe stato creato, ho dovuto sviluppare altri metodi e approssimazioni per analizzare la distribuzione della perdita di energia del proiettile all'interno del materiale bersaglio. Sebbene contengano approssimazioni in alcuni casi discutibili, consentono sempre di ottenere la distribuzione delle perdite di energia che qui interessa. Alla fine, l'algoritmo che ho usato per tutte le irradiazioni era l'algoritmo per lunghezza. Anche se i valori assoluti potrebbero essere criticati, ci si aspetta che almeno i valori relativi siano significativi, il che rende più o meno valido l'approccio comparativo.

Alla fine, i risultati e la loro interpretazione saranno considerati e descritti. Gli irraggiamenti di neutroni e ioni su diversi materiali sonno stati studiati. Le caratteristiche di irraggiamento neutronico sono accuratamente dettagliate nel quadro fisico. Gli ioni considerati in questa tesi di laurea sono Ni a 4 MeV e Au a 12 MeV perché questi ioni a queste energie specifiche possono essere prodotti in alcune strutture e sono già stati studiati [3]. Viene preso in considerazione anche il caso significativamente diverso dell'irradiazione da parte dei protoni. Per quanto riguarda i materiali da indagare, abbiamo scelto con il mio relatore di tesi di laurea il ferro puro come bersaglio monoatomico e tre diversi ossidi, vale a dire Al_2O_3, ZrO_2 e Y_2O_3 per la loro popolarità e l'ampia varietà di applicazioni. Abbiamo considerato al massimo materiali bi-atomici ma il mio codice potrebbe essere generalizzato a qualsiasi materiale n-atomico.

Il mio lavoro è stato incentrato sulla creazione dei difetti dovuti all'irradiazione. L'evoluzione dei difetti nel materiale non sarà studiata. Ciò include le importanti nozioni di mobilità dei difetti, annientamento tra un posto vacante e un interstiziale, coalescenza dei difetti (nella creazione di faglie impilate per esempio). L'intera analisi dei processi di ripristino non verrà eseguita poiché richiederebbe una grande quantità di lavoro extra e codice dinamico molecolare che è molto impegnativo in termini di risorse di calcolo.

Saranno realizzate tabelle per confrontare diverse variabili tra le diverse combinazioni di materiale bersaglio e particella incidente. Indagheremo la produzione di difetti nei materiali per le diverse irradiazioni. Verranno studiate le perdite di energia della particella incidente e determinata la loro distribuzione tra energia immagazzinata, generazione di fononi e dissipazione di ionizzazione. L'energia immagazzinata considerata è costituita solo dall'energia di distorsione attorno ai difetti interstiziali e verranno fornite le giustificazioni. La dissipazione di ionizzazione è l'energia persa agli elettroni (potenza di arresto elettronico). Il legame, ma non l'equivalenza, tra la generazione di fononi e la perdita di energia al nucleo (potere di arresto nucleare) verrà spiegato più in dettaglio. Concluderò dando alcuni spunti e idee per sviluppare potenzialmente ulteriormente questo progetto. Inoltre, in appendice sono riportate informazioni dettagliate su due dei quattro codici sviluppati per questa tesi. Questi codici sorgente potrebbero essere riutilizzati per ulteriori lavori.

Contents

2 Introduction

In order to assess the damage done to a material by a given fluence of neutron irradiation in a nuclear facility, the laboratory experiments may take years to give relevant information. The idea behind this thesis is to predict whether or not ion irradiation in laboratory can give similar damage as neutron irradiation. In fact, the typical time taken for ion experiments tend to be much lower than the ones for neutron (from years to weeks) [1][5] One of my goal was to quantify the time reduction or, in a slightly different approach, to determine the fluence of ion irradiation required, with ions at a given energy and mass, in order to get a similar damage generation as a given fluence of neutron irradiation.

I used the renown TRIM program developed by James F.Ziegler to make the calculation of the ion path taken inside the material and to determine the defects created such as vacancies. Yet, TRIM cannot assess the path in the case of neutron irradiation. A neutron code was not readily available and I needed only the simplest geometries, I therefore decided to develop by myself a neutron code, on which I could have complete control. For the readers to understand my work in the most efficient manner, I will first describe the physical assumptions and calculations used in this work and only then will I described in depth what my code is doing and how I tried to reach my goals by using mainly the TRIM calculator and output files generated by it. Four codes were created in order to use TRIM in an automated way. Two distinct codes were created for the ion and neutron irradiation and these two codes are further divided in order to analyse separately mono-atomic and bi-atomic target materials.

The analyses of the TRIM output were done, mostly, by reading the COLLISION.txt file generated by TRIM. This file is the most complete and detailed output of TRIM. That is why I spent time on it trying to disentangle its valuable but complex content. This file can contain hundreds of thousands of lines reporting all collisions that happened in the material for one and only one projectile. It is divided, for understandable reasons, in tables. Each "Primary Knock-on Atom", defined later in the thesis, has a corresponding table in which are reported all the collisions that aroused due to the current Primary Knock-on Atom (PKA) and its sub cascades. The subdivision does not go beyond that point. A great effort was invested into trying to analyse correctly the COLLISION.txt file, I was yet unable to find an algorithm of completely general validity able to disentangle, from the COLLISION file, the whole, often highly branched, tree of collisions.. A total of three algorithms, based on different methods, were developed and tested (see section 7). They are called algorithm by length, by whole cascade sorting and by phonons.

A particular effort was put into trying to reconstruct through the "algorithm by whole cascade sorting" the structure of the collision tree by using the COLLISION.txt file in order to then recalculate the different energy losses of interest. It would have given very accurate results but despite the persistence and determination I had to extract this branching system architecture from the COLLISION.txt file, I could unfortunately not succeed. The problem mainly was the way TRIM chose to display the different lines in the file. The analyses and the tests are nonetheless reported in both section 6 and 7. The section 6 even offers a possible theoretical framework, with surely perfectible notations, that could be used in order to approach a complex branching system with a tree architecture.

Since the close-to-perfect algorithm would not be created, I had to develop other methods and approximations to analyse the energy loss distribution of the projectile inside the target material. Although they contain approximations which are in some cases questionable, they always allow to obtain the distribution of the energy losses which are of interest here. In the end, the algorithm I used for all irradiations was the algorithm by length. Even if the absolute values could be criticized, at least the relative values are expected to be meaningful which makes the comparative approach more or less valid.

Eventually, the results and their interpretation will be considered and described. Neutron and ion irradiations on different materials will be studied. The neutron irradiation characteristics are thoroughly detailed in the physical framework. The ions considered in this master thesis are Ni at 4 MeV and Au at 12 MeV because these ions at these specific energies can be produced in some facilities and were already studied [3]. The significantly different case of irradiation by protons is also considered. Concerning the materials to be investigated, we chose with my master thesis supervisor pure iron as a mono-atomic target and three different oxides, namely Al_2O_3, ZrO_2 and Y_2O_3 due to their popularity and wide variety of applications. We considered at maximum bi-atomic materials but my code could be generalized to any n-atomic material.

My work is focused on the creation of the defects due to irradiation. The evolution of the defects in the material will not be studied. This includes the important notions of defects mobility, annihilation between a vacancy and an interstitial, coalescence of defects (in the creation of stacking faults for instance). The whole recovery processes analysis will not be performed since it would require a great amount of extra work and molecular dynamic code that are very demanding in term of computation resources.

Tables will be made in order to compare several variables between the different combinations of target material and incident particle. We will investigate the production of defects in the materials for the different irradiations. The energy losses of the incident particle will be studied and their distribution between stored energy, phonons generation and ionization dissipation determined. The stored energy considered consists only of the distortion energy around interstitial defects and the justifications will be given. The ionization dissipation is the energy lost to the electrons (electronic stopping power). The link, but not equivalence, between the phonon generation and the energy loss to nucleus (nuclear stopping power) will be explained in more detail. I will conclude by giving some hints and ideas to potentially further develop this project. Furthermore, in the appendix are reported detailed information on two of the four codes developed for this thesis. These source codes could be used again for further work.

3 Physical framework

3.1 The neutron irradiation

We consider a material of thickness d, typically pure iron at the beginning for it will become difficult with compounds, submitted to an irradiation of neutron as we can find in a nuclear reactor. The material is in the right half-space (x$\geq$0) and projectiles are always incident at the (0,0,0) coordinates. The first consideration to make is that the neutron beam is not collimated because the flux of neutron in a nuclear reactor typically is isotropic. The neutron will enter the material with a random angle $\alpha \in [0, \frac{\pi}{2}]$ with the vector normal of the surface $\vec{n_s}$ and an angle $\beta \in [0, 2\pi]$ around the same $\vec{n_s}$ with, for $\beta = 0$, the vector of the neutron velocity before impact being in the plane formed by the $\vec{n_s}$ and $\vec{e_Y}$. The set of vector ($\vec{e_X}$,$\vec{e_Y}$,$\vec{e_Z}$) being the orthogonal basis of reference with $\vec{e_X}$ pointing towards the depth of the material. Then, unlike the monoenergetical flux of ions in a laboratory, the neutrons will have a spectrum with different energies. I considered for my thesis a distribution of Maxwell-Boltzmann type with a most probable energy of 1MeV and an average of 2MeV (Figure 1) [6]. This is a simple way to approximate the spectrum of neutrons generated by the fission of uranium but will be a sufficient entry for the development of the codes. The neutron spectrum can be modified by the user at the beginning of the code leaving its core unaltered.

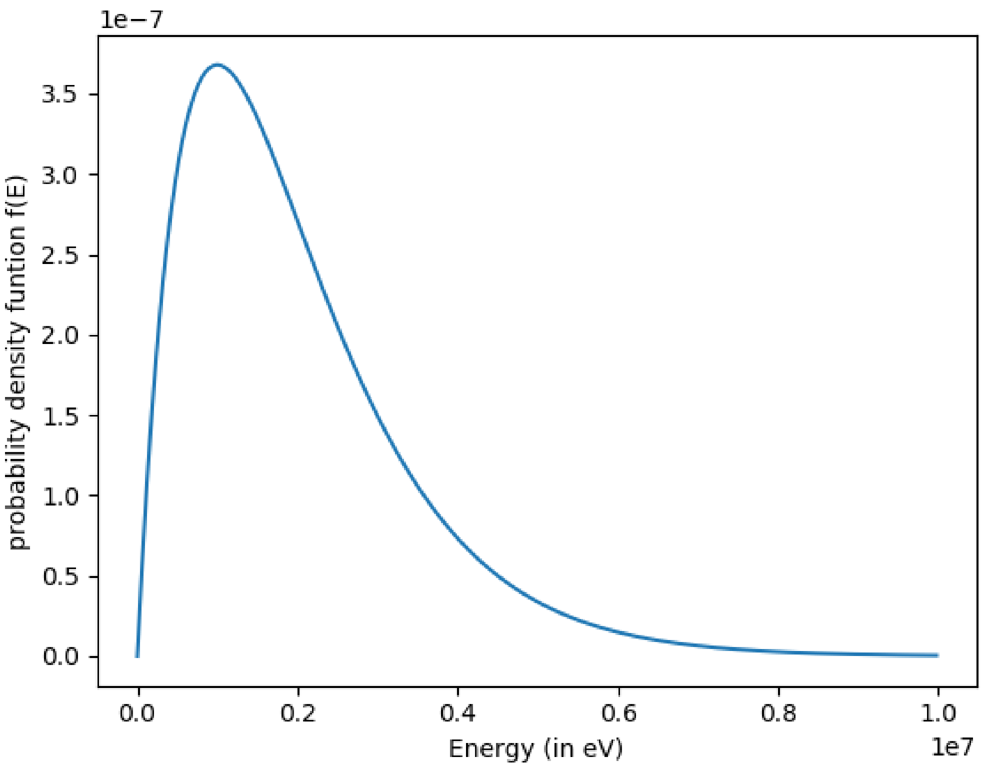


Figure 1: Probability density function of a Maxwell-Boltzmann type with a parameter a= $\sqrt{\frac{E_p}{m_n}}$

Once the incoming neutron flux and energy spectrum are dealt with, we need to consider what happens inside the target material. Unlike the case of ions in which we considered both electronic and nuclear stopping power for both primary and recoil cascades, for neutron irradiation, only recoils cascades contains electronic stopping power because the neutron itself do not interact with the electronic cloud. The neutrons are only stopped due to the successive losses of energy through collisions with nucleus. That is why the primary cascade only contains nuclear stopping power as opposed to the recoil cascades (in the current case: cascades caused by ions thus respecting the physical mechanisms described in the subsection "The ion irradiation").

A major difference between ion and neutron irradiation is the fact that the probability of interaction between ions and target atoms is much greater than for neutron, phenomena due mainly to their respective size. The mean free path for neutrons inside a material is of the order of centimeters while for ions it is closer to the nanometer scale. This means that for ion irradiation, the damage is clustered near the surface at a depth at the micrometer scale, like in the example shown above for Ni ions at 4 MeV. But for neutron irradiation, around each location where collisions between the neutron and the set of PKA occured, we will have a localized cluster of damage. The PKA (Primary Knock-on Atom) are the set of atom initially at a lattice position that are ejected from it directly by the projectile itself (the set of PKA are the vertices drawing the approximate path of the projectile inside the target material). More details are given on the PKA definition later on in the report in section 6. These clusters are spaced from a typical distance of centimeters while the cluster size is closer to the nanometer to micrometer scale depending on the energy transferred to the corresponding PKA. This is a consideration to keep in mind when comparing the number of vacancies later on because defects will have an inherent different distribution between the two cases, a concentrated one at the near surface for ion irradiation and a more diffuse on for the neutron irradiation.

These probabilities of interaction between the neutron and any target nucleus is usually referred to as "cross section". The microscopic cross section σ, expressed in terms of surface, usually barns ($1b = 10^{-24} cm^2$) represents the effective target area of one and only one nucleus. The greater its value, the greater its size and obviously the greater the probability of interaction with the incident neutron. The microscopic cross-section, despite its easy understanding, is not relevant at a greater scale because we also consider the number density of targets which multiplies the microscopic cross section in order to give the macroscopic cross section Σ_t. This formula is based on the assumption that shadowing effects are negligible. The commonly used Σ_t, expressed in terms of cm^{-1}, can be understood like a linear attenuation coefficient, it considers the whole effective target area of all the nuclei in the material thickness. In fact, a collimated neutron beam intensity I_0 inside a mono-atomic material is attenuated following an exponential decreasing function with the coefficient of attenuation being Σ_t, with x being the penetration depth inside the target material [8][9]:

$$I(x) = I_0 e^{-\Sigma_t \mathrm{x}} \tag{1}$$

The obtained profile represents the flux of particles which have not yet interacted. It coincides with the actual flux only if interaction implies absorption. Otherwise, the actual flux includes scattered particles, not accounted for by Eq. 1. More precisely, in our case, we consider elastic scattering only so the neutron does not stop its path upon the first collision by absorption but it will bounce to a certain direction defined later on in this report. This means that the equation 1 will be used in order to determine probabilistically the distance of travel before the next collision on the axis of motion of the neutron, which is not necessarily the X-axis. That point is really important and is the fundamental basis of the later developed neutron code. This code, written using the Python language, is available to the reader in the second part of the Appendix.

3.2 The ion irradiation

The ion irradiation is directly computed with the TRIM program. We consider the same material as in the case of neutron irradiation but this time submitted to irradiation of an ion specimen that can be produced in a laboratory. Some commonly used ions in the field of irradiated material science in laboratory are Ni ions at 4 MeV and Au ions at 12 MeV. In laboratory, the ion beam is collimated and I assume that the angle between the axis of irradiation and the normal of the material surface is null, which is not the case for neutrons as we will see.

As the ion enters inside the material it will collides with atoms and progressively lose energy due to collisions, it is the so-called nuclear stopping power. Between these collisions the ion interacts with the electronic cloud of the material and will give energy to these electrons that will become excited, the ion loses energy due to this mechanism and it is called the electronic stopping power. Both nuclear and electronic stopping power may be expressed in eV/A, so in energy in electron-volt lost per Angstrom travel inside the material, we call it in that case linear stopping power $S = -\frac{dE}{dx}$. The stopping power is highly dependent on the density of the material in which it happens so a usual variable called mass stopping power may be defined as $S_M = \frac{S}{\rho}$ in $MeV/(mg/cm)$ and ρ being the density of the material in mg/cm^3. In figure 1. is represented the electronic and nuclear stopping power (respectively SE and SN) in the case of Au ion inside an Al_2O_3 compound target. We see that for ions at high energies, the electronic stopping poser is the main mechanism of deceleration of the ions. Yet, its importance compared to the nuclear stopping power diminishes as the ion energy decreases. This is due to the cross section dependence on the particle energy. When the ion has a high velocity, its cross section with the target atoms is smaller than at lower velocities. In the latter case, the probability of interaction with nucleus increases and reaches a peak (0,37 MeV in the example shown at Figure 1.). Then, both decreases to zero, with the nuclear stopping power remaining dominant.

All the target atoms of the material that undergo a collision with the ion are called PKA for Primary Knocked-off Atoms, because they are the first to be knocked off from their respective lattice position. The set of all PKA is the primary cascade. Yet, because these PKA generally have enough energy to displace other atoms as they, after the primary collisions, also travel now inside the material. The set of all atoms being knocked-off by PKA are called SKA for Secondary Knocked-off Atoms. From here, we can generalize that a nKA is an atom that was knocked-off from its position due to a collision with a (n-1)KA. For $n \geq 2$, we call the

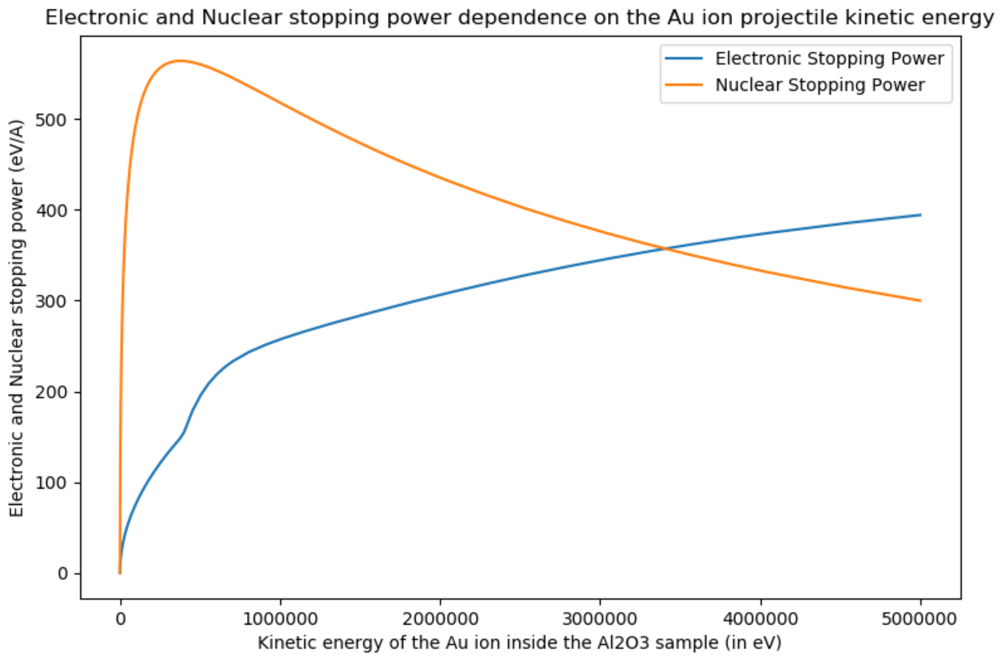


Figure 2: Comparison of the electronic and nuclear stopping power in the particular case of Au ion inside Al_2O_3

corresponding cascades recoil cascades. For instance, in the image on the left side is the representation of the path of a Ni ion initially at 4 MeV (entering the material from the center left). It forms a curve that is a succession of line connecting the dot of each collision. On the right, with an identical ion, we get a different result with in white the primary cascade but in green the recoil cascades. The graphs were computed on SRIM.

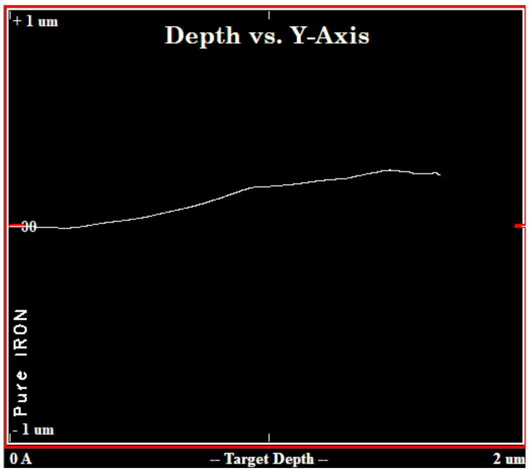


Figure 3: Primary cascade of a Ni ion at 4 MeV inside pure iron

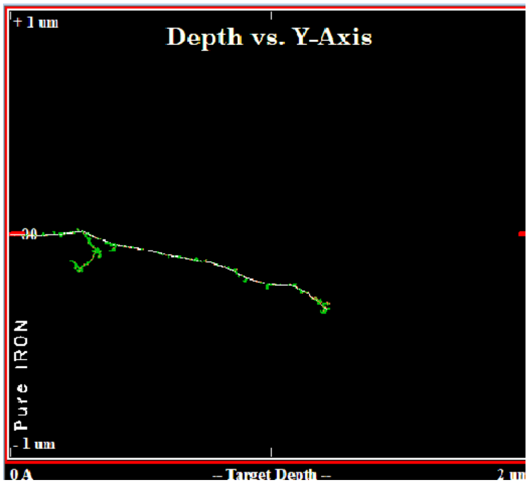


Figure 4: Primary cascade and recoil cascades of a Ni ion at 4 MeV inside pure iron

4 The neutron Code

This code considers as an input the number of neutrons we want to send into the target, the material composition and physical properties and the energy spectrum distribution that every neutron follows. The output needs to be the total number of vacancies the neutron and recoil cascades created inside the material in order to assess the extent of the damage.

4.1 Description of the physics behind the code for single neutron

Each neutron are calculated separately, one-by-one, in an independent way, meaning that the structural defects caused by the nth neutron has no influence on the damage done by the (n+1)th neutron. This approximation arises because TRIM, and also my code, perform Monte Carlo simulations, in which each projectile is treated independently, and only the primary damage is considered, the successive evolution of defects not being modeled. Therefore each projectile impinges on a material which is in the pristine state. And the incidence position is irrelevant, therefore all the projectiles impinge in the same position, with either the same direction or with variable directions. Each neutron will be sent inside the material in a randomly different manner in terms of direction of incidence and energy. In order to simulate the isotropic flux of neutron inside a nuclear reactor, I considered that the incoming neutron can come from any direction from the solid angle with $\theta \in [0, \frac{\pi}{2}]$ and $\phi \in [0, 2\pi]$. The energy will be randomly chosen from a Maxwell-Boltzmann probability density function f(E) is defined with a most probable energy of Ep = 1 MeV and an average of 2 MeV. f(E) is expressed, considering the latter constraint as [6]:

$$f(E) = \frac{2^{\frac{3}{2}}}{\sqrt{\pi}} \frac{E\sqrt{m_n}}{E_p^{\frac{3}{2}}} e^{-\frac{E}{E_p}} \qquad (2)$$

The term m_n represents the mass of one neutron. In the figure 1 is displayed f(E) for $E \in [0, 10MeV]$. For now on, let call E_0 the initial energy of the neutron just before colliding with the target material.

The calculation of the macroscopic cross-section is worth mentioning. I took the data available in [12] of the microscopic cross-section for thermal neutrons (neutrons at 0.025eV) and neutrons at 1 MeV. In between, a simple linear interpolation was made and above 1 MeV, the value of the cross section doesn't change anymore. We can write the microscopic cross section as:

$$\sigma(E) = \begin{cases} \sigma_1 + \frac{\sigma_2 - \sigma_1}{E_2 - E_1}(E - E_1) & ifE < E_2 \\ \sigma_2 & ifE \geq E_2 \end{cases} \tag{3}$$

with σ_1 and σ_2 respectively being the microscopic cross section at $E_1 = 0.025eV$ and $E_2 = 1MeV$. Then for mono compound target, the macroscopic cross section at a particle energy E is defined as [12] :

$$\Sigma(E) = \frac{\rho N_A}{M}\sigma(E) \tag{4}$$

with ρ and M respectively being the density and the molar mass of the target in g/cm^3 and g/mol, and N_A the Avogadro's number. For multi compound target, we need to consider the chemical formula of the compound like Al_2O_3 and consider the stoiechiometry by adding a weight in front of the microscopic cross section ($\frac{2}{5}$ for Al and $\frac{3}{5}$ for O), with M_c being this time the molecular mass of the compound and ρ_c the density of the compound:

$$\Sigma(E) = \frac{\rho_c N_A}{M_c}(\frac{2}{5}\sigma_{Al}(E)) + \frac{3}{5}\sigma_O(E)) \tag{5}$$

The collision between the neutron and the target nucleus will occur after a path travelled following a exponential law with the macroscopic cross-section as λ parameter according to equation (1). The collisions taking place are considered to be elastic and between hard spheres. Here is displayed in figure 5. as a schematics a collision between a neutron and a nucleus in the case of hard spheres:

When the distance until the next collision is calculated using the macroscopic cross section, we now randomly simulate what the impact parameter b$\in [-R_1 - R_2, R_1 + R_2]$, with R_1 and R_2 being the respective radii of the neutron and the target nucleus. I considered a uniform distribution of the neutron center of mass in the disc of radius $R_1 + R_2$. At first, I wanted to take into account the channeling effect we can observe in a crystallographic material by increasing the probability of the neutron to collide with the nucleus at its edge by considering a parabolic distribution probability. I then realized it had no physical sense since the size order of the neutron is much smaller than the typical spacing between two atoms in a crystal, that means that there is no such thing as a channeling effect between a neutron and a crystal. moreover, the distribution of b cannot follow a uniform distribution since the probability of collision is uniform along the whole area of the nucleus and not its radius. That is why we consider a crown 2πb*db at each impact

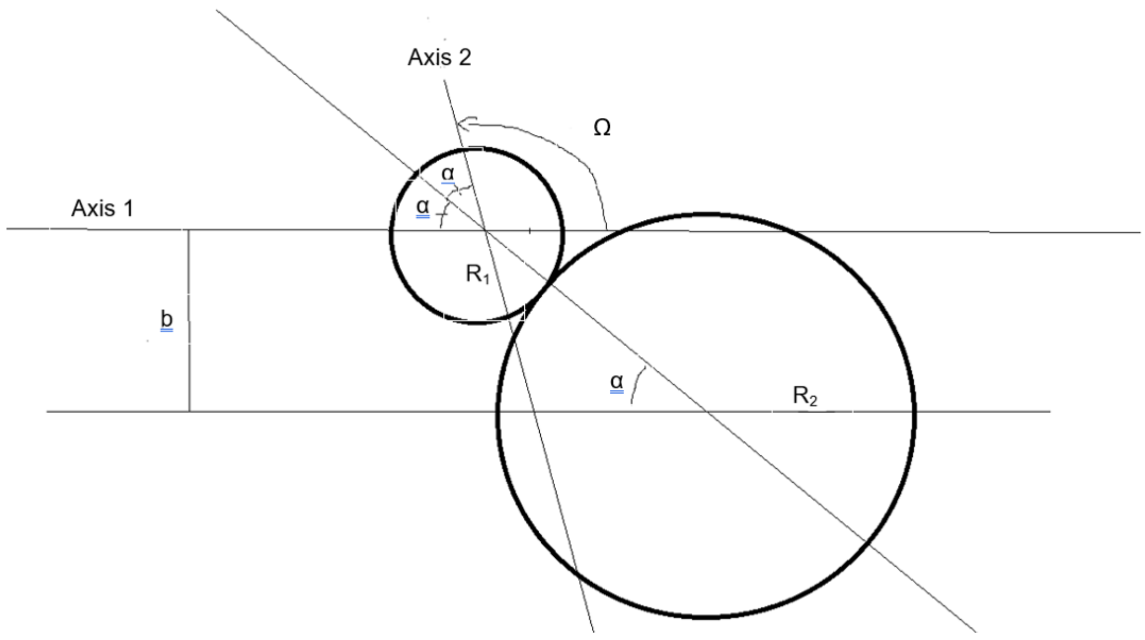


Figure 5: Collision between two hard spheres and the main useful distances and angles to be defined

parameter b. The probability to collide more at the edges of the nucleus is still true but not for the same reasons.

We consider at this stage an angle $\beta \in [0, 2\pi]$ located around the axe 1. It means that the infinitesimal probability that the neutron collides with the nucleus with an impact parameter b and at an angle β is defined as :

$$dP = 2\pi b * dbd\beta \tag{6}$$

We can see that dP increases linearly with b which means that the probability of having an impact close to the edges of the nucleus is higher than the probability of having a head-on collision. Once b is determine we can write the equation of energy and quantity of (linear) momentum conservation in order to express the transmitted energy and the quantity of momentum p_{Af} and p_{Bf} after the collision with the angles α, Ω and the quantity of momentum of the neutron before impact p_{Ai}:

$$\begin{cases} p_{Ai} = p_{Af}cos\upsilon + p_{Bf}cos\alpha \\ 0 = p_{Af}sin\upsilon - P_{Bf}sin\alpha \end{cases} \tag{7}$$

The solution of this system of two equations simply is :

$$\begin{cases} p_{Af} = p_{Ai}(cos\upsilon + \frac{sin\upsilon}{tan\alpha})^{-1} \\ \\ p_{Bf} = \sqrt{\frac{m_B}{m_A}}\sqrt{p_{Ai}^2 - p_{Af}^2} \end{cases} \tag{8}$$

Some geometrical considerations enables us to determine the expression of α and υ as function of b, R_1 and R_2:

$$\begin{cases} \alpha = sin^{-1}(\frac{b}{R_1+R_2}) \\ \upsilon = \pi - 2\alpha \end{cases} \tag{9}$$

An important fact is also that because the masses involved in the collision are different, there are some limits is the amount of transferable energy T_{max}. In the context of a two body elastic collision, the equation that governs the amount of transferred energy $T_{transferred}$ to the target particle is written below. In our particular case, we studied as interaction potential a two body rigid hard spheres. It has been derived by considering the velocity of the particle in the reference of the lab and its center of mass and can be found in some lectures like [2].

$$T_{transferred} = E_{Ai}\frac{m_A m_B}{(m_A+m_B)^2}\frac{1-cos\upsilon}{2} \tag{10}$$

Then my code determine the unit vector that indicates the orientation of the axe labeled 2 given the axe labeled 1 in Figure 5 respectively being the direction of the neutron after and before the collision. I will call them axe_{in} and axe_{out}.

Because the neutron will cause several collisions in its travel inside the material, I will define a formalism to better understand how my code is working. Let for the nth collision call the axe_{in} and axe_{out} respectively axe_n and axe_{n+1}. It is enough because we consider no deviation of the neutron path between the nth and (n+1)th collisions, making the axe_{out} of the nth collision being the axe_{in} of the (n+1)th collision. For instance, if the beam was collimated with an axe of incidence of -$\vec{n_S}$, the axe_{in}=axe_1 would be (1,0,0) in the ($\vec{e_X}$,$\vec{e_Y}$,$\vec{e_Z}$) reference basis. In the case of the isotropic flux of neutron, the axe_1 is random but still pointing towards the right half-space but with an random angle. Let call the energy of the neutron after the nth collision E_n and the transferred energy to the target nucleus for the same collision $T_{transferred,n}$:

$$\forall n \geq 1, \quad E_n = E_{n-1} - T_{transferred,n} \tag{11}$$

The idea is that between the nth and (n+1)th collision, I compute the macroscopic cross section at the energy of the neutron in order to get the exponential distribution law for the distance traveled before the next impact. Then I run the collision function described above and we start again until one of the three following criteria is fulfilled:

1. The neutron energy becomes smaller than the threshold displacement energy E_d.
2. The neutron is back scattered to negative X.
3. The neutron is transmitted through the thickness L of the material.

Below are depicted in a 3D graph the path taken by the neutron inside an iron block for the three different cases described above. Each line breaking corresponds to a collision.

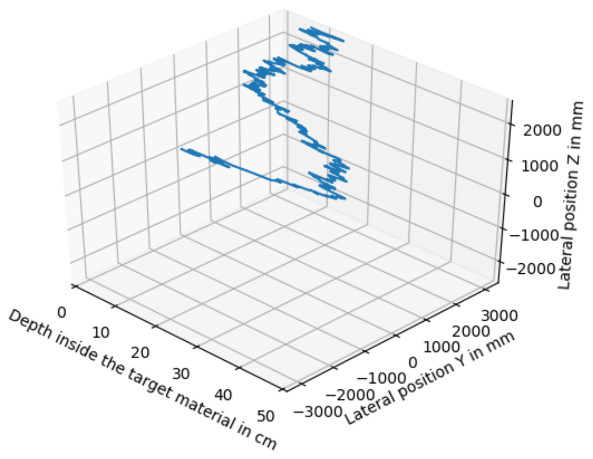


Figure 6: Path taken by a neutron inside iron with d=50 cm that ended due to criteria 1 after 742 collisions.

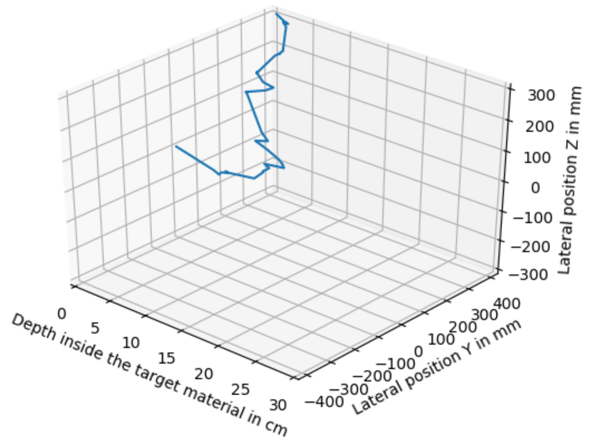


Figure 7: Path taken by a neutron inside iron with d=30 cm that ended due to criteria 2 after 46 collisions.

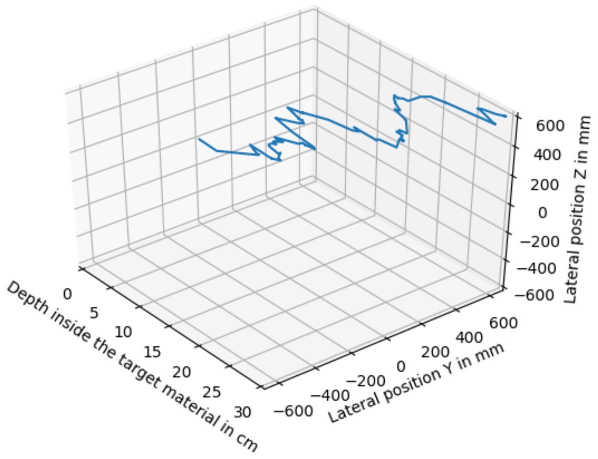


Figure 8: Path taken by a neutron inside iron with d=30 cm that ended due to criteria 3 after 104 collision.

4.2 Description of the physics behind the code for multiple neutrons

By using the neutron code this way, we only get the simulation of one neutron traveling inside the material and we also get the list of the atomic displacement information caused by it. The point now is to use it many time successively to get statistically relevant information.

We can for instance run the code for N=50000 neutrons and observe the number of atoms knocked-off from their lattice position (=the number of vacancies N_V created). The result for a 30 cm thick iron block is shown in the figure 9 and Figure 10 (only zoomed in order to see the higher order of collisions) below.

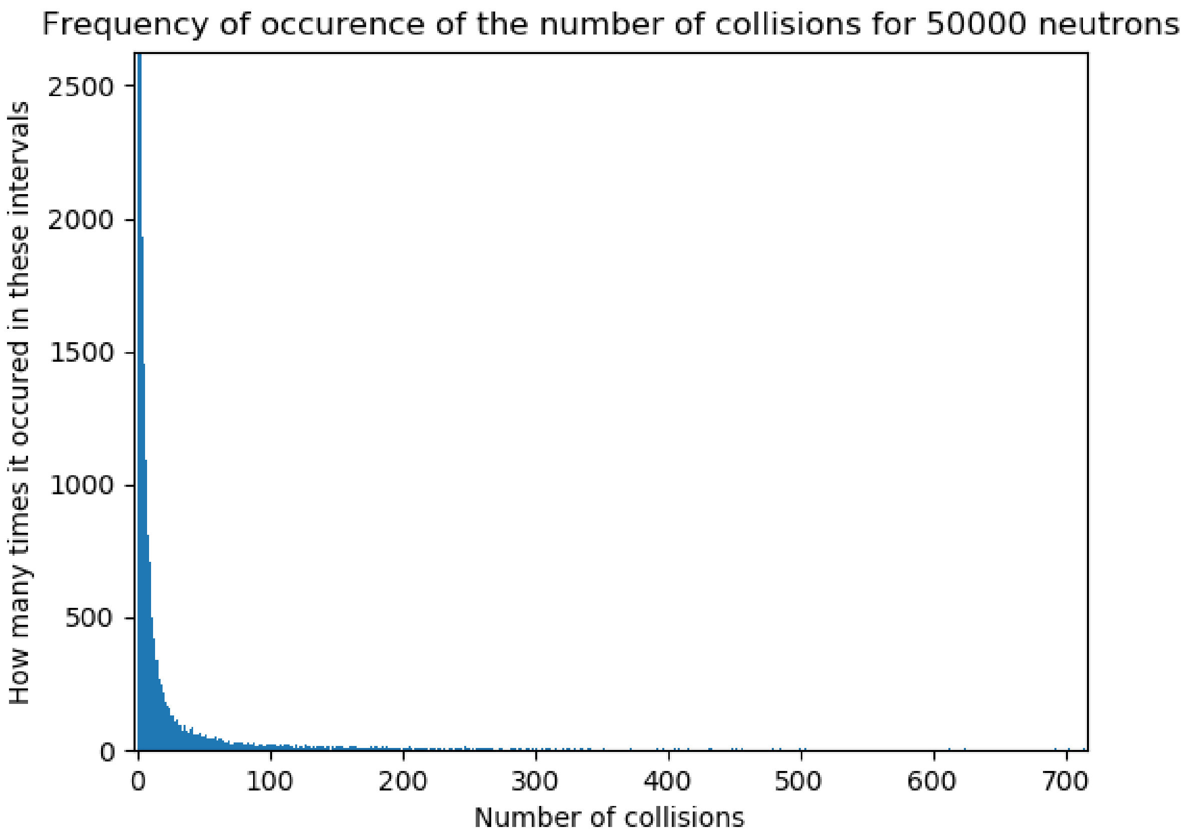


Figure 9: Number of occurence of N_V for collisions in Iron and N=50 000 neutrons

We see that the most probable N_V is very small, only one or two collisions. It seems then that the probability of getting more collisions diminishes exponentially. The extreme peak at low N_V correspond to the cases where the neutron is back scattered (criteria n°2) after only a few collisions. Then as the neutron collides more and more, it may go deeper inside the material where it will increase its probability to cause further damage through collisions. Yet, it may still be back scattered even after a reasonable number of collisions. The opposite, being transmission (criteria n°3), may also happen if the collisions cause the neutron to travel towards positive X more frequently. The highest N_V is usually reached when the neutron lacks energy to cause further collision (criteria n°1) and its remaining energy is lost as phonons.

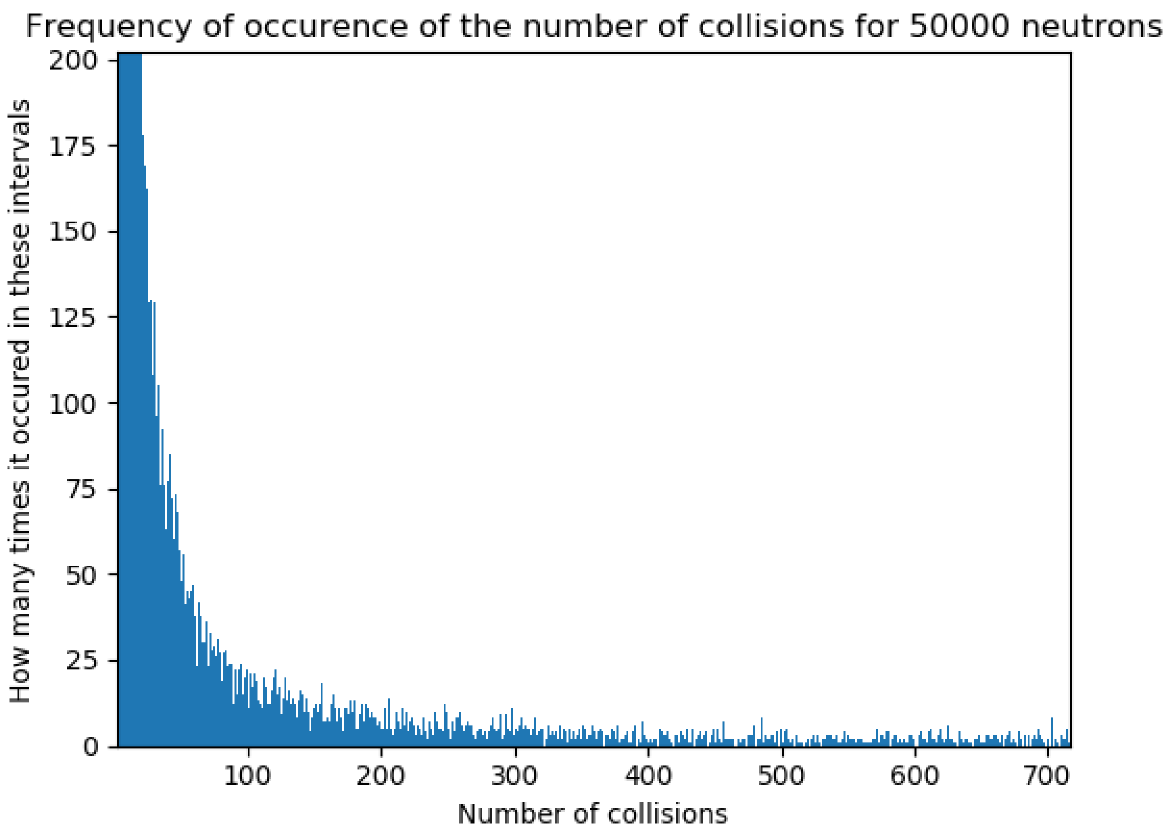


Figure 10: Number of occurence of N_V for collisions in Iron and N=50 000 neutrons

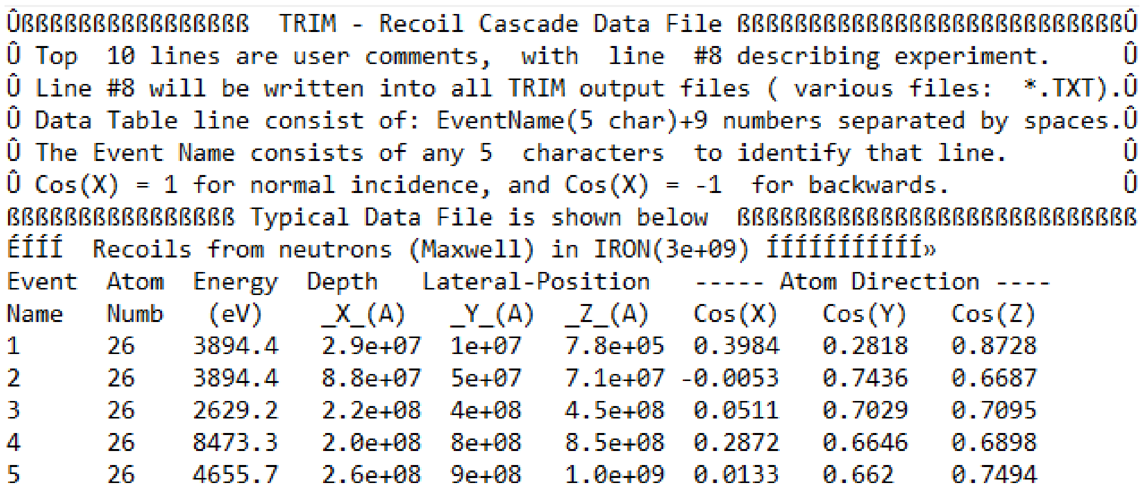
The most frequent ending mode is back scattering with 90 % of single neutron simulation. The remaining transmission and neutron lack of energy respectively happens 9 % and 1% of the cases. The back scattering is overwhelmingly the majority of the ending mode because there is an high probability for the neutron to. This mechanism is severely worsen by the random angle α, especially for α close to $\frac{\pi}{2}$ (grazing incidence) Here are some general statistics obtained after analysing the neutron code for 50 000 neutrons:

1. The average number of collisions exclusively for ending mode with back scattering is about 20 with a standard deviation of 72.
2. The average number of collisions exclusively for ending mode with transmission is about 45 with a standard deviation of 110.
3. The average number of collisions exclusively for ending mode with lack of neutron energy is about 291 with a standard deviation of 330.

We notice that the standard deviation is always very large even for a relatively large number of neutron. The fact is that even by raising the number of neutrons simulated, we won't reduce the standard deviations. This is due, for the example of back scattering, to the discrepancy between the neutrons that are directly back scattered only after a few collisions and the ones that go deeper inside the material and only then are directed back. Yet, the differentiation between the two mechanisms is not clear because N_V stays continuous. But still the values obtained for each of the ending mode is not convergent so raising even more the number of neutrons to get smaller standard deviation is pointless. We may still have interesting reasoning for the two remaining ending modes. For transmission, some neutrons may penetrate quite directly inside the material without colliding too much, but some may collides back and forth for some time before reaching X>L. The large standard deviation for the case of lack of neutron energy is even simpler and its cause also has influence on both other ending mode great standard deviation; it is the difference of energy due to the Maxwell-Boltzmann distribution that explains such a discrepancy. Indeed, in the case where the neutron does not escape by back scattering or transmission, N_V strongly depends on the initial energy of the neutron that may vary from some eV to several MeV (see figure 1). The higher the energies, the higher the number of vacancies created.

5 Requirements and modalities for the execution of TRIM

When using directly the TRIM executable without the user interface SRIM, it needs two file as input, TRIM.IN and TRIM.DAT [15]. The TRIM.IN file contains the information about the target material, the output file requested and the specification in the case of ion irradiation. The TRIM.DAT is a file that specify, for each line it contains, information about the PKA energy, coordinates and direction. This file is only useful when we want to use TRIM as a recoil cascade calculator and not as a primary cascade calculator. For instance, the neutron code is only, in the case of neutron irradiation, a primary cascade calculator that generates a list of PKA. This list of PKA generated must be written in the TRIM.DAT file by using a text editor in my code that will communicate with TRIM. An example of such a file is shown below for neutrons following a Maxwell-Boltzmann distribution in pure iron. We can see that the first iron atom that the neutron collides with (the first PKA) was located at the coordinates (290, 100, 7.8)$*10^5$A with a certain direction and energy for the PKA. And so on until all PKA generated by one and only one neutron are written in the file. The PKA direction (or Atom Direction) is the unit vector of the atom after the collision. For this example, it is only a primary cascade of five atoms that was generated, either the neutron energy was low initially or back scattering occured quickly.

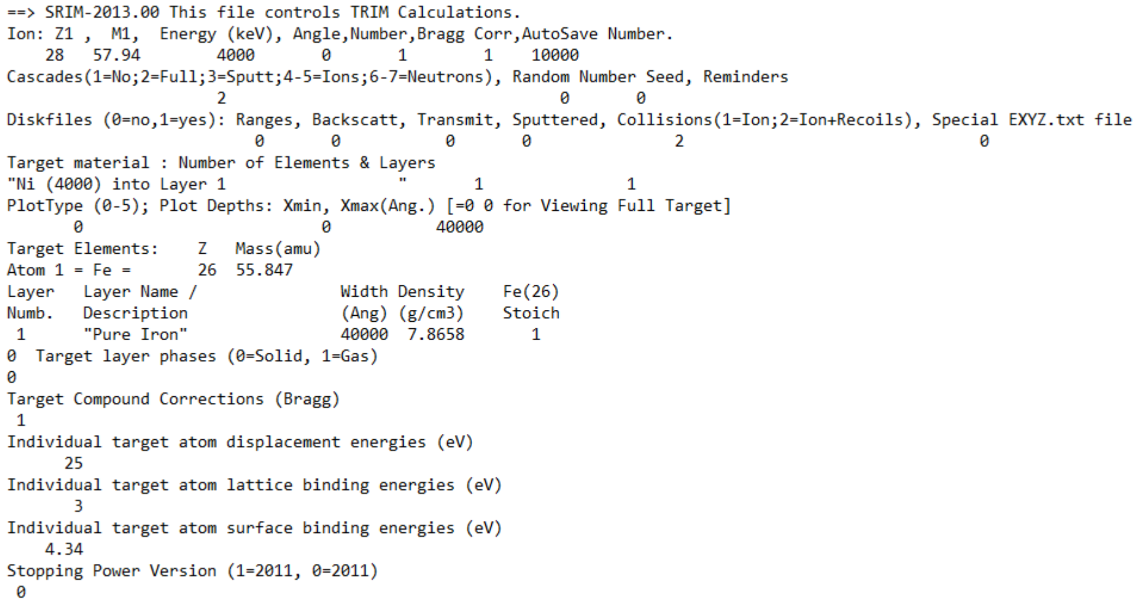


```
Ûßßßßßßßßßßßßßßß  TRIM - Recoil Cascade Data File ßßßßßßßßßßßßßßßßßßßßßßßßßßßßÛ
Û Top  10 lines are user comments,  with  line  #8 describing experiment.    Û
Û Line #8 will be written into all TRIM output files ( various files:  *.TXT).Û
Û Data Table line consist of: EventName(5 char)+9 numbers separated by spaces.Û
Û The Event Name consists of any 5  characters  to identify that line.       Û
Û Cos(X) = 1 for normal incidence, and Cos(X) = -1  for backwards.           Û
ßßßßßßßßßßßßßßß Typical Data File is shown below  ßßßßßßßßßßßßßßßßßßßßßßßßßßß
ÉÍÍÍ  Recoils from neutrons (Maxwell) in IRON(3e+09) ÍÍÍÍÍÍÍÍÍÍ»
Event  Atom  Energy  Depth    Lateral-Position    ----- Atom Direction ----
Name   Numb   (eV)    _X_(A)   _Y_(A)  _Z_(A)    Cos(X)   Cos(Y)   Cos(Z)
1      26    3894.4   2.9e+07  1e+07   7.8e+05  0.3984   0.2818   0.8728
2      26    3894.4   8.8e+07  5e+07   7.1e+07 -0.0053   0.7436   0.6687
3      26    2629.2   2.2e+08  4e+08   4.5e+08  0.0511   0.7029   0.7095
4      26    8473.3   2.0e+08  8e+08   8.5e+08  0.2872   0.6646   0.6898
5      26    4655.7   2.6e+08  9e+08   1.0e+09  0.0133   0.662    0.7494
```

Figure 11: An example of a TRIM.DAT file generated by the neutron code

Now, we will describe the TRIM.IN file that contains information about the material, the ion properties in the case of ion irradiation and the plot type and outputs we want TRIM to display.

1. The third line is only relevant for TRIM in the case of ion irradiation but it will simply be neglected for neutron irradiation. It describes the atomic number, the energy and the direction of the ion and the number of ion we want the simulate. Because I simulate one by one the damage done by either neutron or ion, this number will always be one.



```
==> SRIM-2013.00 This file controls TRIM Calculations.
Ion: Z1 ,  M1,  Energy (keV), Angle,Number,Bragg Corr,AutoSave Number.
    28   57.94       4000      0       1       1    10000
Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7=Neutrons), Random Number Seed, Reminders
                      2                                   0       0
Diskfiles (0=no,1=yes): Ranges, Backscatt, Transmit, Sputtered, Collisions(1=Ion;2=Ion+Recoils), Special EXYZ.txt file
                          0       0           0       0               2                               0
Target material : Number of Elements & Layers
"Ni (4000) into Layer 1                 "        1               1
PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.) [=0 0 for Viewing Full Target]
       0                         0             40000
Target Elements:    Z   Mass(amu)
Atom 1 = Fe =       26  55.847
Layer   Layer Name /               Width Density    Fe(26)
Numb.   Description                (Ang) (g/cm3)    Stoich
 1      "Pure Iron"                40000  7.8658      1
0  Target layer phases (0=Solid, 1=Gas)
0
Target Compound Corrections (Bragg)
 1
Individual target atom displacement energies (eV)
      25
Individual target atom lattice binding energies (eV)
       3
Individual target atom surface binding energies (eV)
    4.34
Stopping Power Version (1=2011, 0=2011)
 0
```

Figure 12: An example of a TRIM.IN file also generated by the neutron code

2. The fifth line contains the mode we want to use TRIM with concerning the cascade calculation. If we want to use the Kinchin-Pease approximation, or calculate the entire recoil cascade. If the damage is done by ions or neutrons:

    (a) 1: Ion distribution and Quick Calculation of Damade (Kinchin-Pease)

    (b) 2: Detailed Calculation with full Damage Cascade

    (c) 3: Monolayer Collision Steps / Surface Sputtering (monolayer means that the ion react with each monolayer it gets through))

    (d) 4: Ion with specific energy/ angle/ depth (quick Damage)

    (e) 5: Ion with specific energy/ angle/ depth (full cascades)

    (f) 6: Recoil cascades with neutrons (full cascades)

    (g) 7: Recoil cascades and monolayer steps (full cascades)

    (h) 8: Recoil cascades with neutrons (quick damage : Kinchin Pease)

    From 4 to the end, it uses the TRIM.DAT file. For the neutron simulation, I use the number 6 for the calculation of the full cascade and for the ion simulation, the number 2 to get comparable calculation by TRIM.

3. The seventh line contains the outputs file we want TRIM to create (1 for yes and 0 for no). I only am interested in the damage done to the material so the file that is of importance for the thesis is the Collision file with the details of the recoil cascades.

4. The eleventh line contains information about the plot we want to see, but I will not use it for the results so this line is not of real importance.

5. The rest of the lines successively describes the material properties like in the thirteenth and fifteenth line that states the list of the target elements and their respective stoeichiometry. Then are displayed useful energies like the threshold

displacement energy E_d and some lattice and surface binding energies that will be treated in details in the next section.

6 Collection and analyses of the data provided by TRIM simulations

When running TRIM in batch mode automatically time after time to get as many simulations as possible, we need to collect the useful information from different files but a major one is the COLLISION.txt file.

6.1 The content of the COLLISION.txt file

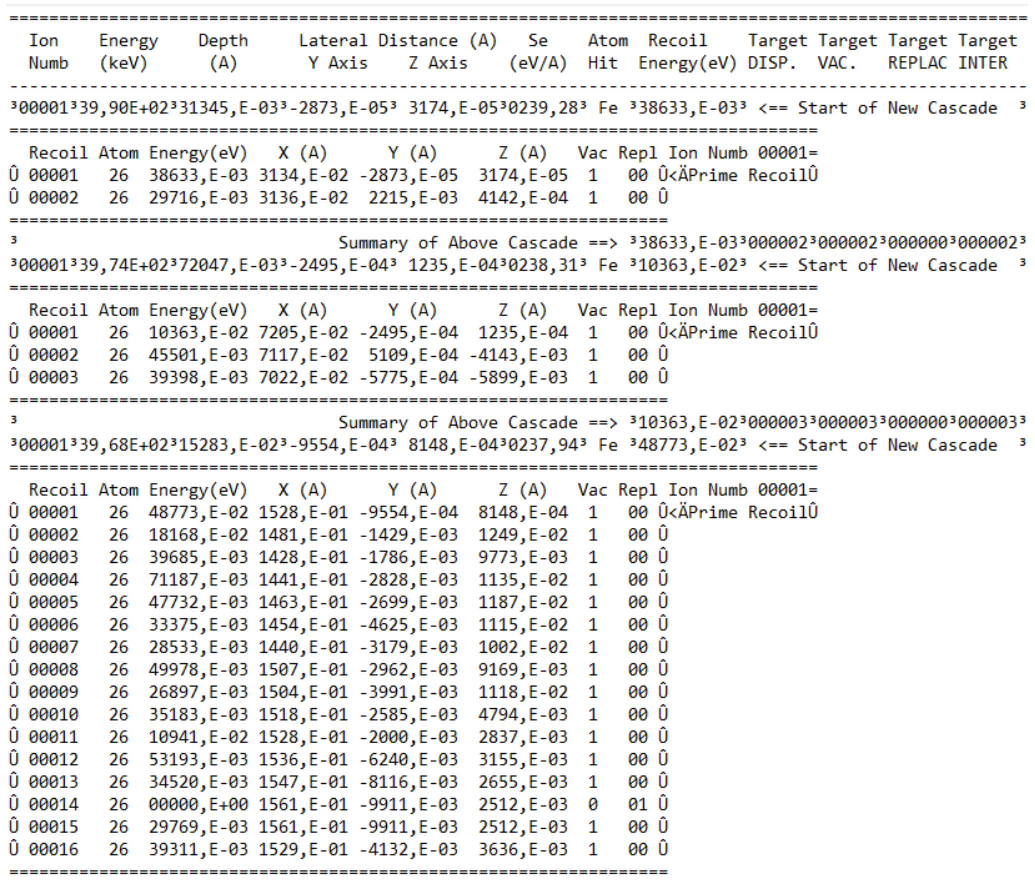
This file can easily reach hundreds of thousands of lines to analyse at the end of each simulation. We can find in it the whole set of information concerning all useful collisions of the primary and recoil cascade like the energy of the particle resulting of these collisions, the coordinates where the collisions occured, the atom of the target material that undergo each collision and what type of lattice defects it left behind (a vacancy or a replacement). In fact, even in the most detailed output file of TRIM, all collisions are not given, only the ones that caused some defect. All collisions for which the transferred energy to the target atom is not sufficient enough to permanently displace it from its lattice position is not listed in the file. A vacancy occurs when both the impinging atom and the target atom have after impact sufficient energy to leave the lattice position, thus leaving a vacancy behind them. However, we sometimes witness a replacement event which corresponds to the fact that the energy transmitted to the target atom during impact is high enough to let it escape from its lattice position but the incoming atom itself becomes trapped in this site, due to a lack of energy to escape.

More specifically, two energies are of interest, the threshold displacement energy E_d and the lattice binding energy E_b. The threshold displacement energy E_d is of the order of 25eV and is the energy necessary to permanently displace an atom from its lattice position in the bulk of the material. The lattice binding energy E_{bs} is the energy that the atom lose when leaving its atomic position. The threshold displacement energy E_d is different from the lattice binding energy E_{bs} because some energy is lost into phonons as the knocked-off atom needs to push its neighbours to get out of its position, which makes the mechanism irreversible.

6.2 Interpretation of the data contained in the output files of TRIM

I could through one of the code I created determine what the number of vacancies caused by each ion or neutron was. For a given number of neutrons simulated, I determine the total number of vacancies it caused inside the material and then, while the ion irradiation is not enough to produce the same damage to the material, I add one by one an additional ion projectile. Then the code outputs for instance that for 100 neutrons simulated, we need 33.8 Ni ions at 4MeV to get the exact same number of vacancies, explaining the counter-intuitive decimal for the ion irradiation.

However, this approach does not consider any regeneration of the material like Frenkel-Pair annihilation, recovery and defects mobility. These considerations may



```
=====================================================================================
  Ion    Energy    Depth      Lateral Distance (A)    Se    Atom  Recoil     Target Target Target Target
  Numb   (keV)      (A)        Y Axis    Z Axis    (eV/A)  Hit  Energy(eV) DISP.  VAC.   REPLAC INTER
-------------------------------------------------------------------------------------
³00001³39,90E+02³31345,E-03³-2873,E-05³ 3174,E-05³0239,28³ Fe ³38633,E-03³ <== Start of New Cascade  ³
=================================================================
  Recoil Atom Energy(eV)   X (A)      Y (A)      Z (A)   Vac Repl Ion Numb 00001=
Û 00001   26  38633,E-03 3134,E-02 -2873,E-05  3174,E-05  1   00 Û<ÄPrime RecoilÛ
Û 00002   26  29716,E-03 3136,E-02  2215,E-03  4142,E-04  1   00 Û
=======================================================
³                                 Summary of Above Cascade ==> ³38633,E-03³000002³000002³000000³000002³
³00001³39,74E+02³72047,E-03³-2495,E-04³ 1235,E-04³0238,31³ Fe ³10363,E-02³ <== Start of New Cascade  ³
=================================================================
  Recoil Atom Energy(eV)   X (A)      Y (A)      Z (A)   Vac Repl Ion Numb 00001=
Û 00001   26  10363,E-02 7205,E-02 -2495,E-04  1235,E-04  1   00 Û<ÄPrime RecoilÛ
Û 00002   26  45501,E-03 7117,E-02  5109,E-04 -4143,E-03  1   00 Û
Û 00003   26  39398,E-03 7022,E-02 -5775,E-04 -5899,E-03  1   00 Û
=======================================================
³                                 Summary of Above Cascade ==> ³10363,E-02³000003³000003³000000³000003³
³00001³39,68E+02³15283,E-02³-9554,E-04³ 8148,E-04³0237,94³ Fe ³48773,E-02³ <== Start of New Cascade  ³
=================================================================
  Recoil Atom Energy(eV)   X (A)      Y (A)      Z (A)   Vac Repl Ion Numb 00001=
Û 00001   26  48773,E-02 1528,E-01 -9554,E-04  8148,E-04  1   00 Û<ÄPrime RecoilÛ
Û 00002   26  18168,E-02 1481,E-01 -1429,E-03  1249,E-02  1   00 Û
Û 00003   26  39685,E-03 1428,E-01 -1786,E-03  9773,E-03  1   00 Û
Û 00004   26  71187,E-03 1441,E-01 -2828,E-03  1135,E-02  1   00 Û
Û 00005   26  47732,E-03 1463,E-01 -2699,E-03  1187,E-02  1   00 Û
Û 00006   26  33375,E-03 1454,E-01 -4625,E-03  1115,E-02  1   00 Û
Û 00007   26  28533,E-03 1440,E-01 -3179,E-03  1002,E-02  1   00 Û
Û 00008   26  49978,E-03 1507,E-01 -2962,E-03  9169,E-03  1   00 Û
Û 00009   26  26897,E-03 1504,E-01 -3991,E-03  1118,E-02  1   00 Û
Û 00010   26  35183,E-03 1518,E-01 -2585,E-03  4794,E-03  1   00 Û
Û 00011   26  10941,E-02 1528,E-01 -2000,E-03  2837,E-03  1   00 Û
Û 00012   26  53193,E-03 1536,E-01 -6240,E-03  3155,E-03  1   00 Û
Û 00013   26  34520,E-03 1547,E-01 -8116,E-03  2655,E-03  1   00 Û
Û 00014   26  00000,E+00 1561,E-01 -9911,E-03  2512,E-03  0   01 Û
Û 00015   26  29769,E-03 1561,E-01 -9911,E-03  2512,E-03  1   00 Û
Û 00016   26  39311,E-03 1529,E-01 -4132,E-03  3636,E-03  1   00 Û
=======================================================
```

Figure 13: An extract from the COLLISION.txt file for a Ni ion at 4 MeV

have a very significant impact on the effective damage done to the material. Even more considering that in the case of ion irradiation, defects are concentrated at the skin surface, making recombination of vacancies and interstitials much more likely than between diffuse clusters around PKA for neutron irradiation. Yet, among each cluster, recombination is still possible. The latter phenomenon, as recovery in general, occurs when the temperature is raised. In our case, the collision cascade locally create high vibrational energy (lattice vibration) that can be considered somehow similar to a local high temperature. Even if temperature is defined as a thermodynamic equilibrium quantity, we still witness a so-called "heat spike" that lasts some picoseconds before being dissipated to the surrounding lattice through energy diffusion mechanisms. This short window offers an opportunity for the mobility of defects to increase and make recombination easier.

In order to assess the difference of defects annihilation between the two type of irradiation, we will study the distribution of the projectile energy into the material. In fact, both for neutron and ion, if we consider that there is no sputtering or transmission through the material, all the energy of the projectile will be transferred somehow to the target. The way this energy is distributed between phonons, electronic excitations and energy necessary to create vacancies and interstitials may be of interest because phonons and electronic excitation are closely related to heat in the way that they can convert their respective energy of vibration or excitation into pure heat, making the local heat spike happen. The only part of energy that may not be converted into heat is the energy stored necessary to create vacancies and interstitials. More presicely, it can be relieved by annealing and the Wigner effect. The latter occurs when the material is heated making the mobility increases that let the annihilation possible between vacancies and interstitials which raises even more the temperature. It is a dangerous positive loop which makes regular recovery necessary to avoid the critical build-up of stored energy. Anyway, when a vacancy is created, the surrounding atoms that where bound to the knocked-off atom will lose their binding energy and they will need an extra energy to create a local "surface", this energy will not directly contribute to the raise of temperature.

Above is the figure 13 that contains a small portion of a COLLISION.txt file. Here is represented the information for the three first recoil cascades (the cascades caused by the three first PKA). For example, at the column energy and the fifth line, we can see that the Ni ion initially at 4 MeV lost 10 keV in electronic stopping power because it collides with the first PKA at 3.99 MeV. The transferred energy from the ion to the PKA is 38,6 eV. To discover what happened for the corresponding recoil cascades, we look to the following lines that begin with the symbol "Ù", here the eighth and ninth lines; only two collisions are caused in this cascade, including the first one between the Ni ion and the first PKA. The transferred energy from the Ni ion to the PKA is 38,6 eV but a certain amount of energy will be consumed because we need to consider the lattice binding energy E_b that the PKA lost after leaving its lattice position. After this proces, the PKA travels inside the material losing an energy E_{LostSE} through electronic stopping power (SE defined in section 3.2 justifying its use in the E_{LostSE} notation) throughout a distance L until the next collision that resulted in a SKA with a kinetic energy of 29,7-E_b eV. It will also lose energy during its path to interaction wwith atom that are not displaced and the transferred energy is dissipated as phonons. The coordinates of the collisions are given on the right of the table which makes the calculation of L possible : 2,27 Angstroms in the case depicted here. The conservation of energy states that E_1=38,6-E_b eV is divided between the energy E_{LostSE} that can be calculated by solving the following integral, $E_{SKA,r}$=29,7 eV for the SKA kinetic energy and the remaining kinetic energy $E_{PKA,r}$ of the PKA after its collision with the SKA. We solve the following equation for E_{LostSE}, L being reported by TRIM through (x,y,z):

$$L = -\int_{E1}^{E_1 - E_{LostSE}} \frac{dE}{Se(E)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{12}$$

When solving numerically this integral in order to find E_{LostSE}, I found that the energy of the PKA after being submitted to electronic stopping power is close to 37,7 eV meaning that only an energy of 0,9 eV has been lost. Nonetheless, it is not an satisfactory argument to dismiss the influence of the electronic stopping power effects on the recoils cascades because in the example taken for illustration purposes, the cascades is very small and not energetic so with an attenuated effects of electrons. Other cascades which include higher transferred energy may display a greater influence of the electronic stopping power as seen in Figure 2. Because it is the last line for this recoil cascade, this SKA will not cause any further collision that results in the creation of a vacancy. In the end we have the PKA and the SKA that travel inside the material with energies just after the collision of $E_{PKA,r}$ and $E_{SKA,r}$, respecting the following law of energy conservation:

$$E_{PKA,r} + E_{SKA,r} = E_1 - E_{LostSE} - E_b \tag{13}$$

Both PKA and SKA will undergo electronic and nuclear stopping power until they have not enough energy to continue anymore and become trapped as interstitials that required an energy of formation of E_{fi} for each interstitial. E_{fi} is defined as $3E_b$ and this approximation will be explained later on. The rest of $E_{PKA,r} + E_{SKA,r}$ is lost to phonons because at low energies, the ratio of nuclear to electronic stopping power is larger like we saw in the part dedicated to the physical framework. In order to know what the total energy given to the phonons is, we just substract to the total energy 41,6 eV the energy corresponding to electrons and stored energy (of distortion around the defects). We get that:

$$E_{PKA,r} + E_{SKA,r} = 38,6 - 0,9 - 6 - 9 = 22,7eV \tag{14}$$

The table below explains how the repartition of energy in eV for this cascade is considered.

Ionization	Phonons	Stored Energy
E_{LostSE}	$E_{PKA,r} + E_{SKA,r}$+$2E_b$	$2E_{fi}$
= 0,9 eV	= $E_{PKA,r} + E_{SKA,r}$+6 eV	= 9 eV
= 0,9 eV	= 28,7 eV	= 9 eV

Essentially, we are trying to determine the distribution of the energy losses inside the material among Ionization, phonons and binding energies:

1. Ionization or the electronic energy loss stands for the energy lost to electrons, particularly important at high velocities.

2. Phonon energy loss corresponds to the energy given from the ion, neutron and atoms to the vibration of the lattice of the target material. It arises from collision that are not causing vacancies. For instance, if the moving atom hits an atom at rest with an energy lower than the displacement energy, the atom at rest will not be able to escape its lattice position and will dissipate the transferred energy it received into phonons. An other case might be when the hitting atom has not enough energy after the collision to escape along with the hit atom that itself received enough energy, causing a replacement event. The energy that the hitting atom has left is also dissipated to the phonons [15].

3. With the term binding energy loss, I refer to the energy necessary to create every interstitials in the material. I do not consider the variation of chemical binding energies for the production of vacancies because the energy converted when an atom is struck out for its lattice position is considered to be dissipated as phonons and is thus already included in the previous section (according to the TRIM documentation). The conversion of energy can be explained as follow. When a bond between two atoms is broken, energy has to be consumed in order to keep the conservation of energy intact. The energy needed to break the bonds (the binding energies) is provided by the incoming projectiles and is considered to be dissipated as phonons. The energy considered in the distribution of energy loss as binding energy (of interstitials) is basically a stored energy of distortion around interstitial defects and will be called more precisely stored energy.

6.3 Determination of the respective contribution of ionization, phonons and stored energy for the energy losses distribution

6.3.1 In all the subcascades generated by PKA

The code will analyse one recoil cascade at a time and store the respective quantities of energy losses and then move on to the next recoil cascade until all are analysed.

For each recoil cascade, we will proceed as follow for the determination of the energy losses distribution. We consider that for a given recoil cascade C_n, with $n \geq 1$, with the PKA transferred energy from the irradiating ion or neutron being $E_n(0)$, causing a number of collision K_n. That means that we can generalize the notation for the transferred energy to the k^{th} displaced atom, with $1 \leq k \leq K_n$, as $E_n(k)$.

1. The easiest available data in the COLLISION.txt file is the number of vacancies caused by a recoil cascade. We have to sum up all the lines corresponding to the cascade under study with a "1" at the column "Vac". An other one directly accessible if we consider the sample thick enough to have no transferred atoms to the other side is the number of interstitials created. Indeed, for each vacancy created, the atom previously filling its position is now an interstitial somewhere else in the material meaning that $N_{interstitials} = N_{vac}$ under the previously stated asssumption. If we consider known the formation energy of an interstitial, we know that the energy going in the section "stored energy" will just be defined as:

   $$E_{stored} = N_{interstitials} * E_{fi} \tag{15}$$

   with E_{fi} being the formation energy of one and only one interstitial. E_{fi} is simply understood as defined in the case of self interstitials but may become much more difficult to grasp for a compound target because not all interstitial position necessarily require the same amount of energy to be occupied. Moreover, this energy also depend on the interstitial atom. But for now, the irradiation of a mono-atomic is assumed for a clear explanation of the method of resolution.

2. Now that the part of energy stored as binding energies is defined and calculated we have to evaluate how the remaining PKA energy is distributed among Ionization and Phonons energies. There are two strategies possible :

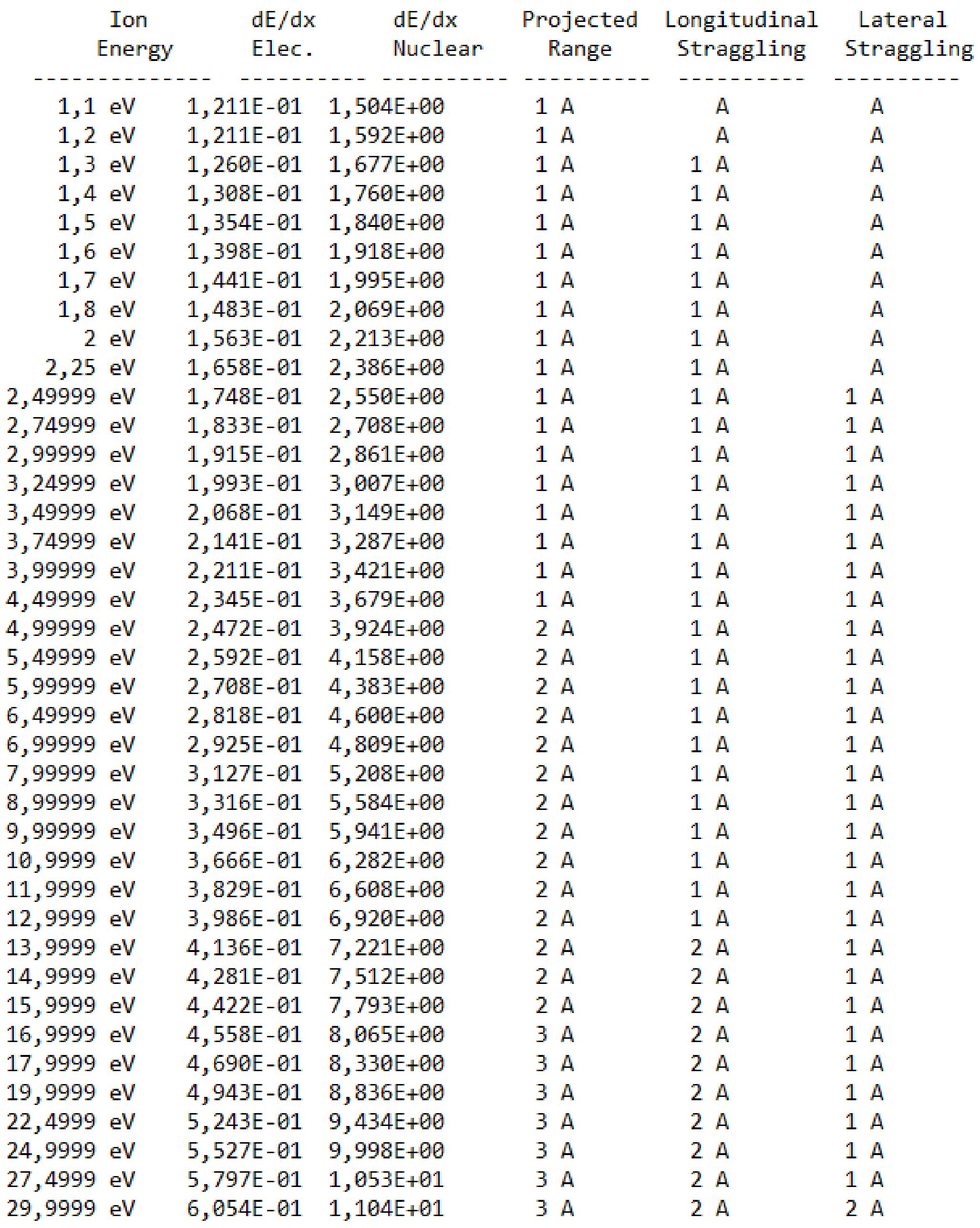
   (a) The first one is to calculate the Ionization energy first and then deduce the phonons energy by substracting the Ionization and Stored energies to the energy of the PKA:

   $$E_{Phonon} = E_n(0) - E_{Ionization} - E_{Stored} \tag{16}$$

   The problem to solve now consists on how to calculate the Ionization energies. SRIM can produce tables of electronic and nuclear stopping power for a given energy range, an example of the graph we can draw from these graphs is displayed on Figure 2. An example of the stopping table itself is displayed in Figure 14. The output file of TRIM also show the coordinates of the collisions so that we can for each displaced atom calculate the energy loss to electron by using the formula in equation 12 with the Se function being defined by the tables. The stopping power is expressed in eV lost by the ion to the electron per Angstrom travelled (for our application, it is the second column that is of interest).

(b) The second option is to first calculate the phonons energy and only then deducing the Ionization energy with a similar conservation of energy as previously. In order to do this, we have to summarize what are all the mode of creating phonons upon irradiation for each recoil cascade. This method is not a sure option. This is due to the fact that there is a confusion concerning the presence or not of minor collisions in the COLLISION.txt file. TRIM states that for the most precise COLLISION.txt files, all collisions are reported, not only the collisions resulting in a displaced atom are listed. This displacement can either be a vacancy creation or a replacement. Yet, it is very rare to encounter a collision for which the transferred energy is smaller than the displacement energy E_d. In the doubt, we will firstly consider that TRIM do not display all collisions and then we will come back to that assumption and trust the TRIM documentation. If all collisions are reported we could work with it. But a non negligible amount of energy is lost to collisions that do not result in a displacement and are thus not necessarily displayed by TRIM. We do not have access to the whole production of phonons and we must, by lack of choice, work with the first method.

We chose (a) to solve the conservation of energy. The first necessary step is to create a function that could be able to sort the collisions in cascade and sub-cascade. In fact, TRIM displays for each recoil cascade a list of all collisions causing a displacement in a row but when we switch from one cascade to an other, we should obviously not take into account the ionization between the localisation of the last collision of the i^{th} and the beginning of the $(i+1)^{th}$ sub-cascade because the corresponding path does not belong to the same cascade or any path traveled by any atom. The key variable to assess and sort the collisions between them in order to sort out the cascade and sub-cascade is the energy of the knock atom just after impact that is at the third column in the COLLISION.txt file. While this number is decreasing as we read line after line, we know that we are at the same branch of a cascade because the energy decreases collisions after collisions and along the path taken. If, up to a point, this number rises again, it means that the exploration of the current branch is over and we jump to the next one, from bottom to up. The expression "bottom to up" means that we explore all the sub branches of a branch before exploring the next branch, with this method, no branch is forgotten. But the drawback with this exploration displayed with a serie of lines is that there is no clear indication when we jump from one cascade to another and it requires a method of disentanglement of the data.



```
      Ion         dE/dx      dE/dx     Projected  Longitudinal   Lateral
     Energy       Elec.      Nuclear     Range     Straggling   Straggling
  -----------  ---------- ---------- ----------  ----------  ----------
    1,1 eV    1,211E-01  1,504E+00       1 A          A           A
    1,2 eV    1,211E-01  1,592E+00       1 A          A           A
    1,3 eV    1,260E-01  1,677E+00       1 A        1 A           A
    1,4 eV    1,308E-01  1,760E+00       1 A        1 A           A
    1,5 eV    1,354E-01  1,840E+00       1 A        1 A           A
    1,6 eV    1,398E-01  1,918E+00       1 A        1 A           A
    1,7 eV    1,441E-01  1,995E+00       1 A        1 A           A
    1,8 eV    1,483E-01  2,069E+00       1 A        1 A           A
      2 eV    1,563E-01  2,213E+00       1 A        1 A           A
   2,25 eV    1,658E-01  2,386E+00       1 A        1 A           A
2,49999 eV    1,748E-01  2,550E+00       1 A        1 A         1 A
2,74999 eV    1,833E-01  2,708E+00       1 A        1 A         1 A
2,99999 eV    1,915E-01  2,861E+00       1 A        1 A         1 A
3,24999 eV    1,993E-01  3,007E+00       1 A        1 A         1 A
3,49999 eV    2,068E-01  3,149E+00       1 A        1 A         1 A
3,74999 eV    2,141E-01  3,287E+00       1 A        1 A         1 A
3,99999 eV    2,211E-01  3,421E+00       1 A        1 A         1 A
4,49999 eV    2,345E-01  3,679E+00       1 A        1 A         1 A
4,99999 eV    2,472E-01  3,924E+00       2 A        1 A         1 A
5,49999 eV    2,592E-01  4,158E+00       2 A        1 A         1 A
5,99999 eV    2,708E-01  4,383E+00       2 A        1 A         1 A
6,49999 eV    2,818E-01  4,600E+00       2 A        1 A         1 A
6,99999 eV    2,925E-01  4,809E+00       2 A        1 A         1 A
7,99999 eV    3,127E-01  5,208E+00       2 A        1 A         1 A
8,99999 eV    3,316E-01  5,584E+00       2 A        1 A         1 A
9,99999 eV    3,496E-01  5,941E+00       2 A        1 A         1 A
10,9999 eV    3,666E-01  6,282E+00       2 A        1 A         1 A
11,9999 eV    3,829E-01  6,608E+00       2 A        1 A         1 A
12,9999 eV    3,986E-01  6,920E+00       2 A        1 A         1 A
13,9999 eV    4,136E-01  7,221E+00       2 A        2 A         1 A
14,9999 eV    4,281E-01  7,512E+00       2 A        2 A         1 A
15,9999 eV    4,422E-01  7,793E+00       2 A        2 A         1 A
16,9999 eV    4,558E-01  8,065E+00       3 A        2 A         1 A
17,9999 eV    4,690E-01  8,330E+00       3 A        2 A         1 A
19,9999 eV    4,943E-01  8,836E+00       3 A        2 A         1 A
22,4999 eV    5,243E-01  9,434E+00       3 A        2 A         1 A
24,9999 eV    5,527E-01  9,998E+00       3 A        2 A         1 A
27,4999 eV    5,797E-01  1,053E+01       3 A        2 A         1 A
29,9999 eV    6,054E-01  1,104E+01       3 A        2 A         2 A
```

Figure 14: An extract from the stopping table for Al ions inside Al_2O_3

Once all the cascades within the recoil cascade are identified, we can then calculate the Ionization energies. We need to determine within each cascade and between all paths travelled by atoms in this cascade, as in equation (12), the energy E_2 just before the $(k+1)^{th}$ collision. The energy E_1 just after the k^{th} collision is known as E_1=$E_n(k)$-E_b. But we need to beware the fact that some collisions are not reported in the output file as mentioned above. The E_1 energy is not equal to the Ionization energy loss during that path plus the $E_n(k+1)$. There is the missing term of energy lost to phonons for minor collisions (those that do not result in atomic displacement). Yet, this will not be an issue because we chose the strategy (a) over the (b) explicitly for that reason. I mentioned this in order not to be surprised that the conservation of energy does not seem to be respected in the TRIM output file.

6.3.2 In the primary cascade

In the case of the recoil cascades, the physical reasoning was the same for neutron irradiation and for ion irradiation. But for the primary cascade, it is different because we have to analyse the motion and energy losses of a neutron in one case and of an ion in the second case which will give totally different results due to their respective interaction type with the medium..

1. In the case of the neutron, it is quite simple, we do not need to consider the formation energy of an interstitial because a neutron cannot form an interstitial at the end of its travel inside the target material. it will most probably be absorbed by an atom. There will also be no Ionization loss for the neutron for it is a neutral particle. Eventually, the only remaining energy loss is the phonon and the neutron code communicates with TRIM all the collisions the neutron caused inside the material and not only the one causing a displacement. At this point, it is possible to create two versions of the neutron code: One that considers all collisions and ignores the requirements in terms of E_b and E_d for the atomic displacements and the other new one that considers only the collisions that cause a displacement (if the neutron transfers more energy to the atoms than E_d). The first one seems obsolete now and the new one has been used starting from here.

2. In the case of ion irradiation, we must consider the tiny contribution of its interstitial formation energy in the target material. We must also more importantly consider the Ionization energy loss between each collision. The remaining energy to determine is the one given to atoms and dissipated as phonons for minor collisions not reported by TRIM :

$$E_{Ion} = E_{TransferredToPKA} + E_{Ionization} + E_{MinorCollisions} \tag{17}$$

with $E_{TransferredToPKA}$ subdivision and calculation being described in the previous subsubsection 6.3.1 for the recoil cascades. The equation (12) was once again used to determine the Ionization energy loss.

6.3.3 The issue of the algorithm complexity related to the method of calculus of the energy loss to electron

For each simulation of one neutron for instance is generated the COLLISION.txt file that can easily reach hundreds of thousands of lines to be analysed. The minority of these lines are table description to make the file more readable. Yet, the vast majority of the file is just filled with collisions data, one line for each collision. We roughly solve for each of these line (see details in the section "In all the subcascades generated by PKA") the equation (12) to find the ionization energy loss. This equation is an integral to solve thanks to the electronic stopping function we got by interpolation (through tables like in Figure 14). If we add to that the reading of the file plus all the others complex functions I needed to define, the complexity of the algorithm would be too high and the time needed to finish it is not reasonable enough for the calculation power of my personal computer to make a significant amount of experiment. I chose to make an acceptable approximation to reduce the complexity of the algorithm.

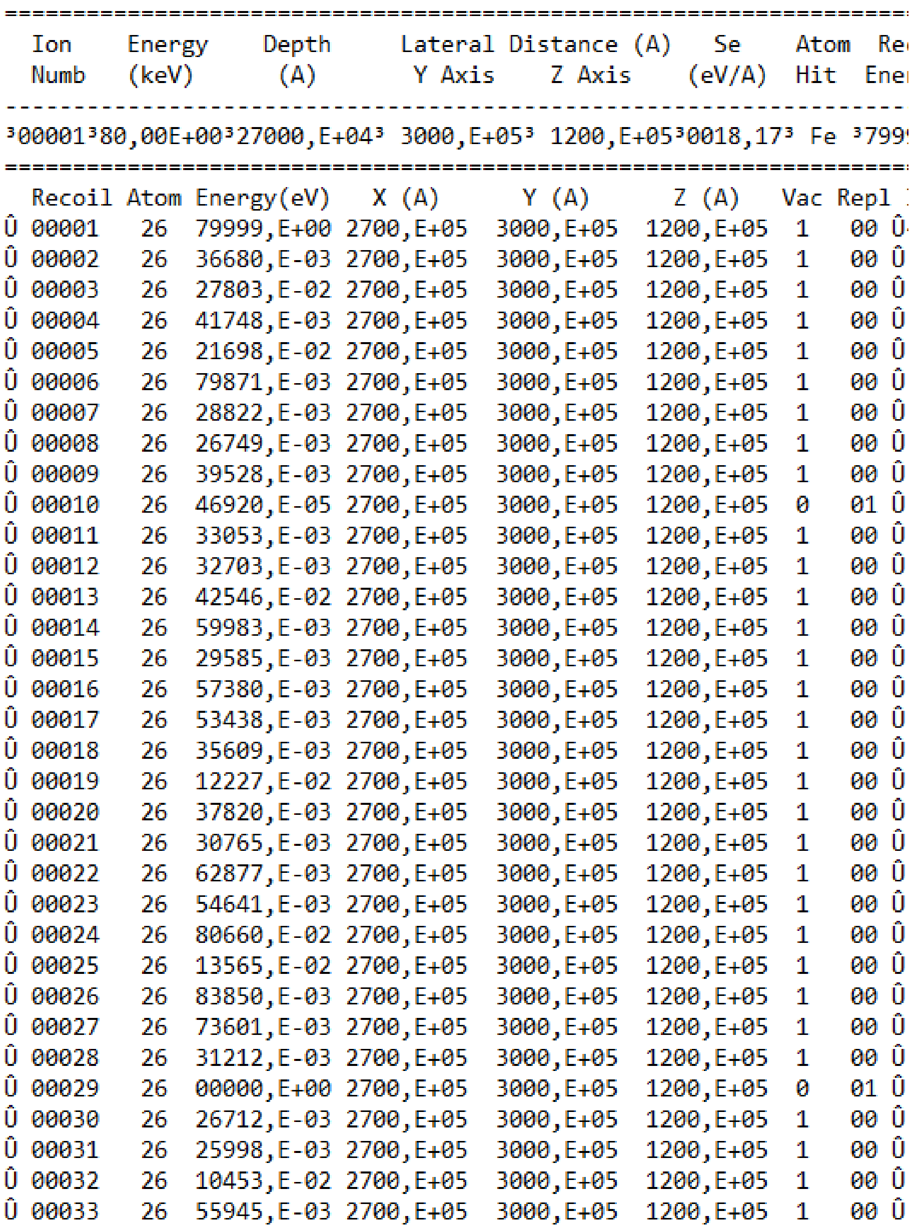
The equation (12) consists in finding the energy of the atom, initially at E_1, after the ionization losses for a traveled distance L by using a $S_e(E)$ electronic stopping function interpolated from a table generated by TRIM. This function is used inside the integral and changes with the energy, meaning that when the energy of the atom decreases when losing energy to the electrons, the $S_e(E)$ function has a new value to be calculated and so on. Instead, we could consider $S_e(E)$ to be constant and equal to $S_e(E_1)$ for the whole travel of the atom up to the next collision. It would overestimate the ionization energy loss because the energy loss to electrons diminishes with decreasing atomic kinetic energy. This method is nonetheless poorly accurate because the electronic stopping function is sensitive to the variation of energy especially at low energy. We may be using a step function to find a middle ground between the extremely demanding calculation of the precise value and the oversimplified approach consisting in using the same value of $S_e(E)$ for all the distance traveled between two collisions.

The method with the step function would work in the following way. By knowing the starting energy just after a collision E_1 and the distance L to be traveled until the next collision, we need to find the energy loss to electrons during this path that will be segmented in a certain number with each one having an attributed constant $S_e(E)$. If we divide the path in n equal distances, each one being $\frac{L}{n}$ long, we would only need to calculate n times the $S_e(E)$ corresponding. We need to determine each energy E_i after a travel of length $\frac{L}{n}$ to know the $S_e(E)$ to attribute. The set of equations labelled by i we need to solve for E_{i+1} is:

$$\forall i \in [\![1; n-1]\!], \quad \frac{L}{n} = \frac{E_i - E_{i+1}}{Se(E_i)} \tag{18}$$

Obviously, if $n \to \infty$, we would come back to the integral formulation in equation (12). I just performed a discretization of an integral. So each E_{i+1} is determine by calculating the right side of the following equation :

$$\forall i \in [\![1; n-1]\!], \quad E_{i+1} = E_i - Se(E_i)\frac{L}{n} \tag{19}$$



```
=================================================================
  Ion    Energy    Depth     Lateral Distance (A)   Se     Atom  Re
  Numb   (keV)      (A)        Y Axis     Z Axis   (eV/A)  Hit   Ene
-----------------------------------------------------------------
³00001³80,00E+00³27000,E+04³ 3000,E+05³ 1200,E+05³0018,17³ Fe ³799
=================================================================
  Recoil Atom Energy(eV)   X (A)      Y (A)      Z (A)    Vac Repl
Û 00001   26  79999,E+00 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00002   26  36680,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00003   26  27803,E-02 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00004   26  41748,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00005   26  21698,E-02 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00006   26  79871,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00007   26  28822,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00008   26  26749,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00009   26  39528,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00010   26  46920,E-05 2700,E+05  3000,E+05  1200,E+05  0    01 Û
Û 00011   26  33053,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00012   26  32703,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00013   26  42546,E-02 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00014   26  59983,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00015   26  29585,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00016   26  57380,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00017   26  53438,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00018   26  35609,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00019   26  12227,E-02 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00020   26  37820,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00021   26  30765,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00022   26  62877,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00023   26  54641,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00024   26  80660,E-02 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00025   26  13565,E-02 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00026   26  83850,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00027   26  73601,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00028   26  31212,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00029   26  00000,E+00 2700,E+05  3000,E+05  1200,E+05  0    01 Û
Û 00030   26  26712,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00031   26  25998,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00032   26  10453,E-02 2700,E+05  3000,E+05  1200,E+05  1    00 Û
Û 00033   26  55945,E-03 2700,E+05  3000,E+05  1200,E+05  1    00 Û
```

Figure 15: An extract from the COLLISION.txt file for neutrons

There is an other major issue with the TRIM output that I haven't mentioned yet, more precisely in the case of neutron irradiation. Because the neutron traveling inside the target material have an higher penetration depth than the ions, the coordinates of the collisions will be extended in both three dimensions. In the figure 15, we see that the coordinates are the same for all collisions, here at a depth of 2.7 cm, so the small relative variations of coordinates for the atom cascade (in the nanometer range) is not able to be displayed with only four digits. The outputs of TRIM can't be modified so the use of the neutron code in conjunction with TRIM is highly compromised.

One way to overcome this issue would be to transfer all PKA generation to the origin (the point where the irradiating particle enters the material) so that all cascades induced by the neutron are still within the range of display of TRIM. Yet, choosing the origin point is not the best idea because some PKA are generated with a velocity pointing towards negative X, and they would consequently leave the material directly after their generation and their path would not be followed. If TRIM can only display four digits, the maximum depth for which we still have the collisions locations with the precision of an Angstrom is 9999 Angstroms. We could choose the middle point at a depth of 5000 Angstroms for the PKA to be generated. With this choice, we are expecting that atoms have enough room on both side to travel and that the data will be correctly collected with no omission.

6.3.4 A first attempt to sort out theoretically the cascade branching system

In order to understand how to sort the cascade out, let us consider a typical recoil cascade C_n by considering the same notations defined in section "In all the subcascades generated by PKA". Starting from the list of all energies $E_n(k)$ after each collision k, we have to determine what path was taken by what atom in order to calculate the ionization energies. The TRIM documentation is not precise enough to explain to what rules the displayed order of recoils obey. The only information given is that "The first line in this box shows the primary recoil and then the succeeding recoils. If some of these secondary recoils cause further recoils, the box gets complicated. But each recoil is followed until it reaches zero energy, and then the last recoil it caused is tracked, etc. until the first recoil caused by the prime recoil is tracked", taken from the chapter 9 of the SRIM documentation book [15].

If all recoils are indeed followed until they reach a zero energy, it would mean that while the successive line are decreasing in energy (the energy displayed is the transferred energy), we are in the same subcascade. In the example above, the ion transferred to the PKA an energy of 80 keV and then the following lines only concern the effect of the PKA to which 80 keV has been transferred. The PKA leave its lattice position at 80 keV - E_b and then it will travel a certain distance until the first major collision it will cause. It will lose energy to the electrons and to the phonons (due to minor collisions) but these energy loses are not explicitly described in the COLLISION.txt file. The first major collision caused by the PKA is at line 2 for which an energy transfer of 36.7 eV was done to an iron atom. This atom could cause further recoil but they will not be written directly after because the PKA is not at zero energy and can still cause further collisions. The second collision caused by the PKA is at line 3 with a transferred energy of 278 eV. We can continue this reasoning until the PKA has no energy anymore because all its energy has been lost to transferred energy (to minor and major collisions) and ionization. The first question to answer is to know what is the exact last line that describes the last collision caused by the PKA itself. Then, when this line is determined, according to the TRIM documentation, the next line describes the last major collision caused by the last recoil atom hit by the PKA. Let add some formalism to clearly understand the process of display.

We need to give a name to each major collision in order to be able to make the reasoning clear. For instance, the first cascade is the cascade displayed is the cascade caused by the PKA. We can call each of the collision of this cascade $K(n_1)$ with n_1 that indicate the index of the collision (the relative position in the concerned cascade). That means that is the PKA cascade contains N_1 collisions, we will call each collision as $K(n_i)$ with $n_i \in [\![1, N]\!]$. Then, the second generation of cascade will be described with $K(n_1, n_2)$ with n_1 that indicates from which collision, in the PKA cascade, the atom that caused the $K(n_1, n_2)$ collision come from. The n_2 index has a similar function has the previous case for the PKA cascade and indicates what is the index of the collision $K(n_1, n_2)$ inside the current cascade (which is indicated with the first index n_1). The latter current cascade contains N_2 collisions so $n_2 \in [\![1, N_2]\!]$. Now, let generalize to the n^{th} generation of subcascade.

Let consider a collision described with the notation $K(n_1, n_2, n_3, ..., n_{m-1}, n_m)$. In order to know where this collision took place in the tree (or graph) of collisions, we need to read the indexes from left to right. The indexes will indicate what is the succession of events that caused the current collision. If there is m indexes, we know that this collision belong to a m^{th} generation of cascade. The $\{n_i\}_{i \in [\![1,m]\!]}$ indexes indicate the relative position in each subcascade branch the intermediate collisions that lead to the $K(n_1, n_2, n_3, ..., n_{m-1}, n_m)$ collision. Each intermediate subcascade contains respectively $\{N_i\}_{i \in [\![1,m]\!]}$ collisions which means that :

$$\forall i \in [\![1, m]\!], \quad 1 \leq n_i \leq N_i \tag{20}$$

The first branch that lead to the $K(n_1, n_2, n_3, ..., n_{m-1}, n_m)$ collision is the one that was created by the $K(n_1)$ collision in the PKA cascade. Then the relative position of the collision in the latter cascade that causes a further deviation is a sub branch is indicated as n_2. This second collision in the branching system is $K(n_1, n_2)$. And the reasoning is the same until we get to the m^{th} generation of cascade where the current $K(n_1, n_2, n_3, ..., n_{m-1}, n_m)$ collision is studied. Eventually, when confronted with any collision $K(n_1, n_2, n_3, ..., n_{m-1}, n_m)$, the path of the m successive collisions that lead to it are $K(n_1)$, $K(n_1, n_2)$, $K(n_1, n_2, n_3)$, ..., $K(n_1, n_2, n_3, ..., n_{m-1})$, $K(n_1, n_2, n_3, ..., n_{m-1}, n_m)$ or more compactly $\{K(n_1, n_2, n_3, ..., n_{j-1}, n_j)\}_{j \in [\![1,m]\!]}$. Another important point has to be mentioned, the N_i indexes are dependent on all the previous indexes $\{N_j\}_{j \in [\![1,i-1]\!]}$. Each cascade may potentially create several sub cascades with different length (in term of number of collisions belonging to the subcascade). Dependent has to be understood in the following way : an index n_i may cause through its own cascade several n_{i+1} cascades with different length (different N_{i+1}). This is why the collision notation $K(n_1, n_2, n_3, ..., n_{m-1}, n_m)$ has to be read from left to right and not the opposite. Any given n_i index cannot be evaluated on its own but in accordance with all the previous ones. To account for this fact we should write for any N_i that it depends on all the previous n_j with $j \leq i - 1$ by writing $N_i(n_1, n_2, ..., n_i)$ but it would make the notation even more complex.

The cascades themselves could be name accordingly. Previously each PKA cascade where named C_n because we needed to differentiate them between the whole primary cascade but now, we are only interested in the cascade caused by one PKA, So calling the PKA cascade C_{PKA} seems clear enough. C_{PKA} contains only the collisions that were strictly caused by the PKA itself so the set of $K(n_1)$ with $n_1 \in [\![1, N_1]\!]$. This cascade C_{PKA} is the only first generation cascade. The second generation cascades are called $C(n_1)$ with $n_1 \in [\![1, N_1]\!]$ and contains all $K(n_1, n_2)$ collisions for each $n_1 \in [\![1, N_1]\!]$ and each $n_2 \in [\![1, N_2]\!]$, with N_2 that depends on the n_1 considered like it was explained in the previous paragraph. More generally, let call $C(n_1, n_2, ..., n_i)$ the $(i + 1)^{th}$ generation cascade that comes from the following branching system (with the nodes indicated, each node being a collision) :

1. The n_1^{th} collision of the C_{PKA} cascade ($K(n_1)$),
2. The n_2^{th} collision of the $C(n_1)$ cascade ($K(n_1, n_2)$),
3. The n_3^{th} collision of the $C(n_1, n_2)$ cascade ($K(n_1, n_2, n_3)$),
4. ...
5. The n_{i-1}^{th} collision of the $C(n_1, n_2, .., n_{i-2})$ cascade ($K(n_1, n_2, ..., n_{i-1})$),
6. The n_i^{th} collision of the $C(n_1, n_2, .., n_{i-1})$ cascade ($K(n_1, n_2, ..., n_i)$),

We also stresses out that a collision do not necessarily eject an atom that will cause another collision. We thus define for each subcascade an index $n_{i,max} \in [\![1, N_i]\!]$ that represents the last recoil of the current sub cascade that will cause further collision (i.e. that a new branch starts from this collision). The transferred energy to the knocked-off atom of this current collision is not necessarily enough for it to cause a major collision afterwards. Only a fraction of each subcascade collisions will continue the branching and extend the tree by causing a new generation of sub cascade. We need to take this phenomenon into account by introducing a new set of notation.

For a given cascade $C(n_1, n_2, ..., n_i)$, the collisions $\{K(n_1, n_2, n_3, ..., n_{j-1}, n_j)\}_{j \in [\![1, i+1]\!]}$ are the ones it contained and since there is no ambiguity for the i^{th} first indexes, only the last index n_{i+1} is of true interest. We know that $1 \leq n_{i+1} \leq N_{i+1}$ but only a sub set of $[\![1, N_{i+1}]\!]$ indexes correspond to collisions that will cause further collisions. Let create a list $L(C(n_1, n_2, ..., n_i)) \subset [\![1, N_{i+1}]\!]$ that contains only the elements of $[\![1, N_{i+1}]\!]$ that do create a new generation of sub cascade. Let call the elements all this list l_j with $j \in [\![1, j_{max}]\!]$. The subscript j describes the index position, among the collisions that caused a next generation of cascade. The subscript j_{max} could be equal to only 0 since a cascade do not necessarily give rise to a next generation cascade through its collisions. If, on the other extreme, all collisions in the $C(n_1, n_2, ..., n_i)$ cascade do create next generation cascade, then $l_{jmax} = N_{i+1}$.

Now that the collisions and cascades are explicitly defined, we can continue the analysis. We consider that the maximum generation of the whole following cascade caused by a PKA is $G_{max} \in \mathbb{N}$. It means that the longest path from vertex to vertex in the tree of collisions is of length G_{max} (contains $G_{max}+1$ vertices and G_{max} edges, with one vertex by sub cascade). When we speak of vertices we imagine straight lines but in this case, it is not necessarily the case in the (x,y,z) space. In fact, a cascade has not all its collision location aligned. It is called a vertex because it connects two nodes of interest but has no dimensional signification. Because the cascade representation is similar to a tree, the vocabulary used is taken from it. All paths that start from the trunk and finishes at the tip of any branch do not have the same length, each path being different. In order to know what the generation G of the last cascade of this current path is, we have to count the number of edges the path is composed of. Considering that the vague affirmation of the TRIM documentation stated above is correctly understood, here is the process of display used by the COLLISION.txt file with the notations defined previously, line after line:

1. The whole PKA cascade C_{PKA} : all $K(n_1)$ with $n_1 \in [\![1, N_1]\!]$.

2. Because TRIM continues with the last recoil, it chooses the last element of the list L(C_{PKA}), $l_{jmax}(C_{PKA})$, and considers the cascade this last collision K($l_{jmax}(C_{PKA})$) caused. This cascade is C($l_{jmax}(C_{PKA})$) and is entirely displayed right after the C_{PKA} cascade.

3. Once the recoil atom ejected from its lattice position at the collision K($l_{jmax}(C_{PKA})$) has no energy anymore, it will stop the cascade and the last atom it ejected and that caused a third generation cascade is studied. This cascade is C($l_{jmax}(C_{PKA}), l_{jmax}(C(l_{jmax}(C_{PKA})))$). This notation is complete but we are only at the third generation and it is already unreadable. Since we read the position from left to right, there is no need anymore to indicate what is previous cascade that the l_j is taken from. The cascade could be rename unambiguously C(l_{jmax}, l_{jmax}) and is displayed after the first two steps. The first C_{PKA} in parenthesis was removed because the atom that caused the cascade $C(l_{jmax})$ necessarily come from the PKA cascade. Now that this cascade is known the next generation necessarily comes from it so the $C(l_{jmax}(C_{PKA}))$ in the second term can also be removed.

4. The pattern repeats and all last recoils are followed until they reach zero energy like the TRIM documentation says. The next cascade to be displayed is C($l_{jmax}, l_{jmax}, l_{jmax}$) with the same reasoning as in step 3. One the first path of the tree is explored up to the end, with the last cascade being $C(l_{jmax}, l_{jmax}, ..., l_{jmax})$, with l_{jmax} written G times. All the cascades $C(l_{jmax}, l_{jmax}, ..., l_{jmax})$, with l_{jmax} written p times ($p \in [\![0, G]\!]$), are displayed in between.

5. The last recoil cascade to be analysed is the $C(l_{jmax}, l_{jmax}, ..., l_{jmax-1})$ cascade. If the last l_{jmax} of this cascade caused a next generation cascade (with G>G($C(l_{jmax}, l_{jmax}, ..., l_{jmax})$), we would need to go deep down into that cascade and all the others above (in the tree metaphor) with the same strategy as the one described now. Once it is done, we study the $C(l_{jmax}, l_{jmax}, ..., l_{jmax-2})$ cascade and so on by decreasing the last index until $j_{max} - k$ reaches 0, with $k \in [\![0, j_{max}]\!]$.

In the figure 16 is an example of a tree of collision. Each vertex correspond to a collision. The numbers written on each vertices correspond to its number of apparition in the list of lines of collisions. The 1 is the collision between the ion (or neutron) with a PKA. The knocked-off PKA follows the cascade that runs from 1 to 7 and it is the first generation cascade C_{PKA}. Then, because we start from the last recoil, we go down the tree from the last branch of the C_{PKA} and we notice that a second generation cascade $C(6)$ is created from the vertex 6 and has only one vertex in it (number 8). Then the cascade C(5) and C(4) are written. The cascade C(2) is followed and is made of the vertices from 15 to 18. In this tree, there is only one third generation cascade and it is the C(2,2). The second index 2 refers to the relative position inside the cascade C(2) from where the third generation cascade starts (vertex 16 which is the second collision of the cascade C(2)). In reality, the trees can contains hundreds of vertices.

The whole point is to find an algorithm able to calculate the ionization energy by having as input the list of vertices with their corresponding transferred energy

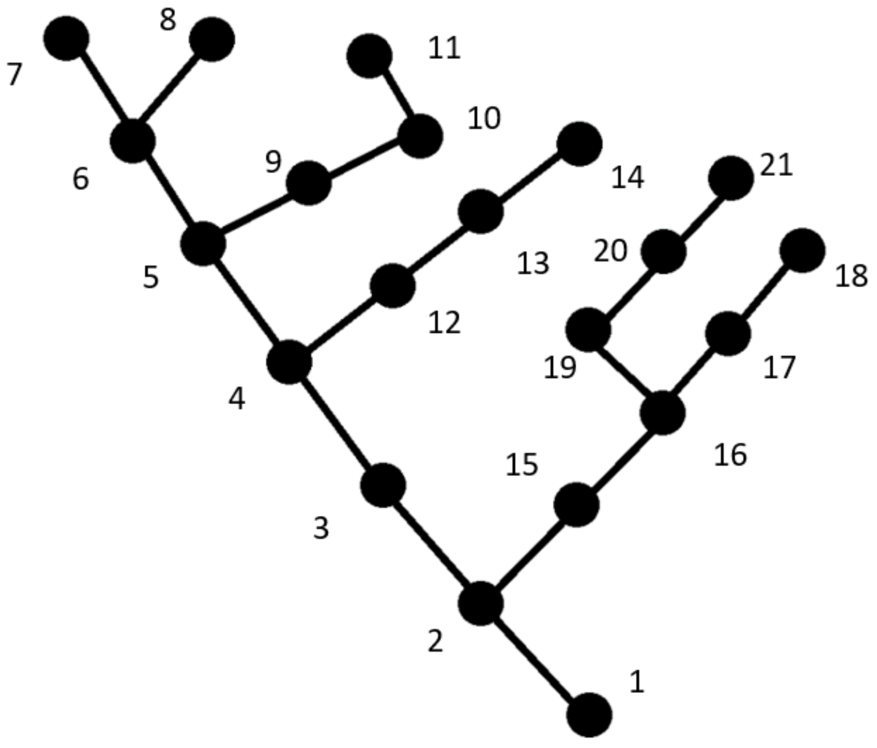


Figure 16: An example of a tree of collision

and localisation. We have to connect the vertices in the correct way for a correct calculation of the ionization energy. In our example, it would make no sense to calculate the ionization energy from 8 to 9 or from 11 to 12 even if they are displayed in a row in the list, because no atom traveled the space between the 8^{th} and 9^{th} collisions. The organization of the list could be done in the following way.

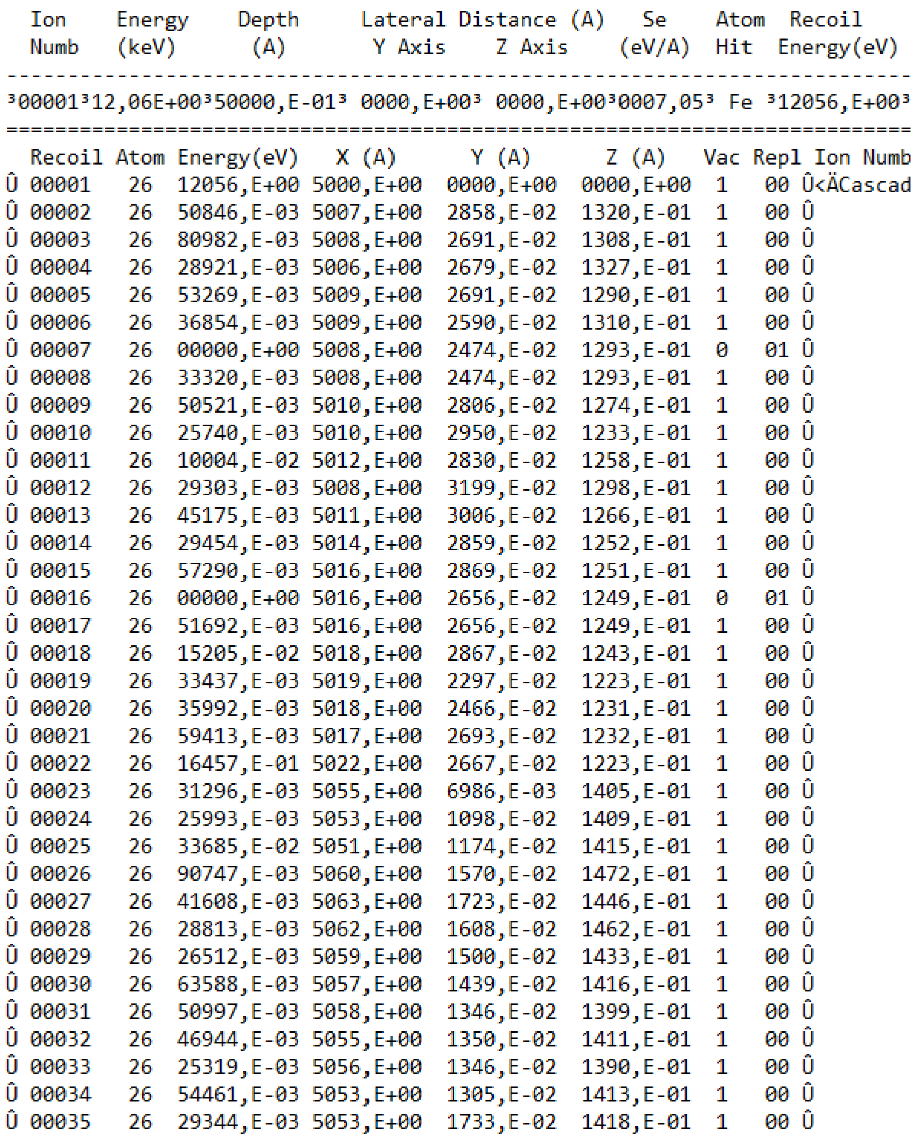
Since we know the transferred energy to the PKA, we know that the first cascade will stop when the PKA energy is exhausted. The PKA losses energy to phonons (due to minor collisions) and to electrons (through ionization) on the edges joining the different vertices of the C_{PKA}. TRIM do not display any information in order to access the energy lost to phonons between the major collisions. The PKA also lost energy to the knocked-off atom during major collisions. Theoretically, when the total energy lost exceeds the transferred energy to the PKA, we know that the last line we went through is not the last collision of the C_{PKA} but the first collision of the last second generation cascade caused by the C_{PKA} (index 8 for the example above). The losses to the transferred energy is directly accessible by reading the file. The ionization energy is calculable with a certain precision. The issue is the estimation of the losses to minor collisions between major collisions. An approximation could be to use the $\frac{dE}{dx}_{nuclear}$ given alongside the $\frac{dE}{dx}_{ionization}$ that I already used for the estimation of the ionization. The document provides the value of energy loss per unit length for both major and minor collisions combined but I sought only the one for the minor collisions, so it may cause an overestimation to the losses to the phonons in the end. At the expense of the computation time, it would be possible to subtract the transferred energy during major collisions from the nuclear stopping power provided by TRIM. It would counterbalance, in an approximated way, the overestimation of the nuclear stopping power when the energy lost is calculated with the use of $\frac{dE}{dx}_{nuclear}$ in addition with the transferred energy during major collisions displayed by TRIM.

6.3.5 Further attempts to sort out theoretically the cascade branching system

By comparing the previous interpretation of the TRIM documentation stated above with the result, I realized that it was not possible. Let us take the example of the cascade displayed in the following figure to explain why. The transmitted energy to the PKA is 12.056 keV. Then in order for the PKA to jump from its initial position to the position of the second collision at the second line (in our previous model), it would lost 1.567 keV to electrons and 12.05 keV to minor collisions on the path. It is impossible since it would lose more energy that it has. If we ignore the energy loss to minor collisions but take into consideration the transferred energy to major collisions, the C_{PKA} cascade would be displayed up to the 90^{th} line with a total energy loss to electrons of 4242 eV and to the major collisions of 7837 eV. The real answer is in between because in the first case, either the energy loss to minor collisions is overestimated or TRIM displays differently from what I understood. In the second case, the existing losses to minor collisions are simply neglected, which is not a reasonable assumption to make, especially when considering the high nuclear to electronic stopping power at relatively low energies.

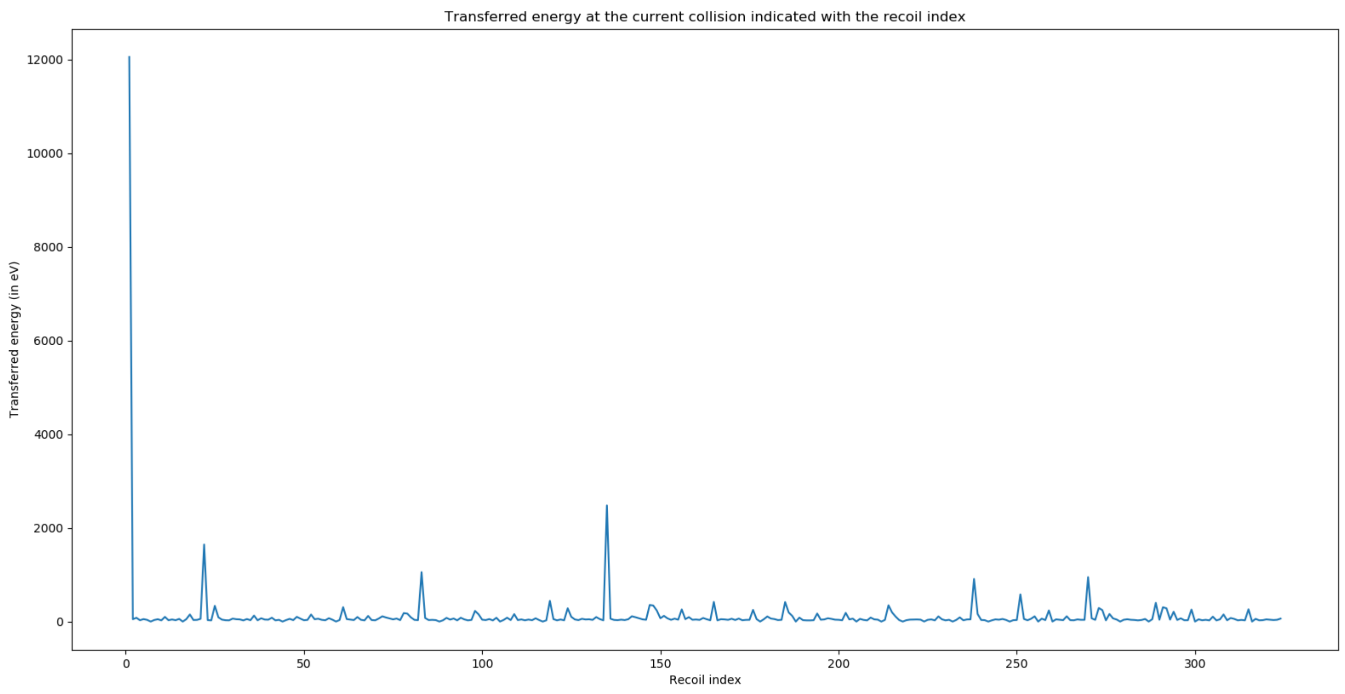
In figure 18 is shown the energy transferred for the same cascade partially displayed in figure 17. In abscissa, is the recoil index number given in the column "Recoil". When the transferred energy is equal to zero, it means that there is a replacement collision, it is only a convention of TRIM. We saw previously that, even if we completely dismiss the minor collisions participation in the loss of kinetic energy of the PKA, the cascade C_{PKA} would stop at the recoil index 90. We see a peak in figure 18 at the recoil index 135 that corresponds to a transferred energy at a collision of 2500 eV that, if we consider the previously explained branching system notation, would have been made by an other atom different from the PKA itself. Yet, the PKA has not, even once, transferred more than about 1600 eV before to an atom. This means that this atom whose transferred energy was 1600 eV could not has transferred 2500 eV to another atom. The collision at the recoil index 135 was necessarily done by the PKA itself and question the initial interpretation display mode of TRIM.

It would seem that TRIM goes through the PKA cascade C_{PKA} and stops when a recoil atom cause a subcascade and then analyses it before going on to the rest of the C_{PKA}. Yet, the method of indexes notation inside the secondary cascade (cascade that are not the C_{PKA}) is the same as in figure 16 for some reasons. In both cases, the second collision written is still the first collision the PKA caused. Yet, like we saw in the previous subsubsection, if both minor collisions and ionization are taken into account, the PKA losses more energy than it has. So an idea to verify if the second element of the list really is the first collision the PKA caused is to compare the unit vector direction of the PKA after its collision with the ion (or neutron) with the normalized vector that goes from the initial position of the PKA (before being knocked-off) and the second line position. The unit vector of the PKA velocity is $\vec{u}_0^{PKA}$=(0.2933 0.4951 0.8178), taken from the TRIM.DAT file and the

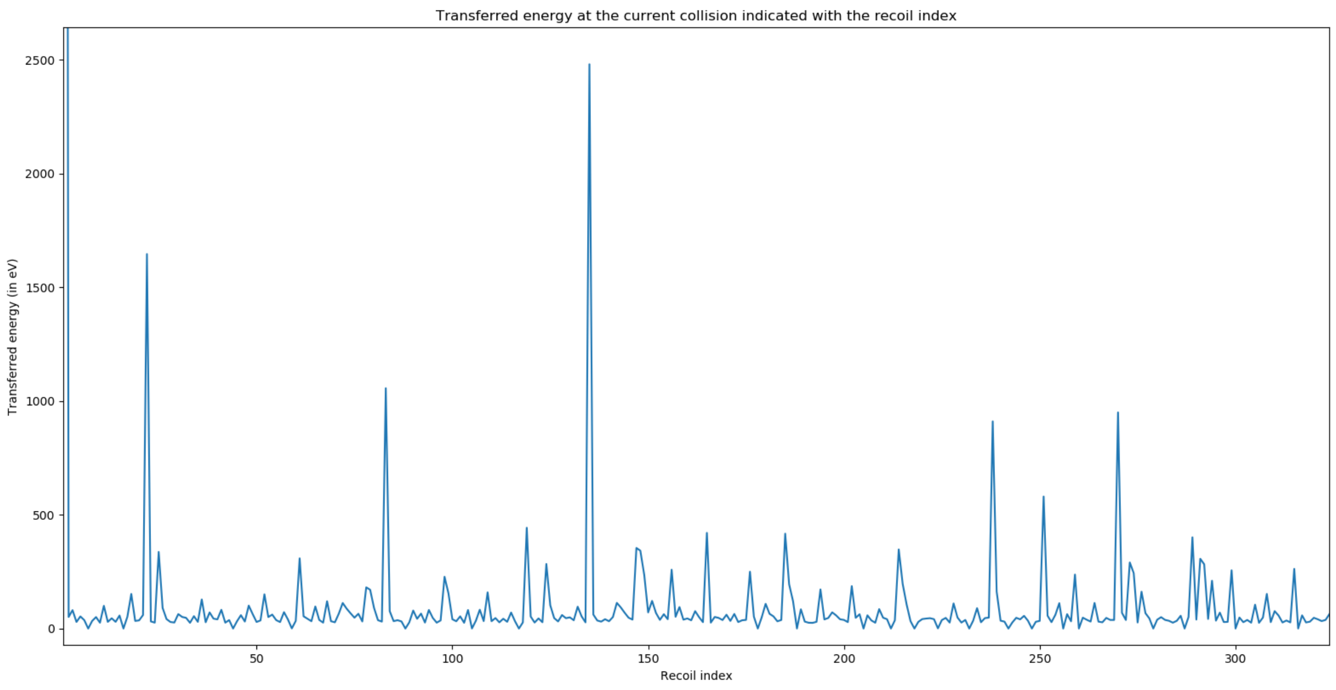


```
  Ion     Energy     Depth      Lateral Distance (A)   Se      Atom  Recoil
  Numb    (keV)       (A)        Y Axis     Z Axis    (eV/A)   Hit  Energy(eV)
---------------------------------------------------------------------------
³00001³12,06E+00³50000,E-01³ 0000,E+00³ 0000,E+00³0007,05³ Fe ³12056,E+00³
===========================================================================
  Recoil Atom Energy(eV)   X (A)      Y (A)      Z (A)    Vac Repl Ion Numb
Û 00001   26   12056,E+00 5000,E+00  0000,E+00  0000,E+00  1    00 Û<ÄCascad
Û 00002   26   50846,E-03 5007,E+00  2858,E-02  1320,E-01  1    00 Û
Û 00003   26   80982,E-03 5008,E+00  2691,E-02  1308,E-01  1    00 Û
Û 00004   26   28921,E-03 5006,E+00  2679,E-02  1327,E-01  1    00 Û
Û 00005   26   53269,E-03 5009,E+00  2691,E-02  1290,E-01  1    00 Û
Û 00006   26   36854,E-03 5009,E+00  2590,E-02  1310,E-01  1    00 Û
Û 00007   26   00000,E+00 5008,E+00  2474,E-02  1293,E-01  0    01 Û
Û 00008   26   33320,E-03 5008,E+00  2474,E-02  1293,E-01  1    00 Û
Û 00009   26   50521,E-03 5010,E+00  2806,E-02  1274,E-01  1    00 Û
Û 00010   26   25740,E-03 5010,E+00  2950,E-02  1233,E-01  1    00 Û
Û 00011   26   10004,E-02 5012,E+00  2830,E-02  1258,E-01  1    00 Û
Û 00012   26   29303,E-03 5008,E+00  3199,E-02  1298,E-01  1    00 Û
Û 00013   26   45175,E-03 5011,E+00  3006,E-02  1266,E-01  1    00 Û
Û 00014   26   29454,E-03 5014,E+00  2859,E-02  1252,E-01  1    00 Û
Û 00015   26   57290,E-03 5016,E+00  2869,E-02  1251,E-01  1    00 Û
Û 00016   26   00000,E+00 5016,E+00  2656,E-02  1249,E-01  0    01 Û
Û 00017   26   51692,E-03 5016,E+00  2656,E-02  1249,E-01  1    00 Û
Û 00018   26   15205,E-02 5018,E+00  2867,E-02  1243,E-01  1    00 Û
Û 00019   26   33437,E-03 5019,E+00  2297,E-02  1223,E-01  1    00 Û
Û 00020   26   35992,E-03 5018,E+00  2466,E-02  1231,E-01  1    00 Û
Û 00021   26   59413,E-03 5017,E+00  2693,E-02  1232,E-01  1    00 Û
Û 00022   26   16457,E-01 5022,E+00  2667,E-02  1223,E-01  1    00 Û
Û 00023   26   31296,E-03 5055,E+00  6986,E-03  1405,E-01  1    00 Û
Û 00024   26   25993,E-03 5053,E+00  1098,E-02  1409,E-01  1    00 Û
Û 00025   26   33685,E-02 5051,E+00  1174,E-02  1415,E-01  1    00 Û
Û 00026   26   90747,E-03 5060,E+00  1570,E-02  1472,E-01  1    00 Û
Û 00027   26   41608,E-03 5063,E+00  1723,E-02  1446,E-01  1    00 Û
Û 00028   26   28813,E-03 5062,E+00  1608,E-02  1462,E-01  1    00 Û
Û 00029   26   26512,E-03 5059,E+00  1500,E-02  1433,E-01  1    00 Û
Û 00030   26   63588,E-03 5057,E+00  1439,E-02  1416,E-01  1    00 Û
Û 00031   26   50997,E-03 5058,E+00  1346,E-02  1399,E-01  1    00 Û
Û 00032   26   46944,E-03 5055,E+00  1350,E-02  1411,E-01  1    00 Û
Û 00033   26   25319,E-03 5056,E+00  1346,E-02  1390,E-01  1    00 Û
Û 00034   26   54461,E-03 5053,E+00  1305,E-02  1413,E-01  1    00 Û
Û 00035   26   29344,E-03 5053,E+00  1733,E-02  1418,E-01  1    00 Û
```

Figure 17: A random cascade displayed by TRIM on which the reasoning explanation was done.



(a)



(b)

Figure 18: Plot of the transferred energy at each collision indicated with the recoil index from the COLLISION.txt file taken from Figure 17. The subfigure (b) is the same as (a) but zoomed in to better see the small variations.

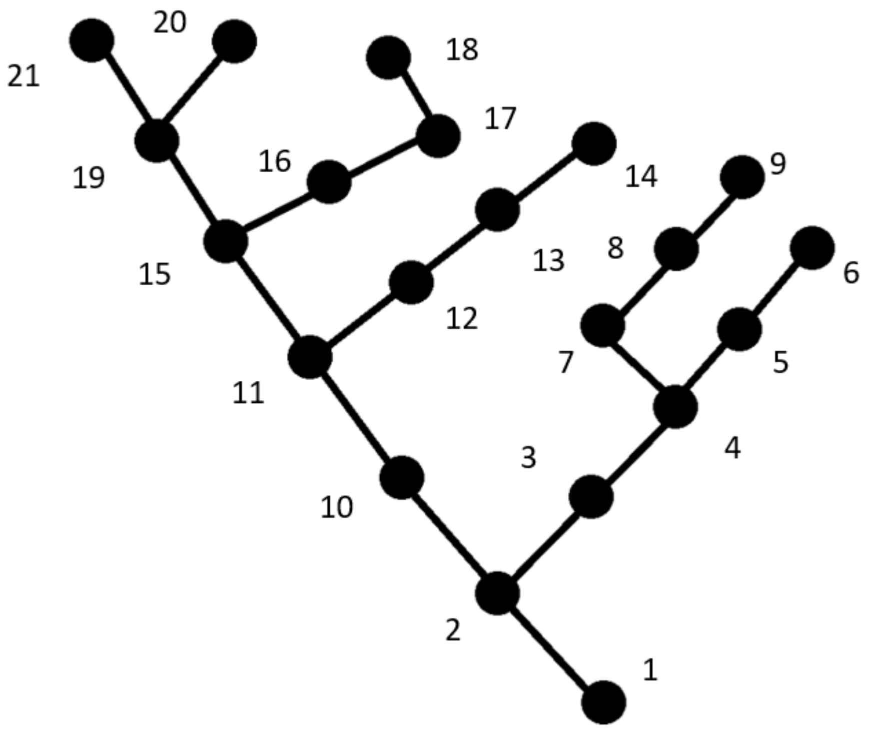
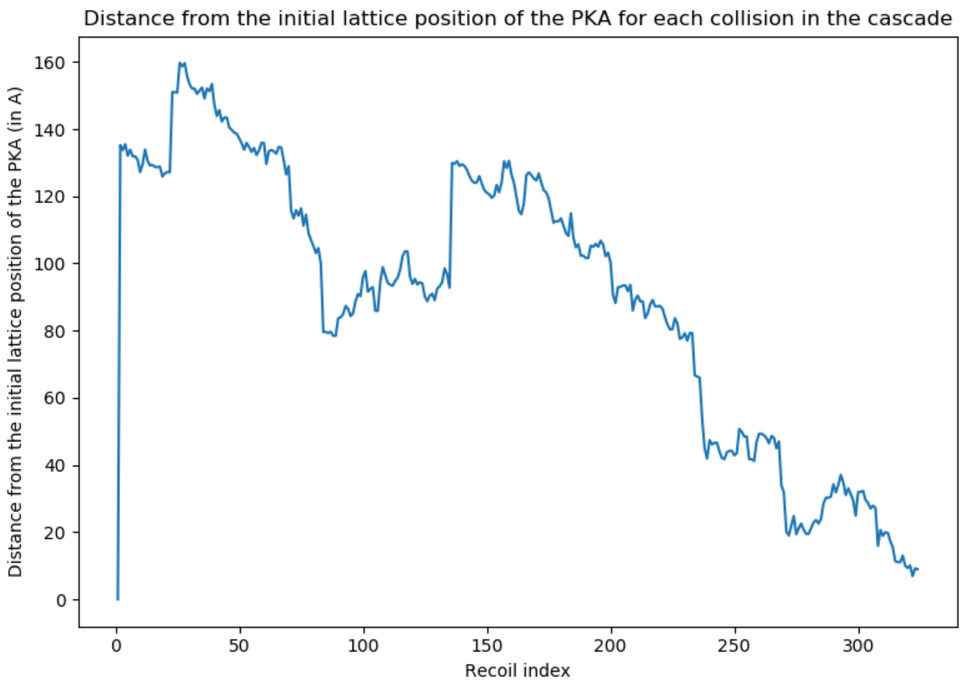


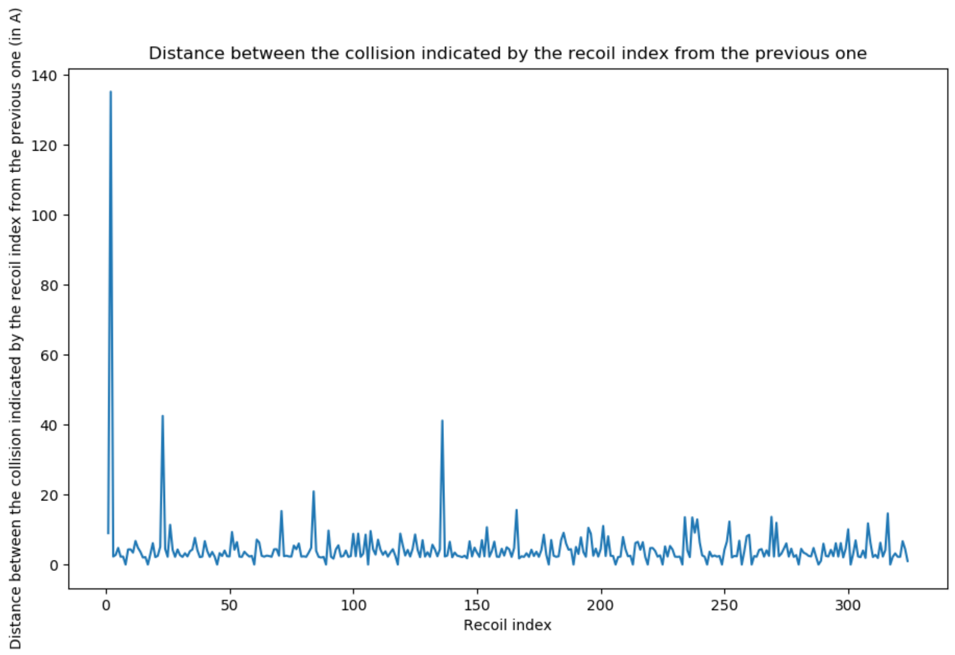
Figure 19: The same example of tree as in figure 16, but with a rearranged index convention.

second vector position is $\vec{v}$=(7 29 132) (in A). When normalized, the vector $\vec{v}$ is equal to $\vec{v}_{norm}$=(0.0519 0.212 0.978), which makes a angle of about 30° between the two vectors $\vec{u}$ and $\vec{v}$. Such an angle does not seem impossible to attain through minor collisions during a distance of length 135 A. It would represent a deviation of 0.22 degree per Angstrom traveled. A minor collision is a collision that do not transfer more than the threshold displacement energy (of about 25 eV) so it will not deviate the incoming PKA (with kinetic energy in the order of 12 keV) from its path by a high angle.

Since this method does not discriminate decisively the second element of the list from the first, we would need a different analyse. In figure 20(a) is plotted the distance of the PKA of the collision indicated with the recoil index from the initial lattice position. In figure 20(b) is plotted the distance between two successive collisions with respect to the recoil indexes. We see that the cascades start quite far away from the initial position of the PKA but then it tends to come nearer and nearer. I computed the same graph for different PKA cascade and the overall shape is the same: high distances at the beginning of the list and more or less gradual decrease as the list goes on. It is highly unlikely that for several cascade, the cascade turns around and come back near to its point of origin. It would suggest that the end of the cascade is displayed first and then the prime recoil. The cascade branching system notation would be made as in the example in figure 21.



(a)



(b)

Figure 20: (a) Plot of the distance of the collision indicated with the recoil index from the initial position of the PKA. (b) Plot of the variations of distance from one collision to the other for the data taken from the COLLISION.txt file from figure 17.

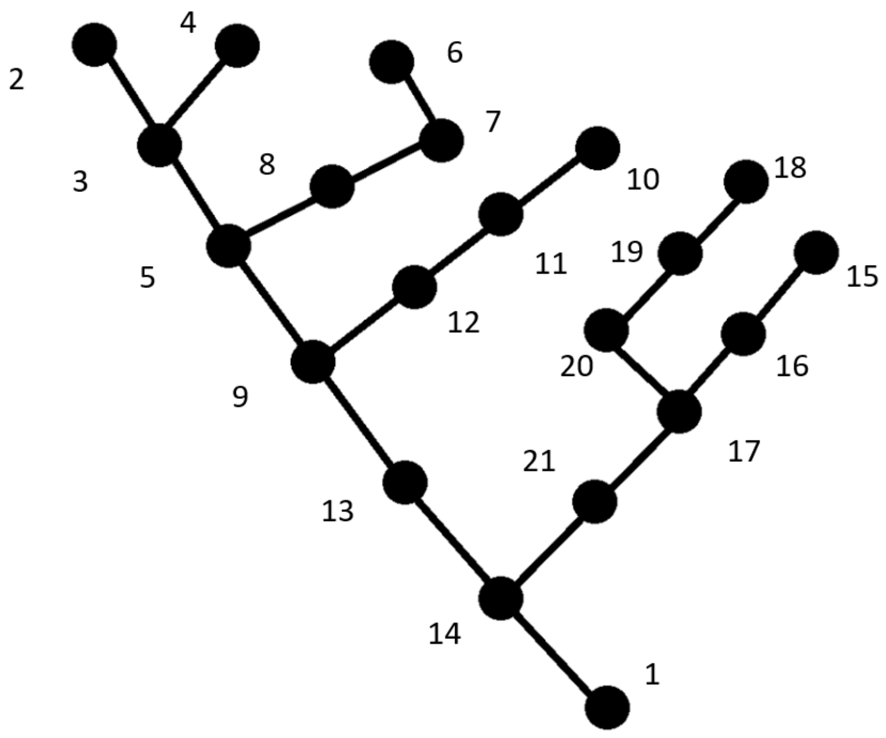


Figure 21: The same example of tree as in figure 16, but with a third rearranged index convention.

7 The creation, implementation and rightness of the algorithms

The whole point to analyse the collision cascades is to analyse the percentage energy loss to electrons, phonons and stored energy. When TRIM is used manually, it is possible to get two files called IONIZ.txt and PHONONS.txt that contain exactly the amount of energy lost to electrons by ionization (for IONIZ.txt) and to phonons (for PHONONS.txt). Even simpler, the TRIM windows directly displays the percentage energy loss distribution. TRIM even sorts out the energy loss distribution for the irradiating ion itself (primary cascade) and the recoils. Yet, when TRIM is used in a semi-automatic way (I used an auto-clicker software), the files cannot be written, the only files capable to be generated through the TRIM.IN file are the Ranges, Backscattered, Transmitted, Sputtered, collision and special EXYZ files. Since a major part of the project aims at evaluating statistically the percentage energy loss distribution, hundreds of simulations have to be made and it has to be automated.

Since the easily readable IONIZ.txt and PHONONS.txt files are not usable for statistical analyses, I had to find ways to estimate as best as possible the values of energy losses. The previous section had the ambition to describe the collection and analyses of the data of interest and the cascade description. In this section, the algorithms used will be described and compared through their performance and correctness. Because the COLLISION.txt data file is very complex to read, some approximations will be made in the different algorithms.

7.0.1 The initial case of mono-atomic materials

For the comparison of the different algorithm output and the SRIM output, let us take a 4 MeV nickel ion projected into a 3 μm thick pure iron material. When computed on SRIM, the result we get for the energy loss is the following :

% Energy Loss	Ions	Recoils	Total
Ionization	44.04	23.9	67.94
Vacancies	0.04	2.89	2.93
Phonons	0.18	28.95	29.13

By "Vacancies", TRIM means the sum of the lattice binding energies lost when an atom leave its lattice position. Yet, on another documentation, TRIM states that this energy is considered to be dissipated as phonons, so in my calculations, the category Vacancies goes into phonons. There also is a stored energy that is not being considered in the TRIM output that is the sum of energies needed to create interstitials. This is the energy that is responsible for the hazardous Wigner effect. In the notation I used, we need to multiply the value of vacancy by the ratio $\frac{E_{fi}}{E_b}$ (because TRIM considers that for each vacancies created, we consume E_b and I consider, which is compatible with TRIM, that for each interstitial created, we consume an energy E_{fi}).

7.0.2 The more complex case of multi component materials

In the case of the irradiation of a mono-atomic material, the ionization energy is easily calculable because all atoms in the recoil cascades are the same. More specifically, when we use the $S_e(E)$ function, it was always the electronic stopping power of the atom that the material is made of into the same material. For example, in the specific case that I studied in depth, namely iron stopping power inside pure iron, we had no need to invoke, in the recoil cascade calculations, the electronic stopping power of the ion specie, like nickel, inside iron because it only concerns the primary cascade that is already sorted out in the collision file through the PKA list. Yet, for multi-atomic materials, the subcascades needs to be exploited more in depth in order to know what $S_e(E)$ function to use at what time.

The generic multi-atomic materials that were studied are oxides written generally as X_nO_m, in which n and m are integers. It means that sometimes X atoms would be knocked off but some other time it may be O. The paths that they would follow after the impacts correspond to different electronic stopping powers. We need to sort the subcascades out by analysing the COLLISION.txt file even further. It provides for all recoil cascades caused by each PKA in the form of lines containing all major collisions (those who ends as a displacement creation) information. The lines contain, as a reminder from the previous sections, the transferred energy to the knock-off atom before that the lattice binding energy has been consumed, the nature of the atom (X or O) and the coordinates of the collision described by the concerned line.

7.1 The algorithm by length

7.1.1 For a mono-atomic target

In the figure 20, we see that very sharp variations in term of the distance between the collisions localisation and the initial PKA lattice position are observable. They correspond to "jumps" from one cascade to the other. Usually, the distances between two successive lines in the COLLISION.txt files are of the order of a few angstroms. When jumps of tens of angstroms are seen, we may expect that a different cascade is analysed. It may be a SKA cascade or even a higher generation cascade. In the simplest case, we can consider that sufficiently high changes are due to a analyse of one SKA cascade to the other with no further subcascade analyse. Then the energy losses on the path of the PKA can be calculated between the different SKA initial lattice position.

The first question that needs an answer is about the minimum distance that will trigger the process of changing from one SKA cascade to an other (inside a PKA cascade). It cannot be the same for a 10-collision cascade and a 10 000-collision cascade because in the latter, there would inevitably be variations of the order of tens if not hundreds of angstrom inside the same SKA cascade that must not be taken into account. In the first case, a variation of ten angstroms surely represents a jump from one SKA cascade to an another one and has to be taken into account. An approximation could be to consider the maximum distance of any collision from the initial lattice position of the PKA and trigger the change of SKA cascade every time the variation of distance from one line to the other exceed a fraction $\frac{d_{max}}{p}$ of this maximum distance d_{max}. The table below shows the result of energy loss percentage distribution "% Energy Loss" for p taking different values and for the "Recoils" only:

p	2	3	4	4.5	5	10	15	20
Ionization	21.1	22.4	23.3	23.8	24.2	34.2	40.3	49.9
Phonons	31.2	29.9	29.0	28.5	28.1	18.1	12	2.4
Vacancies	2.9	2.9	2.9	2.9	2.9	2.9	2.9	2.9

These values only concerns the "Recoils" and do not have to be compared to the primary cascade ("Ions"). Let us consider the percentage energy loss distribution for the "Recoils" from the TRIM table which was 23.9% for ionization, 2.89% for vacancies and 28.95% for phonons. It seems that the best values of p for having the best approximation of the real calculation of TRIM are between 4 and 5. For the rest of this section, I will use p=4.5 because it offers a maximum error of 1.5%. Up to this point, because it is possible to modify and adjust what components are taken into account in the calculations, I could determine that the nuclear stopping power does not have to be considered in order to get similar results as the TRIM output tables. Let us take a look at the common result (the analysis of the primary cascade is independent on the number p we chose for the model) for "Ions".

% Energy Loss	Ions
Ionization	44.7
Vacancies	0.04
Phonons	0.11

The result of the ionization I found is close to the one displayed by TRIM. The method of calculus for calculating the electronic stopping power is confirmed but slightly overestimated. This is normal due to the fact that the calculation of the integral in equation 12 was made from above. Yet, even by increasing the refinement of the calculation, we do not get different results. The ion loses energy from ionization and collisions. The transferred energy to the atom is not counted in this step because it will be latter on converted mainly into ionization and phonons. Ionization is clearly the main source of energy loss while the phonon creation are null (or almost null). Let us try again this algorithm on another random seed number generation to see if the results are still close to the one provided by TRIM (on the left) and by the algorithm by length (on the right):

% Energy Loss	Ions	Recoils	Total
Ionization	55.87	15.06	70.93
Vacancies	0.04	2.61	2.65
Phonons	0.21	26.21	26.42

% Energy Loss	Ions	Recoils	Total	Error
Ionization	56.6	15.3	71.9	1.4%
Vacancies	0.04	2.62	2.66	0.3%
Phonons	0.11	25.3	25.4	-3.9%

The relative error of the total energy loss distribution between the TRIM direct output table and the one obtained by the algorithm by length is at worst 4% and can therefore represents a very good approximation for medium calculation times.

7.1.2 For a diatomic target

In order to test the algorithm by length for diatomic targets, let us consider the case of a nickel ion of 4 MeV projected on a target of Al_2O_3 of 3 μm thick. Here are the result for the direct TRIM output table on the left and the analysis through the algorithm by length on the right. The algorithm was of course to be modify to correctly consider the diatomic nature of the target.

% Energy Loss	Ions	Recoils	Total
Ionization	62.79	18.57	81.36
Vacancies	0.06	1.19	1.25
Phonons	0.34	17.05	17.39

% Energy Loss	Ions	Recoils	Total	Error
Ionization	63.5	15.6	79.1	2.8%
Vacancies	0.04	1.19	1.23	-1.6%
Phonons	0.28	19.4	19.68	11.6%

The electronic stopping power considered in each recoil cascade caused by a PKA was the one of the atom of the PKA for the whole cascade. This is a rough approximation because an aluminum atom as a PKA could possibly transfer energy to an oxygen atom that would also lose energy to the electrons. I basically considered that all collisions that were detected by the algorithm would be caused by the PKA itself. It has its importance in this case because the ionization energy loss is quite high (>80% of the total energy loss) and variations of only few percents would have a large impact on the error concerning the phonon energy loss (=11.6%) because its value is much smaller (<20% of the total energy loss). But anyway, the order of magnitude is the same and for a first order approximation, the output of the algorithm is satisfactory. Let us consider an other example for the same irradiation conditions and the same target.

% Energy Loss	Ions	Recoils	Total
Ionization	54.38	27.93	82.31
Vacancies	0.05	1.13	1.18
Phonons	0.28	16.24	16.52

% Energy Loss	Ions	Recoils	Total	Error
Ionization	53.2	26.1	79.3	3.7%
Vacancies	0.03	1.10	1.13	-4.2%
Phonons	0.24	19.3	19.54	15%

The error are higher than in the first case but of the same order. It would seem that the algorithm by length gives good approximation for mono-atomic targets with very low error but for diatomic targets, the error percentages are increased but with a predictable 10% for phonons and 3% for ionization in the case of nickel ions on a Al_2O_3 target. We could one last time test the algorithm with a completely different set-up : Au ion at 12 MeV on a 5 μm thick ZrO_2 target.

% Energy Loss	Ions	Recoils	Total
Ionization	42.82	28.61	71.43
Vacancies	0.03	1.76	1.79
Phonons	0.09	26.69	26.78

% Energy Loss	Ions	Recoils	Total	Error
Ionization	42.7	40.3	83.0	14%
Vacancies	0.02	1.76	1.78	-0.5%
Phonons	0.14	15.1	15.24	43%

For this case study, the errors are unacceptable. They could be decreased if we adjust the p variable. It seems that each set of ion and target has its own p value that best approximates the energy loss distribution. For gold irradiation on ZrO_2, a value of p between 2 and 2.2 gives better approximations. That means that the algorithm by length could be used but it would need a systematic pre-study to determine an adequate value of p and only for mono-atomic targets. Yet, the value of 2 or 2.2 for gold ion on ZrO_2 is unstable and works only in that case. This method is hazardous because it is highly approximative but statistically, a p coefficient of 4.5 gives good results and since it will be the same coefficient for the analysis of all types of simulation, the error relatively to one to the other should be mitigated. This algorithm gives only orders of magnitude but has a low complexity and uses an understandable process.

7.2 The algorithm by whole cascade sorting

The algorithm could obviously not be constructed for each cascade and the algorithm would not know at the beginning of the analysis how many cascades it contains. We need to use a recursion algorithm, that consists in calling the function inside the function itself. The function takes as input the list of energies transferred and the localisation of the collisions and must output the energy losses to electrons. The function take the whole energy list as input and would call itself to analyse the rest of the energy list. The function solves only one cascade at a time. Each cascade has to be understood in the way defined in the section A first attempt to sort out theoretically the cascade branching system, which is only the series of collision with the common atom that caused them. Very unfortunately, the way TRIM shows the successive collisions like displayed in figure 21 cannot be easily analysed. The display method used in figure 16 and 19 would be better especially because the order of appearance in the list follows the cascade from up-to-bottom and not from bottom-to-up. The principal advantage of a up-to-bottom display is that the electronic stopping power can be calculated alongside the descent in the

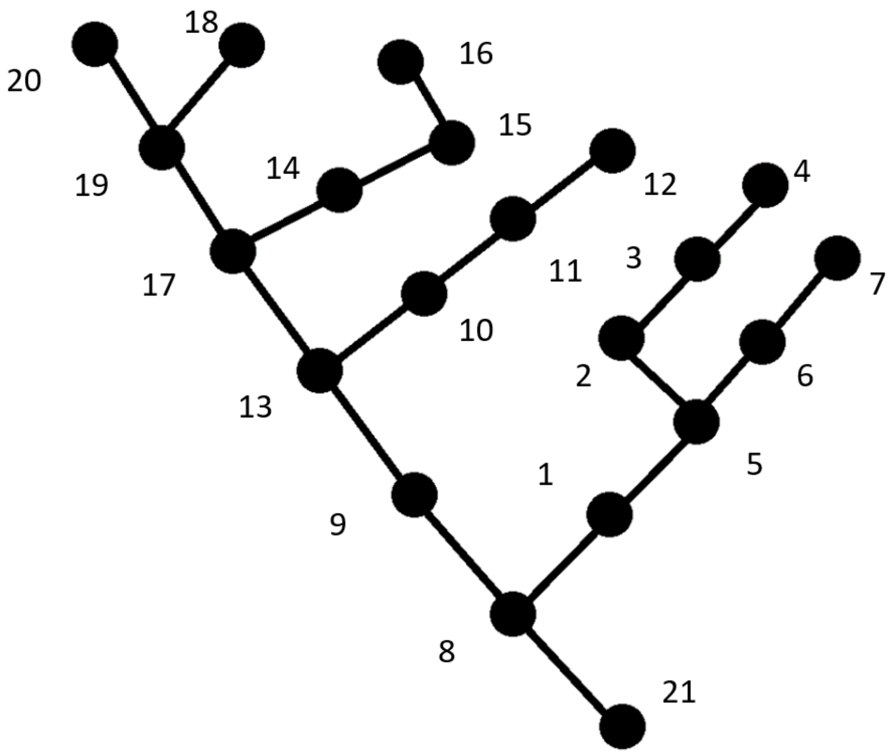


Figure 22: The same example of tree display as in figure 21, but with all indexes being reversed.

cascade because the electronic stopping power depends highly on the energy of the atom and the energy variation of the recoil atoms changes a lot (from a tens of eV to occasionally a few Mev). A constant value approximation would be too rough for an approximation.

An idea to counter this problem is to reverse the whole list after having extracted it from the TRIM output. The tree we would obtain, if we consider once again the same tree analysed before, is displayed in figure 22. As we can see, the indexes are now increasing when the cascade is being analysed deeper and deeper. Yet, every time a new subcascade is being generated, the indexes will follow it. Once all subcascades further generated from this branch are explored, the indexes will come back at the node were the initial new subcascade was generated and it will go down once again. This is a once again a major problem because the initial energy of the cascade we are currently exploring is unknown. Let us take the example in figure 22. At the node one, the energy that is displayed on the corresponding line corresponds to the transferred energy from the atom knocked out at the node 8 to the atom that was in the lattice position at the node 1. But the line does not contain any information on the initial energy transferred to the atom that caused the cascade under study (here the atom ejected from node 8). That means that the ionization cannot be computed.

In our case, the cascade is very small and can be analyzed by hand so we know that the node 8 is at the origin of the cascade C(2). So we could first read the line number 8 and take the energy transferred to the atom at the collision K(2). But in the code, with hundreds and sometimes thousands of line, it has to be found starting from the first position 1. We need to calculate through the energy exhaustion method the index that indicates the start of the current cascade (once all the energy of the initial kinetic energy of the recoil is consumed, the cascade stops necessarily and the previous cascade not fully analyzed continues). Here, there is an incompatibility, we need to find an index value but to do so, we would also need the energy value it contains. The exact exploitation of the cascade seems impossible due to this problem.

By continuing to study the algorithm problem through the example of figure 22, a possible method to counter this issue is to find the index 8 from the node 1 without using the energy exhaustion method. The only physical quantity we have left to analyse the cascade is the distance. Usually when the cascades develop, they tend to spread and to leave the point of origin. Of course, this is not always true, the atoms may bounce back and come closer to the point of origin. But we could make the assumption that in most cases, it will not come back and the distance may be used to determine what was the node index from which the cascade begun. If all adjacent collisions are assumed to be closer than 3 A (or any other value), we could do it. With a while loop that checks for the distance between the first node and the successive ones. But in the end, the assumptions are too rough and the code has so many conditions to check that it would be inefficient in addition to being false. In the end, the complete detailed and exact reconstruction of the cascades structure from the COLLISION.txt file seems impossible. We will only be able to approximate the energy loss distribution with algorithm containing more or less acceptable assumptions.

7.3 The algorithm using the phonons.

Since positioning the collisions and constructing the cascade branching system is very complex with the limited information given by the TRIM output, I came back to an interpretation of the TRIM data made previously. Since there are very few collisions displayed in the COLLISION.txt files for which the transferred energy is smaller than the displacement energy, I assumed that the minor collisions were neglected. Yet, when I calculated the nuclear stopping power in addition to the electronic stopping power for the primary cascade, the energy loss was too high. Due to this fact, if I neglected the nuclear stopping power between the listed collisions on the primary cascades, I did the same assumption for the PKA cascades and all subcascades. On the other hand, all collisions written in these files are associated to a displacement event. This would imply that some minor collisions might happen but are not reported.

In this algorithm, I will as before neglect the minor collisions (not listed in the TRIM output files) but, differently from before, consider the phonon calculation first and then deduce the ionization for the recoils cascade. The calculation of the primary cascade is unchanged. The phonon sources are the end of each branches of the tree of collision, the leaves of a tree to speak metaphorically. The transferred energy to atoms before the extremities of a branching system will not be fully converted to phonons since the recoil atoms will lose energy to electrons in the subcascades they cause. The only energy loss into pure phonons is the transferrerd energy during minor collisions that are not reported. Yet, at the extremities of the branching system, where the kinetic energy of atoms is small (tens of eV), the electronic stopping power is negligible as a first approximation compared to the energy lost to nuclei. This algorithm will simply look in the file for collisions where the transferred energy is smaller than p*E_d with p being a real number that has to be determined in order to optimize the results accuracy. For all these collisions, it will consider that all the transferred energy except the formation energy of intersitital will go into phonon. For all other collisions that do have a transferred energy higher than p*E_d, the only energy considered to be added to phonons is the lattice binding energy (initially stored but dissipated into phonons when an atom leaves its lattice position) according to the TRIM output. The calculation for the binding energies do not change and the ionization for the recoils is only calculated as the remaining energy to reach the initial ion energy for the conservation of energy to be respected.

The result for a 4 MeV nickel ion projected into a 3 μm thick pure iron material are displayed below. The comparison for recoil cascades for different value of p is displayed below :

p	2	2.5	2.8	2.89	2.9	3	3.2	3.5
Vacancies	2.9	2.9	2.9	2.9	2.9	2.9	2.9	2.9
Phonons	19.4	24.6	27.4	28.1	28.2	29.0	30.5	32.5
Ionization	32.9	27.7	24.9	24.1	24.0	23.3	21.8	19.7

The value of p that seems to optimize the accuracy of the algorithm is 2.89. The results are displayed on the right side with the TRIM output table on the left as usual for a direct comparison.

% Energy Loss	Ions	Recoils	Total
Vacancies	0.04	2.89	2.93
Phonons	0.18	28.95	29.13
Ionization	44.04	23.9	67.94

% Energy Loss	Ions	Recoils	Total	Error
Vacancies	0.04	2.91	2.95	-0.7%
Phonons	0.11	28.11	28.22	3.12%
Ionization	44.69	24.13	68.82	1.28%

The comparison for a diatomic target is made with a Au ion at 12 MeV on a 5 μm thick ZrO_2 target with a p coefficient as high as 4. The ionization is once again overestimated even with a large p coefficient. This algorithm is even worse than the algorithm by length because it requires a different p for each simulation. That means that for two different Ni ion at the same initial energy on the same target, the p coefficient neededd will be different. It was not the case for the algorithm by length. This algorithm is completely dismissed and cannot be used at all for the simulations.

% Energy Loss	Ions	Recoils	Total
Ionization	49.49	28.89	78.38
Vacancies	0.03	1.31	1.34
Phonons	0.11	20.18	20.29

% Energy Loss	Ions	Recoils	Total	Error
Ionization	49.72	38.56	88.28	11.2%
Vacancies	0.028	1.31	1.34	0%
Phonons	0.14	10.24	10.38	-48.8%

8 Results of the neutron and ion codes

8.1 Details on the strategy to assess the damage comparison between the two types of irradiation

The term "damage" used to compare the ion to neutron irradiation has to be precised. We consider an arbitrary reference area on which the irradiation is done. This reference area is common in both cases and is equal to 1 cm^2 but has no physical meaning as it will be cancelled in a future ratio. Since the penetration in both cases is largely different, the reference depth considered will not be the same. The average penetration for heavy ions in the tens of MeV range in most material tested is in the order of 5 micrometers whereas a block of approximately 30 centimeters blocks 90 % of the neutron particles according to the results of the neutron code. When multiplying the reference area by the reference depth, we get a reference volume in which displacements will take place.

These displacements have to be averaged on the whole reference volume in order to get a number of displacement per atom (dpa). The neutron and ion codes, for mono-atomic or diatomic compounds (a total of four codes), will run separately. The principal output will be the average displacements per atom per projectile. Each projectile (neutron or ion) will cause a certain number of displacements in the reference volume but this quantity varies a lot between each simulation and in order to get statistically relevant results, the codes will test hundreds of projectiles until the average displacements per atom per projectile stabilizes. The whole graph representing the average displacements per atom per projectile versus the number of projectile simulated is a damped oscillation like the one showed in figure 23. The final value of average displacements per atom per projectile taken is the last indicated on the far right of each generated graph by the codes.

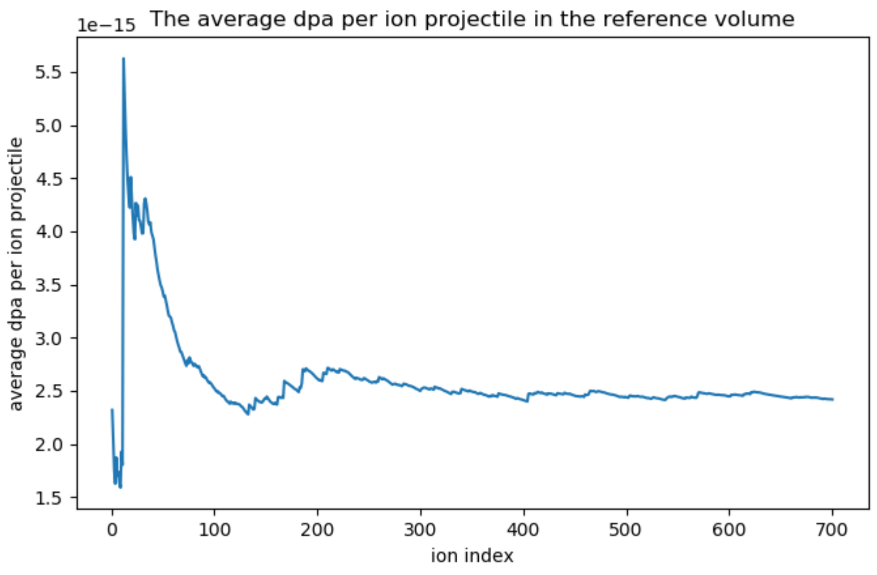


Figure 23: The average displacement per Au projectile on a ZrO_2 target updated after every simulation traced with the ion index.

An another remark has to be made on the reference volume definition. Since the energy dissipated in the material by ion irradiation is dependent on the depth due to the presence of the Bragg's peak, we have to consider that the average value would be very rough and hide a huge disparity between the energy dissipated at the skin of the material and at the depth where the Bragg's peak occur. The average displacements per atom per projectile we would like needs to represent an average of a more or less constant, or slightly oscillating, energy dissipation inside the material. That is why the codes only count the displacements before the beginning of the Bragg peak for the ion irradiation. This localisation is of course different from one simulation from another but a matrix has been made that shows, on average, the depth where the Bragg's peak starts to occur. It has been determined by running manually, for each combination of material and ion, 20 simulations and to take more or less the minimum deposition depth of the ion. By doing this selection of the value we want, we know that the damage is more or less homogeneous in the whole reference volume.

For neutron irradiation, a similar selection has to be performed but this time, there is no such thing as a Bragg's peak. There is a set of cluster where the damage is done when the neutron goes deep enough inside the target material. But most of the cases, the neutrons are backscattered only after a few collisions like it can be seen in the figure 9. In these condition, the gradient of displacement per slices of material is too strong. If we want an homogenized distribution of displacements for a correct comparison, we need to consider only displacements made after a certain depth (like one centimeter so that we avoid to consider the overpopulated area at the skin region). An inflated value at the skin would enhance the average in the whole reference volume and that would overestimate the damages done in the bulk of the material. Then the code has to consider a second reference depth to know where stops the reference volume. Since for a slab of iron of 30 centimeter thick, only 10 % of neutron are transmitted with a relatively low energy compared to their initial energy, it seems reasonable to assume that the amount of damage done after a depth of 30 centimeters is negligible compared to the damage done in the reference volume defined by the reference area between 1 and 30 centimeters. Then the dpa is averaged over the whole reference volume.

8.2 The special case of hydrogen ions irradiation

It was thought to be a good option to consider proton irradiation to best describe the neutron irradiation. The first reason was that if an hydrogen atom is sufficiently accelerated (to about 200 MeV [14]), then the proton would go deep enough (to about 20 cm inside the target material). This would mean that we could at best get the typical range for the neutron irradiation by with an ion. That would dismiss all spacial distribution issues we have with the defects generation comparison. Yet, when simulating with TRIM the behavior of hydrogen irradiation on any of the materials studied above, the outcome is totally different in terms of energy loss distribution and displacements generation. The proton, according to the simulations on TRIM, lost most of its energy (>99.8%) to ionization due to electronic stopping power. Moreover, the amount of displacements generated in the COLLISION.txt file is low compared to the other irradiation. This implies that the hydrogen ion just travels through the material while being stopped by electrons but without any significant number of collisions on its path. Despite the appealing penetration depth on the hydrogen irradiation, it creates too different damage to the material to be able to simulate correctly neutron irradiation.

8.3 Results and comparison

8.3.1 Qualitative time reduction assessment

Once every graph are drawn and the average displacements per atom per projectile determined for all combinations of (Fe, Al_2O_3, ZrO_2 and Y_2O_3) and (neutron, Ni at 4 MeV and Au at 12 MeV), we can continue the reasoning. The average displacements per atom per projectile (ADPP) table summary is displayed below :

ADPP	Neutrons	Ni at 4 MeV	Au at 12 MeV
Fe	2.9×10^{-21}	1.15×10^{-15}	6×10^{-15}
Al_2O_3	7.8×10^{-22}	5.5×10^{-16}	2.5×10^{-15}
ZrO_2	1.5×10^{-21}	5.0×10^{-16}	6×10^{-15}
Y_2O_3	1.25×10^{-21}	5.0×10^{-16}	2.5×10^{-15}

The calculation power of my personal computer being limited, the total numbers of neutron and ion simulated (thousands for neutrons but only hundreds for ions) are small compared to the real life application like the fluence encountered in a nuclear facility. Since the average displacements per atom per projectile are stabilized value of the damped oscillations described above, we can use them to predict more or less accurately that a very high number of neutrons N_{high} will cause an approximate number $N_{high}*ADPP_{neutron}$ of displacement per atom in the reference volume considered.

If we completely dismiss the spacial distribution of defects, then the comparison of the damage between neutron and ion irradiation is quite simple now that the average displacements per atom per projectile is known. Let us say that the nuclear department states that the material will be subjected to a fluence of N_n neutrons during a certain period of time and that they would like to know the fluence of ions required to get an equivalent amount of displacements without having to spend years of irradiating the material with neutrons. Since N_n*ADPP_n is the approximate number of dpa caused by the total fluence of neutrons, the fluence N_{ions} needed to get the same dpa is given by:

$$N_{ions} = N_n \frac{ADPP_n}{ADPP_{ions}} \tag{21}$$

The ratio N_{ions}/N_n for all materials if given in the table below. The required conversion matrix to simulate the neutron irradiation with ions, with the dpa being the only factor taken into account, is determined. Practically, if we want to know how many ions we need to get the same dpa as neutron irradiation, we just need to multiply the N_n by the values in this table. Of course, the result is strongly dependent on the neutron spectrum considered. For this master thesis, I remind the readers that a Maxwell-Boltzmann distribution was considered that simulates the spectrum of fissions neutrons in a reactor. If a different or more precise neutron spectrum needs to be considered, it can simply be done by substituting the neutron sampling part (only one line in the code in the section 12)

$N_{ions}/N_n(*10^{-7})$	Ni at 4 MeV	Au at 12 MeV
Fe	25	4.8
Al_2O_3	14	3.1
ZrO_2	30	2.5
Y_2O_3	25	5.0

All the necessary data was calculated up to this point and we just need calculate the effective time reduction when simulating neutron damage with ion irradiation. In the reference [1], a possible and attainable fluence for ions can be $6 \times 10^{16} ions.cm^{-2}$ in only one hour for Ni ions. The reference [13] confirms this ion beam flux order of magnitude but may vary from one device to the other. This is the typical representative value taken for the ion fluence for the rest of the thesis. Similarly, the typical neutron fluence in a pressure vessel can be obtained in the reference [5] as $3.5 \times 10^{19} neutrons.cm^{-2}$ in the entire lifetime of the vessel (40 years in our case). The flux inside the core of pressure vessel is even larger and can be considered over the course of 40 years to be approximately equal to $3.5 \times 10^{22} neutrons.cm^{-2}$ [2]. The table below shows the time required for ion irradiation to cause the same amount of defects as a 40 years exposition to neutron irradiation in the in-core components. The first step is to calculate the total number of ions required and then by dividing by the ion flux, we find the wanted time t_{ions}.

t_{ions} (in s)	Ni at 4 MeV	Au at 12 MeV
Fe	5250	1010
Al_2O_3	2940	650
ZrO_2	6300	530
Y_2O_3	5250	1050

The values given in seconds are quite small and there are several explanations to this fact. Firstly, t_{ions} depends on the values on fluxes taken and they are only order of magnitude found in articles dealing with ion beam generators but do not necessarily corresponds to existing devices delivering either Ni at 4 MeV or Au at 12 MeV. Still, the fluxes values are the same for all materials and ion taken meaning that the relevant parameter is the relative t_{ions} between different cells in the table and not the absolute values. If someone deems my work worthy of being use further, he can easily modify either the irradiating ion properties or the fluxes. All the parameters are modifiable at the beginning of the code (see the Appendix for more details). An user-friendly interface would make it even easier.

8.3.2 Energy loss distribution results

Now, in order to assess how the material reacts to the irradiation, we need to consider the energy dissipation distribution for the three types of specific irradiations considered.

For neutrons (in %)	Stored Energy	Phonons	Ionization
Fe	3.19	93.97	2.83
Al_2O_3	1.22	90.17	8.61
ZrO_2	1.64	93.01	5.35
Y_2O_3	1.19	94.66	4.15

For the neutron irradiation, more than 90% of the energy is dissipated as phonons. This is due to the absence of colombic interactions between the neutrons and the atoms. The neutron being a neutral particle, it will only collides with nucleus on the primary cascade with no ionization produced. Only recoil cascades cause ionization but since the energy transferred to PKA is generally small compared to the initial neutron energy, the ionization produced in recoil cascades is still small because the more energetic a particle is, the more ionization it produces in general.

For Ni at 4 MeV (in %)	Stored Energy	Phonons	Ionization
Fe	7.1	19.21	73.68
Al_2O_3	4.07	13.36	82.57
ZrO_2	4.12	18.14	77.74
Y_2O_3	3.99	17.61	78.40

For Au at 12 MeV (in %)	Stored Energy	Phonons	Ionization
Fe	8.35	21.62	70.03
Al_2O_3	4.68	13.57	81.75
ZrO_2	4.72	21.03	74.25
Y_2O_3	4.51	20.62	74.87

For ion irradiation, because the colombic interaction exists since the primary cascade, the ionization is much larger than in the case of neutron irradiation. We can witness some variation between the materials. For instance, Al_2O_3 has an even higher ionization percentage compared to the other materials. ZrO_2 and Y_2O_3 have similar energy loss distribution and is intermediate between Al_2O_3 and Fe. In iron, the ionization energy loss percentage is smaller than for the oxides.

If the goal is to get an energy loss distribution for ions as close as possible to the neutron irradiation, we have to choose which ion (here, in our limited study, between Ni and Au) satisfies the best this criterion for each material. Since we need the higher energy loss into phonons to best simulate the neutron irradiation, it is in all cases Au at 12 MeV than meets the requirements better than Ni at 4 MeV.

Phonons are collective vibrations of atoms that occur in a crystal with certain frequencies and has many influence of the material properties like the capability to propagate heat, electricity and sound. They may has a indirect link to the temperature since it makes the atoms move from their lattice position like a thermal vibration of an atom would cause. Phonons may have an influence on the heat. Yet, what is sure is that ionization energy loss directly goes into electrons and that can be considered as thermal energy.

Some major incident in the nuclear facilities were attributed to the Wigner effect like the Windscale fire. The Wigner effect is linked to the release of stored energy at high temperature, like annealing. Comparing the energy lost to stored energy may be interesting. For instance if we measure that the stored energy for a ion irradiation is smaller than the real stored energy with neutron irradiation, it may be an issue since the possibility of a release of energy is more critical than expected. Yet, it seems to be a reoccurring fact that the stored energy systematically is higher for ion irradiation compared to neutron irradiation. As a reminder, the stored energy is the sum of all the distortions around an interstitial atom. This energy is considered to be, for one interstitial atom, three time the lattice binding energy like detailed in the previous sections.

Another remark has to be made on the fact that the stored energy is higher for pure iron than for biatomic oxides and that occurred both for neutron and ion irradiation. If we take a closer look on the values of displacement and lattice binding energies shown in the table below, we can see that only oxygen has a different value for the displacement energy compared to the others. These values are taken from the TRIM database in order to have the same viable source for all simulation and not multiply the references that may have different methodology of determination of the E_d and E_b energies [7]. Yet, they may not be exact since it seems sound to state that the lattice binding energy depends on the chemical surroundings of the atomic specie considered. This energy for an gold atom is for instance dependent on the structure of the material (crystalline, amorphous or in between) and by extension to its surroundings.

Energies (in eV)	Fe	Al	Zr	Y	O
E_d	25	25	25	25	28
E_b	3	3	3	3	3

The mean value of E_b and E_d for a Au atom in Au_2O_3 will be different from the one in pure gold. This is not taken into account by TRIM but may have a significant importance. Anyway, in the case of a compound, TRIM correctly takes into account the effect of the different atomic masses for the transferred energy upon collisions but it does not consider the different binding energies configuration that would require more input parameters and a complex analysis (namely molecular dynamics). I just took the TRIM database without questioning it for two reasons :

1. The evaluation of E_b (a thermodynamic equilibrium value) and E_d (an out-of-equilibrium quantity) for different atoms in different compounds would require a great amount of work and would represent a potential thesis by itself.

2. We are interested in the comparison analysis so the homogeneity of the comparison basis is more important than its accuracy.

Since the calculation of the stored energy in the code is done for all interstitials with the assumption that all displacements (vacancy creation events + replacement events) cause a formation of one interstitials because the materials are considered thick enough to avoid transmission. More precisely, each knocked-off atom ends as an interstitial somewhere except if this atom causes a replacement event. Basically, the stored energy is the total number of displacements minus the total number of replacements, all multiplied by the formation energy of interstitials for mono-atomic target like depicted in the following equation.

$$E_s = (N_{displacements} - N_{replacements}) * E_{fi} = N_{vacancies} * E_{fi} \tag{22}$$

It is slightly more difficult for biatomic targets since we have to select the type of atom (X or O) that cause the displacements but it is well implemented in the code and everything is taken into account in order to give the good result. Yet, in the equation 22, there is only $E_{fi} = 3E_b$ that is equal to 3 eV for all atoms. This means that the discrepancy in the results does not come from the constant values for each specie but from the total number of $N_{displacements} - N_{replacements}$. One explanation could be that because the oxygen's displacement energy is equal to 28 eV instead of 25 eV, the requirement for causing an oxygen atom to displace is more demanding for the incoming particle and thus, a lower amount of displacement event are existing. This phenomenon would cause the stored energy to be lower in the case of irradiation on an oxide compared to on a pure specie with a lower displacement energy.

In order to deepen the comparison between the behavior of iron and the tested oxides, it has to be said that the energy loss to phonons are similar for all materials, except for Al_2O_3, for a given irradiation. So the energy loss to phonons discrepancy between iron and oxides is not counterbalanced by more or less phonons but by more or less ionization as it would appear. This may sound counter intuitive if the reasoning made in the previous paragraph is correct. In fact, if a collision between an incoming atom at 26 eV on an oxygen atom cannot cause a displacement because $26 < E_d = 28$ eV, the former would be expected to continue its path and losing the major part of its remaining energy into phonons since the energy loss to phonons is generally predominant at lower energies compared to ionization losses as seen in the theoretical framework. In order to take a closer to the influence of the value of E_d and the erngy loss distribution, I considered a value for the threshold displacement energy of iron of 40 eV cited in some sources including [10].

Fe with $E_d = 40eV$	Stored Energy	Phonons	Ionization
Ni at 4 MeV	4.20	22.4	73.4
Au at 12 MeV	4.95	25.26	69.80

We can see that increasing the threshold displacement energy E_d has the effect of decreasing the stored energy. When considering furthermore the equation 22, we can tell that, with E_{fi} remaining unchanged, $N_{vacancies}$ is necessarily decreased. This observation is coherent with the reasoning made in the paragraph under the same equation 22. In other words, all knocked atoms will end up as interstitials (except if they cause a replacement event) but they may knock other atoms on their path that could also create interstitials. The phenomenom of knocking-off an atom from its lattice position is more difficult if E_d is larger. This means lower energy loss going into stored energy (energy of distortion around interstitials) and more into phonons, the ionization percentage remains more or less the same (in the case of Ni at 4 MeV on pure iron : $\approx$ 73.5 %). Whatever the physical explanation, it remains sure that the ionization is greater in the case of oxides than for iron for the three types of irradiation, with the Al_2O_3 being particularly highly prone to energy losses into electrons.

Attaining a similar phonon to ionization energy loss ratio seem rather impossible due to the high energy losses of the ion to the electrons on the primary cascade. The idea to say that Au at 12 MeV is better than Ni at 4 MeV because the former has 2-3 percent more energy loss distribution to phonons than the latter is not satisfactory. Even if 21% is closer to 90% than 19%, it is not enough to conclude that using irradiation of Au at 12 MeV is a good simulation of a neutron irradiation. That is why the study of the stored energy was interesting. This type of energy loss is greater in the case of ion irradiation. That may be a positive observation since the experimental conditions are more critical than the real case scenario. In fact, as explained above, the stored energy may be suddenly released causing catastrophic failure and is a critical energy loss compared to ionization or phonons whom energies may be dissipated as heat for example. Stored energy may be released through annealing or defects mobility. Modelling the evolution of defects is an even more difficult task than modeling the creation of defects. It is the way in which the defects evolve (by diffusion, annihilation, coalescence in different ways and/or others) which determines the evolution of the microstructure and therefore the modifications of the macroscopic properties. Yet, these phenomenons are beyond the scope of this master thesis and will not be considered.

9 Conclusion and future work

The objectives defined in the introduction were more or less attained and completed. If we only focus our attention to the creation of defects, it is possible to create conversion tables between neutron and ion irradiation. We saw that all ions are not equivalent and some are more suited to simulate neutron irradiation than others in the limited case of Ni at 4 MeV and Au at 12 MeV. The proton irradiation was also explored but the difference was too high with the neutron irradiation. One of the many key concepts used during this work was the Average Displacement Per Projectile that is the statistical convergence value of the number of displacement the considered projectile causes on average in the target. This variable is at the basis of the conversion tables in order to determine the equivalent time required under ion irradiation to reach the same amount of damage as the neutron irradiation.

A detailed analyse was made in order to determine the energy loss distribution of the incoming projectile, both for ions and neutrons. This distribution is divided between the stored energy as the distortion energy around all interstitials created, the phonons generated and the ionization, which is the energy lost to electrons. Generally, because the ion interacts with the electronic cloud of a material, the ion projectile loses more energy to ionization compared to phonons. The opposite holds true for neutron irradiation with a lot of phonons being generated.

A final point was made on the major importance of the stored energy that could be released dramatically due to the Wigner effect. The Wigner effect is largely material dependent since in some materials, the defects are frozen up to a certain temperature like for graphite [2] causing a dramatic and sudden release of energy if the temperature is exceeded. For some other materials, the mobility of defects is increased gradually with the temperature without dramatic jumps. The stored energy is probably the most important energy loss to consider and study in depth. Essentially, I considered in my work only the distortion energy around interstitials but distortion is also present around vacancies but the literature was quite poor on these topics so I had to make assumptions to be able to develop and finalize my codes. The values present in the TRIM database for crucial parameters like E_b and E_d are general and do not necessarily reflect the reality. The structure at the atomic level is not studied and is not taken into account by TRIM which is one of its major limitation compared to molecular dynamics codes. TRIM could perhaps perform a better analysis if we provide it with more accurate values of E_b and E_d.

My master thesis relies on an extensive use of four different codes than treat mono-atomic and bi-atomic target material generically. They can indeed be adapted for any wanted ion irradiation, neutron energy spectrum, the property of the target material, whether it is mono-atomic (like pure iron) or bi-atomic (like oxides but not exclusively). The codes could even be extended in order to enter any given number of atom type of the target if we want to study the influence of some alloying elements.

Many possible improvements and deepenings of my work could be done in addition to the user interface development. The most obvious one is the consider the mobility of defects after their generation in order to implement the recovery processes and their time dependence. By this formulation, I mean that the time scale of a pressure vessel structure lifetime (40 years) is much larger than the ion irradiation time calculated. The recovery processes may dynamically take place inside a pressure vessel whereas the time scale is too short for ion irradiation. If such a molecular dynamic code exists and is available, it could be coupled to my meutron code fruitfully. That molecular code require a huge capacity of calculation, especially for slab of material as thick as 30cm for a so long period of time. All these parameters could be reflected upon and may also constitute an excellent complement to my work.

It is also worth mentioning that I found out the existence of a code called PySrim that is a Python code developed by C. Ostrouchov [11] for the automation of the use of TRIM. This finalized code enables an easy-to-use programming language to run a multitude of TRIM simulations and plotting through the Python interface. It however does not contain any neutron code but it may be inserted just before the use of the PySrim package with the necessary adaptation. To my knowledge, there is no such thing as a unified code containing both a neutron code and a molecular dynamic code for the simulation of the damage creation with a clear user-interface. I tried on my level to build such a code but I hope for my current work to only be a first step in that direction. TRIM is a renown code but lacks of considering essential phenomenons like the influence of the material structure. For a finalized version, it may be better to use a more advanced code for simulating the ion interaction with the matter than TRIM. My neutron code is statistical and considers an isotropic non-collimated flux of neutron from a population that follows a Maxwell-Boltzmann distribution. These parameters may not always be valid and a user interface for the neutron code itself may be useful for the user to define by himself the type of neutron irradiation he wants.

10 Acknowledgements

I am most grateful towards Mr Marco Beghi, my master thesis supervisor, for generously accompanying me in the development of my work. He provided me lots of useful and interesting information. He always took the necessary time during our long conversations to explain and discuss thoroughly the topics of discussion. He made my return in France possible due to the sanitary crisis and was also able to adapt and follow my work remotely.

I also sincerely thank my family for supporting me during these fluctuating times. They provided a stable environment in which I could work efficiently.

I address my last gratitude to the institution of Politecnico di Milano for welcoming me in their campuses for two years. It has been an enriching experience despite the sudden sanitary crisis. I learnt a lot in terms of engineering and Italian culture, with whom I always had an affinity.

11 References

[1] C. Abromeit. “Aspects of simulation of neutron damage by ion irradiation”. In: *Journal of Nuclear Materials* 216 (Oct 1994), pp. 78–96.

[2] Marco Beghi. “Nuclear physics”. In: *Nuclear Engineering lectures at Politecnico di Milano* (2020).

[3] F. García Ferré et al. “Extreme ion irradiation of oxide nanoceramics: influence of the irradiation spectrum”. In: *Acta Materialia*. 143rd ser. (2017).

[4] F.Garcia Ferré et al. “Radiation endurance in Al2O3 nanoceramics”. In: *Nature*. 33478th ser. (Dec 2016).

[5] Areva NP GmbH. “Nuclear Power Plant Borssele Reactor Pressure Vessel Safety Assessment”. In: NTCM-G0549 (2010).

[6] Archie Harms. *Chapter 3 : Neutron Physics of "An introduction to the CANDU Nuclear Energy Conversion System"*. McMaster University, 1972.

[7] X. J. Liu et al. “Correlation and size dependence of the lattice strain, binding energy, elastic modulus, and thermal stability for Au and Ag nanostructures”. In: *Journal of Applied Physics, 074319* 109 (2011).

[8] Nuclear-power.com. *Macroscopic Cross-Section*. URL: https://www.nuclear-power.com/nuclear-power/reactor-physics/nuclear-engineering-fundamentals/neutron-nuclear-reactions/macroscopic-cross-section/.

[9] Nuclear-power.com. *Neutron Cross-Section*. URL: https://www.nuclear-power.com/neutron-cross-section/.

[10] P. Olsson, C.S. Becquart, and C. Domain. “Ab initio threshold displacement energies in iron”. In: *Materials Research Letters* 4 (2016), pp. 219–225.

[11] Christopher Ostrouchov. *PySrim*. URL: https://pypi.org/project/pysrim/.

[12] P Rinard. *“Neutron Interaction with Matter,” in Passive Nondestructive Assay of Nuclear Materials*. U.S. Nuclear Regulatory Commission, NUREG/CR-5550. ed. by D. Reilly et al., March 1991.

[13] Hiroyuki Sakaida, Naoto Sekimura, and Shiori Ishino. “In-situ observation of cascade damage in nickel and copper under heavy ion irradiation”. In: *Journal of Nuclear Materials*. 179th ser. (1991), pp. 928–930.

[14] G. w. Wheeler et al. *"THE BROOKHAVEN 200-MEV PROTON LINEAR ACCELERATOR"*. ed. by Brookhaven National Laboratory, Upton, New York 11973, U.S.A., 1979, pp. 1–156.

[15] James F. Ziegler, Jochen P. Biersack, and Matthias D. Ziegler. *SRIM : The Stopping and Range of Ions in Matter*. 2008.

12 Appendix

In this appendix, I will partially describe and include some of the codes written in Python I created and used for the completion of my master thesis.

In total, more than ten codes were written from the beginning but some revealed useless up to a point for my purpose. Eventually, I structured the algorithms in order to make four different codes. Two among them concern the neutron irradiation with one for mono-atomic target and the other for bi-atomic targets. The other two concern the ion irradiation with the same distinction between mono-atomic and bi-atomic targets.

12.1 The ion code for mono-atomic targets

The first one to be presented is the shortest and simplest ion code for mono-atomic targets. Its purpose is to print the oscillating graph describing the Average Displacement per Ion and write a text file containing information about the energy loss distribution.

The first part from line 1 to 73 contains the importation of useful packages and definition of mostly basic functions necessary to the upcoming code. Then up to the line 143 are the definition of the physical properties to be used with choice to make. Here a user interface development could be more practical than a manual modification in the source code. Then down to the line 205 is the importation of the electronic stopping power from tables already generated by TRIM beforehand like in the figure 14. Then a big chunk of the code up to the line 345 is the correct writing of the TRIM.IN and TRIM.DAT files for the actual irradiation under study. The line 349 starts the execution of TRIM and the following line impose a pause so tthat TRIM has time to run completely before the further execution of the code. Then the rest of the code is a lecture and a disentanglement of the COLLISION.txt file in order to determine the wanted data. From line 515 to the line 520 is the plotting of the oscillating graph. From line 253 to the end is the generation of a very simple text file storing the data concerning the energy loss distribution.

```
 1 import os
 2 import matplotlib.pyplot as plt
 3 import numpy as np
 4 import scipy.integrate as integrate
 5 import time
 6 import random as rd
 7 from mpl_toolkits import mplot3d
 8 from mpl_toolkits.mplot3d import axes3d
 9 from matplotlib import cm
10 from srim import TRIM, Ion, Layer, Target
11 from srim.output import Results
12 from scipy.interpolate import interp1d
13
14 def TakeFromLine(listofline,beg,end): # return the float contained in
       the listofline between beg and end (NOT included).
15     list=[listofline[i] for i in range(beg-1,end-1)]
16     content=''
17     for l in list:
        content += l
    return float(content.replace(',','.'))

def EnergyAfterSPSegment(Sfunction,E1,L,n):
    E=E1
    for i in range(n):
        E -=  Sfunction(E)*L/n
    return E

def decimal_round(x):
    if x==0:
        return 0
    if  np.int(np.log10(np.abs(x))) >= 5:
        return 0
    elif np.int(np.log10(np.abs(x))) >= 1 and np.int(np.log10(np.abs(x)
        )) <= 4:
        return 1
    else:
        return 5

def ajout(x):
    if x==1 or x==0:
        return 0
    elif x==-1:
        return -1

def meanlist(list):
    if len(list)==0:
        return 0
    return sum(list)/len(list)

def ecart_type(list):
    if len(list)==0:
        return 0
    mean=meanlist(list)
    return np.sqrt(meanlist([(l-mean)**2 for l in list]))

def absmaxlist(list):
    max=abs(list[0])
    for r in list:
        if abs(r)>max:
            max=abs(r)
    return max

def minlist(list):
    min=list[0]
    for e in list:
        if e<min:
            min=e
    return min

def countlist(list,element):
    n=0
    for e in list:
        if e==element:
            n+=1
    return n

```

```
Nsegment=10
alline=24
RN = 8e-6 # Radius of a neutron (in A)
Mneutron = 1.675e-24 # in g
NAvogadro = 6.022e23  #in /mol
ReferenceArea=1 # cm square
SumDisplacements=4.2e-21 #to take from the neutron code output
NumberOfIons=700
XmaxNoScientific=3e9 #in A
Xmax = '{:.0e}'.format(XmaxNoScientific)

## TARGET MATERIAL

ElementName = 'Fe'
Name = 'IRON'
CorrectName='Iron'
stoiechiometry=1
AtomicNumber=26
VolumicMass=7.87 # in g per cubic cm
MolarMass=55.845  # in g/mol-1
AtomicMassOfTarget= MolarMass/NAvogadro # mass in g of a nucleus
NumberDensity=VolumicMass/AtomicMassOfTarget
RM=1.3*1e-5*((2*AtomicNumber)**(1/3)) # approximation in A RM=r0*
    numerofnucleon**(1/3) with r0 =1.3 fm=1.3e(-5)A    = 4.85 fm
DisplacementEnergy = 40 #eV
LatticeBindingEnergy = 3
FormationEnergyOfInterstitial=3*LatticeBindingEnergy


time1=time.time()


# ElementName = 'Zn'
# Name = 'Zinc'
# stoiechiometry=1
# AtomicNumber=30
# VolumicMass=7.14 # in g per cubic cm
# MolarMass=65.39  # in g/mol-1
# AtomicMassOfTarget= MolarMass/NAvogadro # mass in g of a nucleus
# RM=1.3*1e-5*((2*AtomicNumber)**(1/3)) # approximation in A RM=r0*
    numerofnucleon**(1/3) with r0 =1.3 fm=1.3e(-5)A    = 4.85 fm
# DisplacementEnergy = 25 #eV
# LatticeBindingEnergy = 3
# FormationEnergyOfInterstitial=3*LatticeBindingEnergy

## Ion information:

IonElement = 'Ni'
CompleteNameIon='Nickel'
IonEnergy = 4e6 #eV
IonAtomicNumber=28
AtomicMassIon=58.7 #in amu
ReferenceDepth=1000 #A

# Ion information:

# IonElement = 'Au'
# CompleteNameIon='Gold'
```

```python
# IonEnergy = 12e6 #eV
# IonAtomicNumber=79
# AtomicMassIon=197 #in amu
# ReferenceDepth=500 #A

# Ion information:

# IonElement = 'H'
# CompleteNameIon='Hydrogen'
# IonEnergy = 200e6 #eV
# IonAtomicNumber=1
# AtomicMassIon=1.008 #in amu
# ReferenceDepth=2.5e9 #A

eV=1
keV=0
MeV=0
ELIST=[]
SELIST=[]
SNLIST=[]

IRON_IN_IRON=open("C:/Users/Julien Bouix/Desktop/SRIM-2013/SRIM Outputs/{} in {}.txt".format(Name,Name),"r")

IRON_IN_IRON_lines=IRON_IN_IRON.readlines()
IRON_IN_IRON.close()

NICKEL_IN_IRON=open("C:/Users/Julien Bouix/Desktop/SRIM-2013/SRIM Outputs/{} in {}.txt".format(CompleteNameIon,Name),"r")

NICKEL_IN_IRON_lines=NICKEL_IN_IRON.readlines()
NICKEL_IN_IRON.close()


while IRON_IN_IRON_lines[alline][0]!='-':

    if TakeFromLine(IRON_IN_IRON_lines[alline],1,8)<TakeFromLine(IRON_IN_IRON_lines[alline-1],1,8) and keV==1:
        keV=0
        MeV=1
    if TakeFromLine(IRON_IN_IRON_lines[alline],1,8) < TakeFromLine(IRON_IN_IRON_lines[alline-1],1,8) and keV==0 and eV==1:
        keV=1
        eV=0

    ELIST.append((MeV*1e6+keV*1e3+eV)*TakeFromLine(IRON_IN_IRON_lines[alline],1,8))
    SELIST.append(TakeFromLine(IRON_IN_IRON_lines[alline],15,24))
    SNLIST.append(TakeFromLine(IRON_IN_IRON_lines[alline],26,35))

    alline+=1

SEfunctionIroninIron = interp1d(ELIST,SELIST,fill_value="extrapolate")
SNfunctionIroninIron = interp1d(ELIST,SNLIST,fill_value="extrapolate")
alline=24
eV=1
keV=0
MeV=0
```

```
ELIST=[]
SELIST=[]
SNLIST=[]

while NICKEL_IN_IRON_lines[alline][0]!='-':

    if TakeFromLine(NICKEL_IN_IRON_lines[alline],1,8)<TakeFromLine(
        NICKEL_IN_IRON_lines[alline-1],1,8) and keV==1:
        keV=0
        MeV=1
    if TakeFromLine(NICKEL_IN_IRON_lines[alline],1,8) < TakeFromLine(
        NICKEL_IN_IRON_lines[alline-1],1,8) and keV==0 and eV==1:
        keV=1
        eV=0

    ELIST.append((MeV*1e6+keV*1e3+eV)*TakeFromLine(NICKEL_IN_IRON_lines
        [alline],1,8))
    SELIST.append(TakeFromLine(NICKEL_IN_IRON_lines[alline],15,24))
    SNLIST.append(TakeFromLine(NICKEL_IN_IRON_lines[alline],26,35))

    alline+=1

SEfunctionNickelinIron = interp1d(ELIST,SELIST,fill_value="extrapolate"
    )
SNfunctionNickelinIron = interp1d(ELIST,SNLIST,fill_value="extrapolate"
    )

CurrentDisplacements=0
ListOfVacancies=[]
IonCount=0
ListOfDPAPerParticle=[]
ListOfVacancies=[]
IONIZATIONListPRIMARY=[]
BINDINGListPRIMARY=[]
PHONONListPRIMARY=[]


IONIZATIONListNONPRIMARY=[]
BINDINGListNONPRIMARY=[]
PHONONListNONPRIMARY=[]

## EDITING OF THE TRIM.IN FILE

    # We select the type of calculation we want :
    # 1: Ion distribution and Quick Calculation of Damade (Kinchin-
        Pease) (TYPE A COLLISION.txt file)
    # 2: Detailed Calculation with full Damage Cascade (TYPE B
        COLLISION.txt file)
    # 3: Monolayer Collision Steps / Surface Sputtering (TYPE B
        COLLISION.txt file) (monolayer means that the ion react with
        each monolayer it gets through))
    # From 4 to the end, it uses TRIM.DAT :
    # 4: Ion with specific energy/ angle/ depth (quick Damage)
    # 5: Ion with specific energy/ angle/ depth (full cascades)
    # 6: Recoil cascades with neutrons (full cascades)
    # 7: Recoil cascades and monolayer steps (full cascades) (TYPE B
        COLLISION.txt file)
    # 8: Recoil cascades with neutrons (quick damage : Kinchin Pease)
# FOR NOW I TAKE 8


TRIMIN = open("TRIM.IN","r")
TRIMINNEW = open("TRIMN.IN","w")
TRIMINLINES = TRIMIN.readlines()
line=0


while TRIMINLINES[line][0] != 'C' :
    TRIMINNEW.write(TRIMINLINES[line])
    line+=1
TRIMINNEW.write(TRIMINLINES[line])
line+=1
TRIMINNEW.write('                              6                              0          0\n') # FOR NOW I TAKE 6
line+=1

while TRIMINLINES[line][0] != 'T' :
    TRIMINNEW.write(TRIMINLINES[line])
    line+=1
TRIMINNEW.write(TRIMINLINES[line])
line+=1
TRIMINNEW.write('"H (10) into Layer 1                          "          1                        1\n')
line+=1
TRIMINNEW.write(TRIMINLINES[line])
line+=1
TRIMINNEW.write('          0                              0              0\n')
line+=1

while TRIMINLINES[line][0] != 'A' :
    TRIMINNEW.write(TRIMINLINES[line])
    line+=1
TRIMINNEW.write('Atom 1 = {} ='.format(ElementName)+' '*(7-(len(Name)-2))+'{}'.format(AtomicNumber)+' '*(3-np.int(np.log10(AtomicNumber)))+'{}\n'.format(MolarMass))
line+=1

lenTRIMIN=len(TRIMINLINES)
if lenTRIMIN>28:
    beforecompound=1
    for i in range(lenTRIMIN-28):
        line+=1

TRIMINNEW.write('Layer   Layer Name /                    Width Density     {}({})\n'.format(Name,AtomicNumber))
line+=1
TRIMINNEW.write('Numb.   Description                     (Ang) (g/cm3)     Stoich\n')
line+=1
TRIMINNEW.write(' 1        "{}"'.format(Name)+' '*(21-(len(Name)-4))+Xmax+' '*(2-(len(str(Xmax))-7))+'{}'.format(VolumicMass)+' '*(7-(len(str(VolumicMass))-6))+'1\n')

TRIMINNEW.write('0  Target layer phases (0=Solid , 1=Gas)\n0 \nTarget Compound Corrections (Bragg)\n 1\nIndividual target atom displacement energies (eV)\n      {}\nIndividual target atom
lattice binding energies (eV)\n        3\nIndividual target atom
    surface binding energies (eV)\n    4.34\nStopping Power Version
    (1=2011, 0=2011)\n 0'.format(DisplacementEnergy))

TRIMIN.close()
TRIMINNEW.close()
os.remove('TRIM.IN')
os.rename('TRIMN.IN', 'TRIM.IN')


for nbion in range(NumberOfIons):

    alphaangle=(rd.random()-0.5)*np.pi*2*5/180 # 5   deviation possible
        =5*np.pi/180
    betaangle=rd.random()*2*np.pi
    axe=[np.cos(alphaangle),np.sin(alphaangle)*np.sin(betaangle),np.sin
        (alphaangle)*np.cos(betaangle)]

## We edit the TRIM.DAT for creating a small angle

    TRIMDAT = open("TRIM.DAT","r")
    TRIMDATNEW = open("TRIMN.DAT","w")
    TRIMLINES = TRIMDAT.readlines()
    line=0


    while TRIMLINES[line][0] != '   ':
        TRIMDATNEW.write(TRIMLINES[line])
        line+=1
    TRIMDATNEW.write('              Recoils from {}eV {} ions in {}({})
                                 \n'.format('{:.1e}'.format(IonEnergy),
        IonElement,Name,Xmax))
    line+=1
    TRIMDATNEW.write(TRIMLINES[line])
    line+=1
    TRIMDATNEW.write(TRIMLINES[line])
    line+=1

    TRIMDATNEW.write('1        {}      {}      0          0         0     {}     {}
          {}'.format(IonAtomicNumber,'{:.1e}'.format(IonEnergy),round(
        axe[0],4),round(axe[1],4),round(axe[2],4)))

    TRIMDAT.close()
    TRIMDATNEW.close()
    os.remove("TRIM.DAT")
    os.rename("TRIMN.DAT","TRIM.DAT")

## We edit the TRIM.IN for Random Seed Number

    TRIMIN = open("TRIM.IN","r")
    TRIMINNEW = open("TRIMI.IN","w")
    TRIMINLINES = TRIMIN.readlines()
    line=0

    while TRIMINLINES[line][0] != 'I':
        TRIMINNEW.write(TRIMINLINES[line])
        line+=1
    TRIMINNEW.write(TRIMINLINES[line])
    line+=1
    TRIMINNEW.write('       {}     {}           {}          0          {}          1      10000\n'.format(IonAtomicNumber,AtomicMassIon,int(IonEnergy*10**(-3)),1))
    line+=1
    TRIMINNEW.write('Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7=Neutrons), Random Number Seed, Reminders\n')
    line+=1
    TRIMINNEW.write('                              5                              {}          0\n'.format(int(1000000*rd.random())))

    for lin in range(line+1,len(TRIMINLINES)):              #the rest of the file to be copied unaltered
        if TRIMINLINES[lin-1][0]=='P':
            TRIMINNEW.write('          0                              0          40000\n') # the ion ranges deep to 4 um
        else:
            TRIMINNEW.write(TRIMINLINES[lin])

    TRIMIN.close()
    TRIMINNEW.close()
    os.remove('TRIM.IN')
    os.rename('TRIMI.IN','TRIM.IN')

## We execute

    os.startfile('C:/Users/Julien Bouix/Desktop/SRIM-2013/TRIM.exe')
    time.sleep(25) #25 for Ni #46 for A

    COLLISION = open("C:/Users/Julien Bouix/Desktop/SRIM-2013/SRIM Outputs/COLLISION.txt","r")
    COLLISIONLINES=COLLISION.readlines()
    COLLISION.close()
    LengthCOLLISION=len(COLLISIONLINES)
    PrimaryPhonon=[]
    ## Here, we col lect the informations needed for calculating the stopping power

    Xpkalist=[]
    Ypkalist=[]
    Zpkalist=[]

    IonEnergyBeforeCollision=[]

    NbOfPKAn=0
    for i in range(LengthCOLLISION):
        if COLLISIONLINES[i][0] ==' ':
            NbOfPKAn +=1

    NbOfPKAn=int(NbOfPKAn/2)

    EnergyTransmittedToPKAn=[]

    ListOfIONIZATION=[]
    ListOfIONIZATIONPKA=[]
    ListOfBINDING=[]
    ListOfBINDINGexceptprimary=[]
    ListOfPHONON=[]
    Nreplacement=[0]
    Nvac=[0]
    currentline=23

    for nbpkan in range(NbOfPKAn):

        IONIZATIONRecoil=[]
        IONIZATIONRecoilPKA=[]
        NUCLEARRecoil=[]
        BINDINGRecoil=0
        PHONONRecoil=0
        Xlist=[]
        Ylist=[]
        Zlist=[]
        Nreplacement.append(0)
        Nvac.append(0)
        RecoilEnergyList=[]
        currentline +=2

        if currentline==LengthCOLLISION:
            break

        while COLLISIONLINES[currentline][0]!=' ' and currentline<=
            LengthCOLLISION-2: #if no recoil cascade for this pka
            currentline+=1

        if currentline==LengthCOLLISION-1:
            break

        # start of a new cascade
        linecontent=COLLISIONLINES[currentline]

        try:
            linecontent=COLLISIONLINES[currentline]
            Xpkalist.append(TakeFromLine(linecontent,18,28))
            Ypkalist.append(TakeFromLine(linecontent,29,39))
            Zpkalist.append(TakeFromLine(linecontent,40,50))
            EnergyTransmittedToPKAn.append(TakeFromLine(linecontent
                ,64,74))
        except:
            currentline+=-2
            linecontent=COLLISIONLINES[currentline]
            Xpkalist.append(TakeFromLine(linecontent,18,28))
            Ypkalist.append(TakeFromLine(linecontent,29,39))
            Zpkalist.append(TakeFromLine(linecontent,40,50))
            EnergyTransmittedToPKAn.append(TakeFromLine(linecontent
                ,64,74))

        currentline+=3

        while COLLISIONLINES[currentline][0] ==' ':
            linecontent=COLLISIONLINES[currentline]
            RecoilEnergyList.append(TakeFromLine(linecontent,15,25))
            Xlist.append(TakeFromLine(linecontent,26,35))
            Ylist.append(TakeFromLine(linecontent,36,46))
            Zlist.append(TakeFromLine(linecontent,47,57))
            Nvac[-1]+=float(COLLISIONLINES[currentline][58])
            Nreplacement[-1]+=float(COLLISIONLINES[currentline][63])
            currentline+=1

        if COLLISIONLINES[currentline][0] ==' ':
            currentline+=1

        ##ONLY BY READING AND DEDUCING....

        if len(RecoilEnergyList)==1:
            BINDINGRecoil    += FormationEnergyOfInterstitial
            IONIZATIONRecoil.append(0)
            PrimaryPhonon.append(RecoilEnergyList[0]-FormationEnergyOfInterstitial)


        else:
            lengthLIST=len(Xlist)

            ## BINDINGS

            ListOfBINDINGexceptprimary.append(int(Nvac[-1]-1-Nreplacement[-1])*FormationEnergyOfInterstitial)

            maxdistance = max([np.sqrt((Xlist[i]-Xlist[0])**2+(Ylist[i]-Ylist[0])**2+(Zlist[i]-Zlist[0])**2) for i in range(len(Xlist))])

            ListofSKAIndexes=[1]
            ListOfInitialKineticEnergiesSKA=[RecoilEnergyList[1]]

            for i in range(2,len(Xlist)):
                if np.sqrt((Xlist[i]-Xlist[i-1])**2+(Ylist[i]-Ylist[i-1])**2+(Zlist[i]-Zlist[i-1])**2) > maxdistance/4.5:
                    ListofSKAIndexes.append(i)
                    ListOfInitialKineticEnergiesSKA.append(RecoilEnergyList[i])

            ## CALCULATION OF PKA CASCADE

            sumenergy=0
            ListofSKAIndexes.reverse()

            for i in ListofSKAIndexes:

                IONIZATIONRecoilPKA.append(RecoilEnergyList[0]-sumenergy-EnergyAfterSPSegment(SEfunctionIroninIron,RecoilEnergyList[0]-sumenergy,np.sqrt((Xlist[i]-Xlist[i-1])**2+(Ylist[i]-Ylist[i-1])**2+(Zlist[i]-Zlist[i-1])**2),Nsegment))

            IONIZATIONRecoilPKA2=sum(IONIZATIONRecoilPKA)
            ListOfIONIZATIONPKA.append(IONIZATIONRecoilPKA2)
        ListOfIONIZATION.append(sum(IONIZATIONRecoil))
        ListOfBINDING.append(BINDINGRecoil)

    PrimaryIonization = [IonEnergy - EnergyAfterSPSegment(SEfunctionNickelinIron,IonEnergy,np.sqrt(Xpkalist[0]**2+
        Ypkalist[0]**2+Zpkalist[0]**2),Nsegment)]


    sumenergyprimary=PrimaryIonization[0]+EnergyTransmittedToPKAn[0]
    for i in range(NbOfPKAn-1):

        PrimaryIonization.append(IonEnergy-sumenergyprimary-EnergyAfterSPSegment(SEfunctionNickelinIron,IonEnergy-sumenergyprimary,np.sqrt((Xpkalist[i]-Xpkalist[i+1])**2+(Ypkalist[i]-Ypkalist[i+1])**2+(Zpkalist[i]-Zpkalist[i+1])**2),Nsegment))


        sumenergyprimary+=PrimaryIonization[-1]+EnergyTransmittedToPKAn[i+1]

    IONIZATIONexceptprimary=sum(ListOfIONIZATION)+sum(ListOfIONIZATIONPKA)


    IONIZATIONListPRIMARY.append(100*sum(PrimaryIonization)/IonEnergy)
    BINDINGListPRIMARY.append(100*NbOfPKAn*FormationEnergyOfInterstitial/IonEnergy)
    PHONONListPRIMARY.append(100*sum(PrimaryPhonon)/IonEnergy)

    totalprimary= sum(PrimaryIonization)/IonEnergy + NbOfPKAn*FormationEnergyOfInterstitial/IonEnergy + sum(PrimaryPhonon)/IonEnergy

    IONIZATIONListNONPRIMARY.append(100*IONIZATIONexceptprimary/IonEnergy)
    BINDINGListNONPRIMARY.append(100*sum(ListOfBINDINGexceptprimary)/IonEnergy)
    PHONONListNONPRIMARY.append(100*(1-totalprimary-IONIZATIONexceptprimary/IonEnergy-sum(ListOfBINDINGexceptprimary)/IonEnergy))


    ##  To know what is the average vacancy induced by each ion.

    listvac=0
    for i in range(NbOfPKAn):

        if Xpkalist[i] < ReferenceDepth:

            listvac += Nvac[i]
    ListOfVacancies.append(listvac)

    ListOfDPAPerParticle.append(sum(ListOfVacancies)/((nbion+1)*ReferenceDepth*10**(-8)*ReferenceArea*NumberDensity))

plt.ion()
plt.plot([i for i in range(1,len(ListOfDPAPerParticle)+1)],ListOfDPAPerParticle)
plt.title('The average dpa per ion projectile in the reference volume')
plt.ylabel('average dpa per ion projectile')
plt.xlabel('ion index')
plt.show()
```

```
resultfile=open("C:/Users/Julien Bouix/Desktop/SRIM-2013/Result Of the
    ion Code.txt",'w')
resultfile.write("NOW PRIMARY ION\n")
resultfile.write('{}\n'.format(meanlist(BINDINGListPRIMARY)))
resultfile.write('{}\n'.format(meanlist(IONIZATIONListPRIMARY)))
resultfile.write('{}\n'.format(meanlist(PHONONListPRIMARY)))
resultfile.write('\n\n')

resultfile.write("NOW Recoils\n")
resultfile.write('{}\n'.format(meanlist(BINDINGListNONPRIMARY)))
resultfile.write('{}\n'.format(meanlist(IONIZATIONListNONPRIMARY)))
resultfile.write('{}\n'.format(meanlist(PHONONListNONPRIMARY)))
resultfile.write('\n\n')

resultfile.write("NOW TOTAL\n")
resultfile.write('{}\n'.format(meanlist(BINDINGListNONPRIMARY)+meanlist
    (BINDINGListPRIMARY)))
resultfile.write('{}\n'.format(meanlist(IONIZATIONListNONPRIMARY)+
    meanlist(IONIZATIONListPRIMARY)))
resultfile.write('{}\n'.format(meanlist(PHONONListNONPRIMARY)+meanlist(
    PHONONListPRIMARY)))

resultfile.close()

time2=time.time()

print(time2-time1)
```

12.2 The neutron code for bi-atomic targets

For the last pages of my master thesis report, I would like to address the most difficult code among the four previously introduced.

The neutron code for bi-atomic targets contains both the neutron code developed in the early stage of my work and also the adaptation of the analyse from a mono-atomic target to a bi-atomic one. Step by step, I will take the time to enumerate what my code is doing.

1. From line 1 to 14, the code imports all necessary packages.

2. From line 16 to 166, minor functions are defined like the ones from the ion code for mono-atomic target but also complex functions for the neutron code like the major COLLISION function defined at the line 39.

3. From line 169 to the line 290 are the physical definition of the constants used below. Here again, a user interface would make it much more convenient for a regular use. The line 185 is a very important one since its calculate the energy profile of a population of neutrons following a Maxwell-Boltzmann distribution.

4. Then down to the line 360 is the same importation of the various electronic stopping power functions from pre-generated files. This time the code is

adapted for bi-atomic targets. This means files for X and O into X oxides separately. In the case of ion irradiation (not the case here), we also need to import teh electronic stopping power files of the considered ion into the oxide.

5. From the line 361 to the line 423 is the generation of the corrected TRIM.IN file.

6. Then the neutron code is inserted from the line 431 to the line 621. It will be detailed further down on its own.

7. Down to the line 725 is the editing of the TRIM.DAT file that contains the data generated by the neutron code. That data is information on the atom knocked-off by the neutron like its atomic number, its energy, its initial lattice position and its direction after impact.

8. From the line 726 to the line 934 is the reading of the output file adapted for bi-atomic targets that is quite complex so it will not be detailed here. To keep it simple, it acts accordingly to the atomic number read each line of the COLLISION.txt file. This information is contained in the list ALORO that is an trace of the beginning of the code development when only Al_2O_3 was studied (motivated by extending the work of [4] introduced by my master thesis supervisor Marco Beghi). ALORO stands for "Al or O" but works the same for any X oxides.

9. The rest of the file is a generation of a more complex text file that contains information about the neutron code and also the energy loss distribution.

Two particular aspects of the code deserve to be further developed to the reader; the COLLISION function and the neutron code itself.

1. Starting with the COLLISION function that is written from the line 39 to 79. Its purpose is to calculate values related to the figure 5 and the collision between two spherical particles of radii and mass R_1, m_1, R_2 and m_2 with the first particle at energy E_1 and the second immobile. The collision takes place with an impact parameter b and the incoming particle axis direction is also an input of the function ("axein"). Then it calculates the transferred energy according to the functions defined in the physical framework. The most difficult part was to determine the axeout of the incoming particle after impact. It needed a definition of an other vector $\vec{b}$ defined perpendicularly to the axein. Then, in order to grasp its location in relation to the classical (X,Y,Z) space definition, a second order equation has to be solved. Many exceptions need to be considered. The process to find out this process was relatively tricky. Eventually, the COLLISION function returns the unit vector direction ("axeout") of the initial particle after impact and the unit vector direction ("recoilaxe") of the second particle along with the final energies of both particles and useful angles. The vectors are defined in the (X,Y,Z) reference system.

2. The neutron code starts at the line 431 and ends at the line 621. The first "for" loop runs over the total number of neutron specified at the line 178. All necessary variables are defined at the right time and the code runs while the

condition defined at the line 478 holds true. This condition states that while the neutron simultaneously has enough energy to displace an atom, is not backscattered and not transmitted, the calculation continues. Then, whether the atom with which the neutron collides, the line 540 or the line 542 calls the COLLISION function defined above. the ancient axeout becomes the new axein and the cycle continues until one condition is broken. Then, the information are updated and what is required for the TRIM.DAT file is stored in the TRIMDAT_INPUT file. These information are the position of the current collision, the axe of the neutron velocity and the transferred energy to the atoms. The impact parameter are calculated probabilistically in the lines 493 and 494. The condition written in the line 561 is also essential for the neutron code in the bi-atomic target case. It determines if the next atom to undergo a collision with the neutron is X or O. This determination is based on a comparison of the exponential distribution of the macroscopic cross section between both X and O.

```
1  import os
2  import time
3  import numpy as np
4  import random as rd
5  import math
6  import time
7  import scipy.integrate as integrate
8  import matplotlib.pyplot as plt
9  from mpl_toolkits import mplot3d
10 from mpl_toolkits.mplot3d import axes3d
11 from matplotlib import cm
12 from scipy.interpolate import interp1d
13 from scipy.stats import expon
14 from scipy.stats import uniform
15
16 ## ALL USEFUL FUNCTIONS USED IN THIS CODE.
17
18 def SigmaMicro(S1,S2,E1,E2,E):    #Returns the microscopic cross section
       for neutrons of energy E with a linear approximation between 0.025
     eV and 1MeV
19     if E>E2:
20         return S2  # after E2 = 1 MeV, I consider the cross-section
              constant.
21     else:
22         return S1+(S2-S1)/(E2-E1)*(E-E1)
23
24 def SigmaMacro(Vmass,MolarMass,stoiech,S1a,S2a,S1b,S2b,E1,E2,E): #
     Returns the macroscopic cross section in A**(-1) for neutrons of
     energy E
25     return 10**(-32)*(Vmass*NAvogadro/MolarMass)*(stoiech*SigmaMicro(
         S1a,S2a,E1,E2,E)+(1-stoiech)*SigmaMicro(S1b,S2b,E1,E2,E))
26
27 def TransferredEnergyCoeff(b,R1,R2):
28     return 0.5*(1-np.cos(np.arccos(2*(b/(R1+R2))**2-1)))
29
30 def Angle(axis1,axis2):
31     [x1,x2,x3],[y1,y2,y3]=axis1,axis2
32     return np.arccos((x1*y1+x2*y2+x3*y3)/(np.sqrt(x1**2+x2**2+x3**2)*np
         .sqrt(y1**2+y2**2+y3**2)))
```

```

def Normalization(axe):
    [x1,x2,x3]=axe
    modulus=np.sqrt(x1**2+x2**2+x3**2)
    return[x1/modulus,x2/modulus,x3/modulus]

def COLLISION(axein,b,R1,R2,m1,m2,E1):    #COLLISION calculate a list of 4 elements :quantity of movement final for both particles and the recoil angle and scattering angle. axe is [cos(X),cos(Y),cos(Z)] all incoming. The randomness of beta angle will be calculated here.

    alpha=np.arcsin(b/(R1+R2))
    scattering_angle=np.pi-2*alpha
    E2f=E1*(m1*m2)/((m1+m2)**2)*0.5*(1-np.cos(scattering_angle))
    E1f=E1-E2f

    # Now,finding the axe out:
    [x1,x2,x3]=axein
    if x2!=0:

        ## Now we need to choose a random unit bvector that is perpendicular to axein
        # First find b1 using the only vector bnominal that is in the plane formed by (Xaxis,axein)
        beta = rd.random()*2*np.pi
        #gamma = Angle([1,0,0],axein)
        b1=np.cos(beta)*np.sqrt((1-x1**2)**2+x1**2*x2**2+x1**2*x3**2)

        ## Solve for b3 the second order equation that arises from the condition of unit vector and scalar product between bvector and axein to be null.

        b3=(-2*b1*x1*x3+2*np.sqrt(b1**2*x1**2*x3**2-(x2**2+x3**2)*(b1**2*(x1**2+x2**2)-x2**2)))/(2*(x2**2+x3**2))
        b2=np.sqrt(1-b1**2-b3**2)

    if x2==0 and x3!=0:
        b1=2*(rd.random()-0.5)
        b3=-b1*x1/x3
        b2=np.sqrt(1-b1**2-b3**2)

    if x2==0 and x3==0:
        b1=0
        beta=rd.random()*np.pi*2
        b2=np.cos(beta)
        b3=np.sin(beta)
    bvector=[b1,b2,b3]

        ##find the axeout director vector:
    axeout=Normalization([np.sin(scattering_angle)*bvector[i]+np.cos(scattering_angle)*axein[i] for i in range(3)]) # axeout is expressed in the orthogonal base (bvector,axein)

        ##find the recoilaxe (direction where the knocked atom goes):

    recoilaxe=Normalization([np.sin(alpha)*bvector[i]+np.cos(alpha)*axein[i] for i in range(3)]) #recoilaxe is expressed in the orthogonal base (bvector,axein)
```

```python
    return [axeout, recoilaxe, E1f, E2f, alpha, scattering_angle]

def parabolic_law(a,b,diff): # Returns a random value following a parabolic law between a and b with a maximum probability at its edge and a difference (defined as a ratio between the probability at its edge and its center (diff>1)
    def PL(x): # The law is alpha*x  +beta*x*gamma
        alpha=(diff-1)/(diff*((a+b)/2)**2-b*a)/(b-a-((a+b)*(b**2-a**2)/2-(b**3-a**3)/3)*(diff-1)/(diff*((a+b)/2)**2-b*a))
        beta=-alpha*(b+a)
        gamma= alpha/((diff-1)/(diff*((a+b)/2)**2-b*a))
        return alpha*x**2+beta*x+gamma
    t=rd.random()
    y=a
    while integrate.quad(PL,a,y)[0] < t:
        y+=(b-a)/200
    return min(y,b)

def LinearCrown(R): #returns a random number between 0 and R that follows a linear increasing probability distribution function.
    t= rd.random()
    y=0
    func = lambda b: 2*b/(R**2)
    while integrate.quad(func,0,y)[0] < t:
        y += R/200
    return min(R,y)

def decimal_round(x):   #CRITICAL: To use as a indicator for the python function round that round number and this function defines the number of decimals I decided to keep.
    return 0

def ajout(x):
    if x==1 or x==0:
        return 0
    elif x==-1:
        return -1

def meanlist(list):
    if len(list)==0:
        return 0
    return sum(list)/len(list)

def ecart_type(list):
    if len(list)==0:
        return 0
    mean=meanlist(list)
    return np.sqrt(meanlist([(l-mean)**2 for l in list]))

def absmaxlist(list):
    max=abs(list[0])
    for r in list:
        if abs(r)>max:
            max=abs(r)
    return max

def minlist(list):
    min=list[0]
    for e in list:
        if e<min:
            min=e
    return min

def countlist(list,element):
    n=0
    for e in list:
        if e==element:
            n+=1
    return n

def NeutronEnergyDistr(E1): # returns a value of energy that follow a maxwellian distribution and E1 if the most probable energy
    t = rd.random()
    y=0
    a=np.sqrt(E1/(Mneutron))
    def PDF_NEDnonnorma(E):
        return np.sqrt(2**3/np.pi)*E/(Mneutron*a**3)*np.exp(-E/E1)
    def PDF_NED(E):
        return PDF_NEDnonnorma(E)/integrate.quad(PDF_NEDnonnorma,0,20*E1)[0]

    while integrate.quad(PDF_NED,0,y)[0] < t:
        y += 20*E1/200
    return min(y,20*E1)

def TakeFromLine(listofline,beg,end): # return the float contained in the listofline between beg and end (NOT included).
    list=[listofline[i] for i in range(beg-1,end-1)]
    content=''
    for l in list:
        content += l
    return float(content.replace(',','.'))

def EnergyAfterSPSegment(Sfunction,E1,L,n):
    E=E1
    for i in range(n):
        E -= Sfunction(E)*L/n
    return E


UNIFIED_ATOMIC_MASS = 1.66e-24 #in g
time1= time.time()

XmaxNoScientific=3e9 #in A
Xmax = '{:.0e}'.format(XmaxNoScientific)


Nsegment=10
alline=24
NumberOfNeutrons=50  # How much neutrons we consider
RN = 8e-6 # Radius of a neutron (in A)
Mneutron = 1.675e-24 # in g
NAvogadro = 6.022e23  #in /mol
ReferenceArea=1 # cm square
ReferenceDepth1=1e8
ReferenceDepth2=30e8
```

```
ListOfNeutronEnergy=[NeutronEnergyDistr(1e6) for i in range(
    NumberOfNeutrons)] #if  1MeV is the most probable

# IndexOfSimulation=1

##for Al2O3
# VolumicMass = 3.99
# CompoundName='Al2O3'  #Called genericall XnOm, element 1 is X !!
# n=2
# m=3
# E1 = 0.025
# E2 = 1e6
# SigmaMicroScatteringONEAtE1 = 1.41 #in barn: 1b = 1e-28 m
# SigmaMicroScatteringONEAtE2 = 1.41

##for Zro2
# VolumicMass = 5.85
# CompoundName='ZrO2'  #Called genericall XnOm, element 1 is X !!
# n=1
# m=2
# E1 = 0.025
# E2 = 1e6
# SigmaMicroScatteringONEAtE1 = 5.34 #in barn: 1b = 1e-28 m
# SigmaMicroScatteringONEAtE2 = 5.34

##for Y2O3
VolumicMass = 5.03
CompoundName='Y2O3'  #Called genericall XnOm, element 1 is X !!
n=2
m=3
E1 = 0.025
E2 = 1e6
SigmaMicroScatteringONEAtE1 = 7.6 #in barn: 1b = 1e-28 m
SigmaMicroScatteringONEAtE2 = 7.6


##for HfO2
# VolumicMass = 9.68
# CompoundName='HfO2'  #Called genericall XnOm, element 1 is X !!
# n=1
# m=2
# SigmaMicroScatteringONEAtE1 = 14.07-2.56 #in barn: 1b = 1e-28 m
# SigmaMicroScatteringONEAtE2 = 5.19-0.003

#Aluminium
# ElementName1 = 'Al'
# CompleteName1='Aluminum'
# AtomicNumber1=13
# MolarMass1=26.982  # in g/mol-1
# AtomicMassOfTarget1= MolarMass1/NAvogadro # mass in g of a nucleus
# NumberDensity1=VolumicMass/AtomicMassOfTarget1
# RM1=1.3*1e-5*((2*AtomicNumber1)**(1/3)) # approximation in A RM=r0*
    numerofnucleon**(1/3) with r0 =1.3 fm=1.3e(-5)A     = 4.85 fm
# DisplacementEnergy1 = 25 #eV
# LatticeBindingEnergy1 = 3
# FormationEnergyOfInterstitial1=3*LatticeBindingEnergy1

#Zirconium
```

```
# ElementName1 = 'Zr'
# CompleteName1='Zirconium'
# AtomicNumber1=40
# MolarMass1=91.22  # in g/mol-1
# AtomicMassOfTarget1= MolarMass1/NAvogadro # mass in g of a nucleus
# NumberDensity1=VolumicMass/AtomicMassOfTarget1
# RM1=1.3*1e-5*((2*AtomicNumber1)**(1/3)) # approximation in A RM=r0*
    numerofnucleon**(1/3) with r0 =1.3 fm=1.3e(-5)A     = 4.85 fm
# DisplacementEnergy1 = 25 #eV
# LatticeBindingEnergy1 = 3
# FormationEnergyOfInterstitial1=3*LatticeBindingEnergy1

# #Yttrium
ElementName1 = 'Y'
CompleteName1='Yttrium'
AtomicNumber1=39
MolarMass1=88.906  # in g/mol-1
AtomicMassOfTarget1= MolarMass1/NAvogadro # mass in g of a nucleus
NumberDensity1=VolumicMass/AtomicMassOfTarget1
RM1=1.3*1e-5*((2*AtomicNumber1)**(1/3)) # approximation in A RM=r0*
    numerofnucleon**(1/3) with r0 =1.3 fm=1.3e(-5)A     = 4.85 fm
DisplacementEnergy1 = 25 #eV
LatticeBindingEnergy1 = 3
FormationEnergyOfInterstitial1=3*LatticeBindingEnergy1

#Hafnium
# ElementName1 = 'Hf'
# CompleteName1='Hafnium'
# AtomicNumber1=72
# MolarMass1=178.49  # in g/mol-1
# AtomicMassOfTarget1= MolarMass1/NAvogadro # mass in g of a nucleus
# RM1=1.3*1e-5*((2*AtomicNumber1)**(1/3)) # approximation in A RM=r0*
    numerofnucleon**(1/3) with r0 =1.3 fm=1.3e(-5)A     = 4.85 fm
# DisplacementEnergy1 = 25 #eV
# LatticeBindingEnergy1 = 3
# FormationEnergyOfInterstitial1=3*LatticeBindingEnergy1

#Oxygen
ElementName2 = 'O'
AtomicNumber2=8
MolarMass2=15.999  # in g/mol-1
AtomicMassOfTarget2= MolarMass2/NAvogadro # mass in g of a nucleus
NumberDensity2=VolumicMass/AtomicMassOfTarget2
RM2=1.3*1e-5*((2*AtomicNumber2)**(1/3)) # approximation in A RM=r0*
    numerofnucleon**(1/3) with r0 =1.3 fm=1.3e(-5)A     = 4.85 fm
DisplacementEnergy2 = 28 #eV
LatticeBindingEnergy2 = 3
FormationEnergyOfInterstitial2=3*LatticeBindingEnergy2
SigmaMicroScatteringTWOAtE1 = 3.83 #in barn: 1b = 1e-28 m
SigmaMicroScatteringTWOAtE2 = 3.83

CompoundMolarMass= n*MolarMass1+m*MolarMass2
stoiechiometry=n/(n+m)
NumberDensity=VolumicMass/(n*AtomicMassOfTarget1+m*AtomicMassOfTarget2)
    *(n+m)

## for OXYGEN first:

```

```
if len(ElementName1)==1:
    OXYGEN_IN_AL2O3=open("C:/Users/Julien Bouix/Desktop/SRIM-2013/SRIM
        Outputs/Oxygen in  {}- O.txt".format(ElementName1),"r")
else:
    OXYGEN_IN_AL2O3=open("C:/Users/Julien Bouix/Desktop/SRIM-2013/SRIM
        Outputs/Oxygen in {}- O.txt".format(ElementName1),"r")
O_IN_Al2O3_lines=OXYGEN_IN_AL2O3.readlines()
OXYGEN_IN_AL2O3.close()

alline=25
eV=1
keV=0
MeV=0
ELIST=[]
SELIST=[]
SNLIST=[]

while O_IN_Al2O3_lines[alline][0]!='-':

    if TakeFromLine(O_IN_Al2O3_lines[alline],1,8)<TakeFromLine(
        O_IN_Al2O3_lines[alline-1],1,8) and keV==1:
        keV=0
        MeV=1
    if TakeFromLine(O_IN_Al2O3_lines[alline],1,8) < TakeFromLine(
        O_IN_Al2O3_lines[alline-1],1,8) and keV==0 and eV==1:
        keV=1
        eV=0

    ELIST.append((MeV*1e6+keV*1e3+eV)*TakeFromLine(O_IN_Al2O3_lines[
        alline],1,8))
    SELIST.append(TakeFromLine(O_IN_Al2O3_lines[alline],15,24))
    SNLIST.append(TakeFromLine(O_IN_Al2O3_lines[alline],26,35))

    alline+=1

SEfunctionO = interp1d(ELIST,SELIST,fill_value="extrapolate")
SNfunctionO = interp1d(ELIST,SNLIST,fill_value="extrapolate")

## for X next :

if len(ElementName1)==1:
    ALUMINIUM_IN_AL2O3=open("C:/Users/Julien Bouix/Desktop/SRIM-2013/
        SRIM Outputs/{} in  {}- O.txt".format(CompleteName1,
        ElementName1),"r")
else:
    ALUMINIUM_IN_AL2O3=open("C:/Users/Julien Bouix/Desktop/SRIM-2013/
        SRIM Outputs/{} in {}- O.txt".format(CompleteName1,ElementName1
        ),"r")
Al_IN_Al2O3_lines=ALUMINIUM_IN_AL2O3.readlines()
ALUMINIUM_IN_AL2O3.close()

alline=25
eV=1
keV=0
MeV=0
ELIST=[]
SELIST=[]
SNLIST=[]
```

```

while Al_IN_Al2O3_lines[alline][0]!= '-':

    if TakeFromLine(Al_IN_Al2O3_lines[alline],1,8)<TakeFromLine(Al_IN_Al2O3_lines[alline -1],1,8) and keV==1:
        keV=0
        MeV=1
    if TakeFromLine(Al_IN_Al2O3_lines[alline],1,8) < TakeFromLine(Al_IN_Al2O3_lines[alline -1],1,8) and keV==0 and eV==1:
        keV=1
        eV=0

    ELIST.append((MeV*1e6+keV*1e3+eV)*TakeFromLine(Al_IN_Al2O3_lines[alline],1,8))
    SELIST.append(TakeFromLine(Al_IN_Al2O3_lines[alline],15,24))
    SNLIST.append(TakeFromLine(Al_IN_Al2O3_lines[alline],26,35))

    alline+=1

SEfunctionX = interp1d(ELIST,SELIST,fill_value="extrapolate")
SNfunctionX = interp1d(ELIST,SNLIST,fill_value="extrapolate")

## EDITING OF THE TRIM.IN FILE

    # We select the type of calculation we want :
    # 1: Ion distribution and Quick Calculation of Damade (Kinchin-Pease) (TYPE A COLLISION.txt file)
    # 2: Detailed Calculation with full Damage Cascade (TYPE B COLLISION.txt file)
    # 3: Monolayer Collision Steps / Surface Sputtering (TYPE B COLLISION.txt file) (monolayer means that the ion react with each monolayer it gets through))
    # From 4 to the end, it uses TRIM.DAT :
    # 4: Ion with specific energy/ angle/ depth (quick Damage)
    # 5: Ion with specific energy/ angle/ depth (full cascades)
    # 6: Recoil cascades with neutrons (full cascades)
    # 7: Recoil cascades and monolayer steps (full cascades) (TYPE B COLLISION.txt file)
    # 8: Recoil cascades with neutrons (quick damage : Kinchin Pease)
# FOR NOW I TAKE 8


TRIMIN = open("TRIM.IN","r")
TRIMINNEW = open("TRIMN.IN","w")
TRIMINLINES = TRIMIN.readlines()
line=0


while TRIMINLINES[line][0] != 'C' :
    TRIMINNEW.write(TRIMINLINES[line])
    line+=1
TRIMINNEW.write(TRIMINLINES[line])
line+=1
TRIMINNEW.write('                              6                              0          0\n') # FOR NOW I TAKE 6
line+=1

while TRIMINLINES[line][0] != 'T' :
    TRIMINNEW.write(TRIMINLINES[line])
    line+=1
TRIMINNEW.write(TRIMINLINES[line])
line+=1
TRIMINNEW.write('"H (10) into Layer 1                    "        2
        1\n')
line+=1
TRIMINNEW.write('PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.) [=0 0
    for Viewing Full Target] \n')
line+=1
TRIMINNEW.write('        0                        0            0\n')
line+=1

while TRIMINLINES[line][0] != 'A' :
    TRIMINNEW.write(TRIMINLINES[line])
    line+=1

TRIMINNEW.write('Atom 1 = {} ='.format(ElementName1)+' '*(7-(len(
    ElementName1)-2))+'{}'.format(AtomicNumber1)+' '*(3-np.int(np.log10
    (AtomicNumber1)))+'{}\n'.format(MolarMass1))
line+=1
TRIMINNEW.write('Atom 2 = {} ='.format(ElementName2)+' '*(7-(len(
    ElementName2)-2))+'{}'.format(AtomicNumber2)+' '*(3-np.int(np.log10
    (AtomicNumber2)))+'{}\n'.format(MolarMass2))
line+=1

TRIMINNEW.write('Layer    Layer Name /              Width Density
    {}({})     {}({})\n'.format(ElementName1,AtomicNumber1,ElementName2,
    AtomicNumber2))
line+=1
TRIMINNEW.write('Numb.    Description               (Ang) (g/cm3)
    Stoich\n')
line+=1
TRIMINNEW.write(' 1      "Layer 1"           {}  {}      {}      {}\n'.
    format(Xmax,VolumicMass,stoiechiometry,1-stoiechiometry))

TRIMINNEW.write('0  Target layer phases (0=Solid, 1=Gas)\n0 \nTarget
    Compound Corrections (Bragg)\n 1  \nIndividual target atom
    displacement energies (eV)\n      25      28\nIndividual target
    atom lattice binding energies (eV)\n       3       3\nIndividual
    target atom surface binding energies (eV)\n    3.36       2\
    nStopping Power Version (1=2011, 0=2011)\n 0\n ')

TRIMIN.close()
TRIMINNEW.close()
os.remove('TRIM.IN')
os.rename('TRIMN.IN','TRIM.IN')



bmin1 = -RN-RM1
bmin2 = -RN-RM2
ProbabilityBlock = 1.25 # we run simulation of collision while T_max is
     greater than the displacement energy but then, the impact
    parameter distribution will not necessarily give T_max

        ## FOR EACH NEUTRON INCOMING, WE SIMULATE ITS TRAVEL INSIDE THE
            TARGET MATERIAL AND WHAT ATOMS IT WILL KNOCK OFF.
```

```

ListOfNumberOfCollision=[]
ListOfFinalEnergyOfNeutron=[]
ListOfLateralDispersion=[]
ListOfDepthPenetration=[]
ListOfEndingMode=[]  # 1 for backscattering, 2 for transmission, 3 for
    lack of energy
ListOfCollisionsForBack=[]
ListOfCollisionsForTrans=[]
ListOfCollisionsForLack=[]
ListOfVacancies=[]
ListOfDPAPerParticle=[]
ListOfVacanciesForBack=[]
ListOfVacanciesForTrans=[]
ListOfVacanciesForLack=[]
ListOfWhichOne=[] #0 for O and 1 for X

#For each primary cascade, we add an element to the following lists:
IONIZATIONListn=[]
BINDINGListn=[]
PHONONListn=[]

ionizationpercentage=[]
bindingpercentage=[]
phononpercentage=[]

CurrentNeutronIndex=1

for NEUTRONINDEX in range(NumberOfNeutrons):

    Xdepth=[]
    Ein = ListOfNeutronEnergy[NEUTRONINDEX]

    ## In a nuclear reactor, there is no collimated beam of neutron, so
        I consider an isotropic neutron direction impinging on the
        target
    alphaangle=(rd.random()-0.5)*np.pi
    betaangle=rd.random()*2*np.pi
    axe=[np.cos(alphaangle),np.sin(alphaangle)*np.sin(betaangle),np.sin
        (alphaangle)*np.cos(betaangle)]  #axe is the [X,Y,Z] unit
        vector that indicate the current direction of the neutron
        inside the target. X +Y +Z =1

    TRIMDAT_INPUT=[]
    CollisionIndex=0
    BACKSCATTERED = False
    TRANSMITTED = False
    T_max1 = Ein*(Mneutron*AtomicMassOfTarget1)/((Mneutron+
        AtomicMassOfTarget1)**2)
    T_max2 = Ein*(Mneutron*AtomicMassOfTarget2)/((Mneutron+
        AtomicMassOfTarget2)**2)



    while (T_max1 > ProbabilityBlock*DisplacementEnergy1 or T_max2 >
        ProbabilityBlock*DisplacementEnergy2) and BACKSCATTERED==False
        and TRANSMITTED==False:

```

```
        TRIMDAT_INPUT.append([0 for i in range(8)]) # We prepare the room for the information concerning a new collision

        ## The simulation of the impact parameter b:
            ## I first imposed a uniform distribution in he interval [-RN-RM,RN+RM] but the backscattering is too high (high probability to hit the atom at its center) and physically,due to channeling is a real crystal, I assume that most impact will occur with a greater probability at the border of the atom.

        #impactparameter=rd.uniform(bmin,-1*bmin)

            ##To correct this non uniformity, I try to model the impact parameter distribution with a parabolic law with a minimum at its center (b=0) and its maximum at {-RN-RM, RN+RM}. But this correction drastically reduces the transferred energy to the atoms making none of them energetic enough to produce a collision cascade.

        #impactparameter=parabolic_law(-RN-RM,RN+RM,DIFF) #diff of 100 is very selective: it means that the parabolic law has a value at its border 100 times greaater than at its center.

            ## Finally I dismiss the channeling due to the huge relative difference of size between the nucleus radius and the interatomic distance and consider a uniform distribution along the whole area of the radius so a infinitesimal area of 2*pi*b*db for having an impact parameter b.

        impactparameter1=LinearCrown(-bmin1)
        impactparameter2=LinearCrown(-bmin2)

        if CollisionIndex==0:

            CurrentSigmaMacroScattering1 = SigmaMacro(VolumicMass, CompoundMolarMass,1,SigmaMicroScatteringONEAtE1, SigmaMicroScatteringONEAtE2,SigmaMicroScatteringTWOAtE1,SigmaMicroScatteringTWOAtE2,E1,E2,Ein)
            DistanceUntilNextCollision1 = rd.expovariate(CurrentSigmaMacroScattering1)

            CurrentSigmaMacroScattering2 = SigmaMacro(VolumicMass, CompoundMolarMass,0,SigmaMicroScatteringONEAtE1, SigmaMicroScatteringONEAtE2,SigmaMicroScatteringTWOAtE1,SigmaMicroScatteringTWOAtE2,E1,E2,Ein)
            DistanceUntilNextCollision2 = rd.expovariate(CurrentSigmaMacroScattering2)

            if DistanceUntilNextCollision1>=DistanceUntilNextCollision2: # then collision with oxygen(2)

                ListOfWhichOne.append(0)
                PositionX=DistanceUntilNextCollision2*np.cos(Angle(axe,[1,0,0]))
                LateralPositionY= DistanceUntilNextCollision2*np.cos(Angle(axe,[0,1,0]))
                LateralPositionZ= DistanceUntilNextCollision2*np.cos(Angle(axe,[0,0,1]))
                TransferredEnergy = TransferredEnergyCoeff(impactparameter2,RN,RM2)*(Ein)*(Mneutron*AtomicMassOfTarget2)/((Mneutron+AtomicMassOfTarget2)**2)

            else: #then it is X

                ListOfWhichOne.append(1)
                PositionX=DistanceUntilNextCollision1*np.cos(Angle(axe,[1,0,0]))
                LateralPositionY= DistanceUntilNextCollision1*np.cos(Angle(axe,[0,1,0]))
                LateralPositionZ= DistanceUntilNextCollision1*np.cos(Angle(axe,[0,0,1]))
                TransferredEnergy = TransferredEnergyCoeff(impactparameter1,RN,RM1)*(Ein)*(Mneutron*AtomicMassOfTarget1)/((Mneutron+AtomicMassOfTarget1)**2)

        ## We add the information concerning the localisation of the current collision in the TRIM.DAT file

        if ListOfWhichOne[-1]==1:
            TRIMDAT_INPUT[CollisionIndex][0]=AtomicNumber1
        else:
            TRIMDAT_INPUT[CollisionIndex][0]=AtomicNumber2


        TRIMDAT_INPUT[CollisionIndex][1]=int(TransferredEnergy)#,0 decimal_round(TransferredEnergy))

        # X localisation of the collision
        TRIMDAT_INPUT[CollisionIndex][2]=np.int(PositionX)

        # Y and Z (Lateral Positions) localisation of the collision:
        TRIMDAT_INPUT[CollisionIndex][3]=np.int(LateralPositionY)
        TRIMDAT_INPUT[CollisionIndex][4]=np.int(LateralPositionZ)

        ## We run the simulation of the collision (see above for the definition of COLLISSION function I defined.

        if ListOfWhichOne[-1]==1:
             [axe,recoilaxe,Ein,TransferredEnergy,alpha,scattering_angle]=COLLISION(axe,impactparameter1,RN,RM1,Mneutron,AtomicMassOfTarget1,Ein)
        else:
             [axe,recoilaxe,Ein,TransferredEnergy,alpha,scattering_angle]=COLLISION(axe,impactparameter2,RN,RM2,Mneutron,AtomicMassOfTarget2,Ein)

        ## We add the information concerning the knocked atom direction (cos(X), cos(Y) and cos(Z)) during the current collision in the TRIM.DAT file

        # Ion/Atom direction:
        TRIMDAT_INPUT[CollisionIndex][5]=round(recoilaxe[0],4)
        TRIMDAT_INPUT[CollisionIndex][6]=round(recoilaxe[1],4)
        TRIMDAT_INPUT[CollisionIndex][7]=round(recoilaxe[2],4)

        ## What changes for the next collision ?

        T_max1 = Ein*(Mneutron*AtomicMassOfTarget1)/((Mneutron+AtomicMassOfTarget1)**2)
        CurrentSigmaMacroScattering1 = SigmaMacro(VolumicMass, CompoundMolarMass,1,SigmaMicroScatteringONEAtE1, SigmaMicroScatteringONEAtE2,SigmaMicroScatteringTWOAtE1, SigmaMicroScatteringTWOAtE2,E1,E2,Ein)
        DistanceUntilNextCollision1 = rd.expovariate(CurrentSigmaMacroScattering1)

        T_max2 = Ein*(Mneutron*AtomicMassOfTarget2)/((Mneutron+AtomicMassOfTarget2)**2)
        CurrentSigmaMacroScattering2 = SigmaMacro(VolumicMass, CompoundMolarMass,0,SigmaMicroScatteringONEAtE1, SigmaMicroScatteringONEAtE2,SigmaMicroScatteringTWOAtE1, SigmaMicroScatteringTWOAtE2,E1,E2,Ein)
        DistanceUntilNextCollision2 = rd.expovariate(CurrentSigmaMacroScattering2)

        if DistanceUntilNextCollision1<=DistanceUntilNextCollision2: # then Al
            ListOfWhichOne.append(1)
            PositionX += DistanceUntilNextCollision1*np.cos(Angle(axe,[1,0,0]))
            LateralPositionY += DistanceUntilNextCollision1*np.cos(Angle(axe,[0,1,0]))
            LateralPositionZ += DistanceUntilNextCollision1*np.cos(Angle(axe,[0,0,1]))

        else:
            ListOfWhichOne.append(0)
            PositionX += DistanceUntilNextCollision2*np.cos(Angle(axe,[1,0,0]))
            LateralPositionY += DistanceUntilNextCollision2*np.cos(Angle(axe,[0,1,0]))
            LateralPositionZ += DistanceUntilNextCollision2*np.cos(Angle(axe,[0,0,1]))

        CollisionIndex +=1

        #print('{}:{}'.format(CurrentNeutronIndex,CollisionIndex))
        if PositionX<0:
            BACKSCATTERED = True
            ListOfEndingMode.append(1)
            #print('Backscattering occured at a relative neutron energy {} and after {} collision(s)'.format(Ein/EnergyNeutronIn,CollisionIndex))

        if PositionX>XmaxNoScientific:
            TRANSMITTED=True
            ListOfEndingMode.append(2)
            #print('Transmission occured at a relative neutron energy {} and after {} collision(s)'.format(Ein/EnergyNeutronIn,CollisionIndex))

        if T_max1 < ProbabilityBlock*DisplacementEnergy1 and T_max2 < ProbabilityBlock*DisplacementEnergy2:
            ListOfEndingMode.append(3)
            #print('The neutron has not enough energy to continue its travel inside the target material and is stopped after {} collisions'.format(CollisionIndex))

    X=[TRIMDAT_INPUT[i][2]*10**(-8) for i in range(CollisionIndex)]# all X in cm
    Y=[TRIMDAT_INPUT[i][3]*10**(-7) for i in range(CollisionIndex)]
    Z=[TRIMDAT_INPUT[i][4]*10**(-7) for i in range(CollisionIndex)]# all Y and Z in mm
    X0=[0]+X
    Y0=[0]+Y
    Z0=[0]+Z

    if CurrentNeutronIndex/50==int(CurrentNeutronIndex/50):
        print('We are at the {} neutron simulation'.format(CurrentNeutronIndex))
    CurrentNeutronIndex+=1

    ListOfDepthPenetration.append(absmaxlist(X))
    ListOfLateralDispersion.append(max(absmaxlist(Y),absmaxlist(Z)))
    ListOfNumberOfCollision.append(CollisionIndex)
    ListOfFinalEnergyOfNeutron.append(Ein)

    ## Correction of the TRIM.DAT input to delete all collisions where the transferred energy is lower than Ed and remove Eb to the kinetic energy of the atom:

    I=[]
    for i in range(len(TRIMDAT_INPUT)):

        if TRIMDAT_INPUT[i][1] >= max(DisplacementEnergy1, DisplacementEnergy2):
            I.append(i)

    TRIMDAT_INPUT=[TRIMDAT_INPUT[i] for i in I]

    for i in range(len(TRIMDAT_INPUT)):
        if TRIMDAT_INPUT[i][0]== AtomicNumber1:
            TRIMDAT_INPUT[i][1] = TRIMDAT_INPUT[i][1] - LatticeBindingEnergy1
        elif TRIMDAT_INPUT[i][0]== AtomicNumber2:
            TRIMDAT_INPUT[i][1] = TRIMDAT_INPUT[i][1] - LatticeBindingEnergy2

    ## PLOT THE NEUTRON PATH INSIDE THE TARGET MATERIAL :
    #

    # plt.ion()
    # fig = plt.figure()
    # ax = fig.gca(projection="3d")
    # ax.plot(X0,Y0,Z0)
    #
    # ax.set_xlabel('Depth inside the target material in cm')
    # ax.set_xlim(0,XmaxNoScientific*10**(-8))
    # ax.set_ylabel('Lateral position Y in mm')
    # ax.set_ylim(-max(absmaxlist(Y),DiameterOfTheBeam*10**(-7)/2), max
        (absmaxlist(Y),DiameterOfTheBeam*10**(-7)/2))
    # ax.set_zlabel('Lateral position Z in mm')
    # ax.set_zlim(-max(absmaxlist(Z),DiameterOfTheBeam*10**(-7)/2), max
        (absmaxlist(Z),DiameterOfTheBeam*10**(-7)/2))

   ##    plt.show()


## EDITING OF THE TRIM.DAT FILE ##For each collision with an atom fo
    the lattice, we add to the TRIM.DAT file the information needed

# We need as an input for the TRIM.DAT the list of all PKA made by the
    neutron code with, for each one of them a list containing their
    atomic number, energy (eV), depth (A), lateral position (X and Y)
    and their direction (Cos(X), cos(Y) and cos(Z)). The input is a
    list of list of 8 elements.

    TRIMDAT = open("TRIM.DAT","r")
    TRIMDATNEW = open("TRIMN.DAT","w")
    TRIMLINES = TRIMDAT.readlines()
    line=0


        ## Due to the issue with too high depth, I give 0 to Y and Z
            and 5000A to X.

    for i in range(len(TRIMDAT_INPUT)):
        Xdepth.append(TRIMDAT_INPUT[i][2])
        TRIMDAT_INPUT[i][2]=5000
        TRIMDAT_INPUT[i][3]=0
        TRIMDAT_INPUT[i][4]=0

    while TRIMLINES[line][0] != '  ':
        TRIMDATNEW.write(TRIMLINES[line])
        line+=1
    TRIMDATNEW.write('            Recoils from neutrons (Maxwell) in
        {}({})                          \n'.format(CompoundName,Xmax))
    line+=1
    TRIMDATNEW.write(TRIMLINES[line])
    line+=1
    TRIMDATNEW.write(TRIMLINES[line])
    line+=1
    IndexIon=1

    for PKA in TRIMDAT_INPUT:
        if PKA[1]==0:
            pass
        else:

            if PKA[0]>=10:     # According to the atomic number we have
                to adjust the spacing between the event name and the
                atomic number
                TRIMDATNEW.write('{}'.format(IndexIon)+' '*(7-len(str(
                    int(IndexIon)))))
            else:
                TRIMDATNEW.write('{}'.format(IndexIon)+' '*(8-len(str(
                    int(IndexIon)))))

            # WRITE ATOMIC NUMBER:
            TRIMDATNEW.write(str(PKA[0])+'      ')

            # WRITE ENERGY :

            TRIMDATNEW.write(str(PKA[1])+' '*(9-len(str(PKA[1]))))

            # WRITE X:
            Xscientific='{:.1e}'.format(PKA[2])
            TRIMDATNEW.write(Xscientific+' '*(9-len(Xscientific)+ajout(
                np.sign(PKA[3]))))

            # WRITE Y:
            Yscientific='{:.0e}'.format(PKA[3])
            if PKA[3]>=0:
                TRIMDATNEW.write(Yscientific+' '*(8-len(Yscientific)+
                    ajout(np.sign(PKA[4]))))
            else:
                TRIMDATNEW.write(Yscientific+' '*(9-len(Yscientific)+
                    ajout(np.sign(PKA[4]))))

            # WRITE Z:
            Zscientific='{:.1e}'.format(PKA[4])
            if PKA[4]>=0:
                TRIMDATNEW.write(Zscientific+' '*(9-len(Zscientific)+
                    ajout(np.sign(PKA[5]))))
            else:
                TRIMDATNEW.write(Zscientific+' '*(10-len(Zscientific)+
                    ajout(np.sign(PKA[5]))))

            # WRITE COS(X)
            if PKA[5]>=0:
                TRIMDATNEW.write(str(PKA[5])+' '*(9-len(str(PKA[5]))+
                    ajout(np.sign(PKA[6]))))
            else:
                TRIMDATNEW.write(str(PKA[5])+' '*(10-len(str(PKA[5]))+
                    ajout(np.sign(PKA[6]))))

            # WRITE COS(Y)
            if PKA[6]>=0:
                TRIMDATNEW.write(str(PKA[6])+' '*(9-len(str(PKA[6]))+
                    ajout(np.sign(PKA[7]))))
            else:
                TRIMDATNEW.write(str(PKA[6])+' '*(10-len(str(PKA[6]))+
                    ajout(np.sign(PKA[7]))))

            # WRITE COS(Z)
            TRIMDATNEW.write(str(PKA[7])+'\n')

            line+=1
            IndexIon+=1

    TRIMDAT.close()
    TRIMDATNEW.close()
    os.remove("TRIM.DAT")
    os.rename("TRIMN.DAT","TRIM.DAT")
```

```


## EXECUTION ?
# a windown opens in the case 6 and 8 So try to use 7 (Recoil cascades
    and monolayer steps (full cascades)) which is good. TYPE OF
    COLLISION.txt is B for 7.

    os.startfile('C:/Users/Julien Bouix/Desktop/SRIM-2013/TRIM.exe')
    time.sleep(15+CollisionIndex*1/6)


    COLLISION = open("C:/Users/Julien Bouix/Desktop/SRIM-2013/SRIM
        Outputs/COLLISION.txt","r")
    COLLISIONLINES=COLLISION.readlines()
    COLLISION.close()
    LengthCOLLISION=len(COLLISIONLINES)

    PrimaryPhonon=[]

    Xpkalist=[]
    Ypkalist=[]
    Zpkalist=[]

    IonEnergyBeforeCollision=[]

    NbOfPKAn=0
    for i in range(LengthCOLLISION):
        if COLLISIONLINES[i][0] =='    ':
            NbOfPKAn +=1

    NbOfPKAn=int(NbOfPKAn/2)

    EnergyTransmittedToPKAn=[]

    ListOfIONIZATION=[]
    ListOfIONIZATIONPKA=[]
    ListOfBINDING=[]
    ListOfBINDINGexceptprimary=[]
    ListOfPHONON=[]
    Nreplacement=[]
    Nvac=[]
    currentline=26
    PKAALORO=[]
    NbofProblemExcept=0

    for nbpkan in range(NbOfPKAn):

        IONIZATIONRecoil=[]
        IONIZATIONRecoilPKA=[]
        NUCLEARRecoil=[]
        BINDINGRecoil=0
        PHONONRecoil=0
        ALORO=[]
        Xlist=[]
        Ylist=[]
        Zlist=[]
        Nreplacement.append([0,0])
        Nvac.append([0,0])
        RecoilEnergyList=[]
        currentline +=2

        if currentline==LengthCOLLISION:
            break

        while COLLISIONLINES[currentline][0]!=' ' and currentline<=LengthCOLLISION-2: #if no recoil cascade for this pka
            currentline+=1

        if currentline==LengthCOLLISION-1:
            break

        # start of a new cascade
        linecontent=COLLISIONLINES[currentline]

        try:

            linecontent=COLLISIONLINES[currentline]
            Xpkalist.append(TakeFromLine(linecontent,18,28))
            Ypkalist.append(TakeFromLine(linecontent,29,39))
            Zpkalist.append(TakeFromLine(linecontent,40,50))
            #Selistn.append(TakeFromLine(linecontent,51,58))
            EnergyTransmittedToPKAn.append(TakeFromLine(linecontent,64,74))
        except:
            NbofProblemExcept+=1
            currentline+=+1
            linecontent=COLLISIONLINES[currentline]
            Xpkalist.append(TakeFromLine(linecontent,18,28))
            Ypkalist.append(TakeFromLine(linecontent,29,39))
            Zpkalist.append(TakeFromLine(linecontent,40,50))
            #Selistn.append(TakeFromLine(linecontent,51,58))
            EnergyTransmittedToPKAn.append(TakeFromLine(linecontent,64,74))

        currentline+=3
        insidecounter=0
        while COLLISIONLINES[currentline][0] ==' ':
            linecontent=COLLISIONLINES[currentline]
            if insidecounter==0:
                if COLLISIONLINES[currentline][10]+COLLISIONLINES[currentline][11]==str(AtomicNumber1): #if X
                    PKAALORO.append(0)
                else:
                    PKAALORO.append(1)
            insidecounter+=1
            if COLLISIONLINES[currentline][10]+COLLISIONLINES[currentline][11]==str(AtomicNumber1): #if X

                ALORO.append(0)
                Nvac[-1][0]+=float(COLLISIONLINES[currentline][58])
                Nreplacement[-1][0]+=float(COLLISIONLINES[currentline][63])
            else:
                ALORO.append(1)
                Nvac[-1][1]+=float(COLLISIONLINES[currentline][58])
                Nreplacement[-1][1]+=float(COLLISIONLINES[currentline
                    ][63])
            RecoilEnergyList.append(TakeFromLine(linecontent,15,25))
            Xlist.append(TakeFromLine(linecontent,26,35))
            Ylist.append(TakeFromLine(linecontent,36,46))
            Zlist.append(TakeFromLine(linecontent,47,57))
            currentline+=1

        if COLLISIONLINES[currentline][0] ==' ':

        ##ONLY BY READING AND DEDUCING....

        if len(RecoilEnergyList)==1:

            if ALORO[0]==1: # if Oxygen

                BINDINGRecoil      += FormationEnergyOfInterstitial2
                IONIZATIONRecoil.append(0)

            else:

                BINDINGRecoil      += FormationEnergyOfInterstitial1
                IONIZATIONRecoil.append(0)

        else:

            lengthLIST=len(Xlist)

            ## BINDINGS (stored energy)

            if ALORO[0]==1:

                ListOfBINDING.append(int(Nvac[-1][0]-Nreplacement[-1][0])*FormationEnergyOfInterstitial1+int(Nvac[-1][1]-1-Nreplacement[-1][1])*FormationEnergyOfInterstitial2) # Each collision will end up with the outgoing particle as interstittial or replacement (not counted in this case)
            else:

                ListOfBINDING.append(int(Nvac[-1][0]-1-Nreplacement[-1][0])*FormationEnergyOfInterstitial1+int(Nvac[-1][1]-Nreplacement[-1][1])*FormationEnergyOfInterstitial2)


            maxdistance = max([np.sqrt((Xlist[i]-Xlist[0])**2+(Ylist[i]-Ylist[0])**2+(Zlist[i]-Zlist[0])**2) for i in range(len(Xlist))])

            ListofSKAIndexes=[1]
            ListOfInitialKineticEnergiesSKA=[RecoilEnergyList[1]]

            for i in range(2,len(Xlist)):
                if np.sqrt((Xlist[i]-Xlist[i-1])**2+(Ylist[i]-Ylist[i-1])**2+(Zlist[i]-Zlist[i-1])**2) > maxdistance/4.5:
                    ListofSKAIndexes.append(i)
```

```
                    ListOfInitialKineticEnergiesSKA.append(RecoilEnergyList[i])

            ## CALCULATION OF PKA CASCADE

            sumenergy=0
            ListofSKAIndexes.reverse()
            if ALORO[0]==1: #if Oxygen
                PKAALORO.append(1)
                for i in ListofSKAIndexes:

                    IONIZATIONRecoilPKA.append(RecoilEnergyList[0]-sumenergy-EnergyAfterSPSegment(SEfunctionO,RecoilEnergyList[0]-sumenergy,np.sqrt((Xlist[i]-Xlist[i-1])**2+(Ylist[i]-Ylist[i-1])**2+(Zlist[i]-Zlist[i-1])**2),Nsegment))

            else:
                PKAALORO.append(0)
                for i in ListofSKAIndexes:

                    IONIZATIONRecoilPKA.append(RecoilEnergyList[0]-sumenergy-EnergyAfterSPSegment(SEfunctionX,RecoilEnergyList[0]-sumenergy,np.sqrt((Xlist[i]-Xlist[i-1])**2+(Ylist[i]-Ylist[i-1])**2+(Zlist[i]-Zlist[i-1])**2),Nsegment))

            IONIZATIONRecoilPKA2=sum(IONIZATIONRecoilPKA)
            ListOfIONIZATIONPKA.append(IONIZATIONRecoilPKA2)
        ListOfIONIZATION.append(sum(IONIZATIONRecoil))
        ListOfBINDING.append(BINDINGRecoil)

    IONIZATIONListn.append(sum(ListOfIONIZATION)+sum(ListOfIONIZATIONPKA))
    BINDINGListn.append(sum(ListOfBINDING))
    PHONONListn.append(ListOfNeutronEnergy[NEUTRONINDEX]-IONIZATIONListn[-1]-BINDINGListn[-1])

    listvac=0
    for i in range(NbOfPKAn):

        if ReferenceDepth1 < Xdepth[i] < ReferenceDepth2:

            listvac += Nvac[i][0]+Nvac[i][1]
    ListOfVacancies.append(listvac)

    ListOfDPAPerParticle.append(sum(ListOfVacancies)/((NEUTRONINDEX+1)*(ReferenceDepth2-ReferenceDepth1)*10**(-8)*ReferenceArea*NumberDensity))

##Create an histogram:

for i in range(NumberOfNeutrons):
    if ListOfEndingMode[i]==1:
        ListOfCollisionsForBack.append(ListOfNumberOfCollision[i])
        ListOfVacanciesForBack.append(ListOfVacancies[i])

    if ListOfEndingMode[i]==2:
        ListOfCollisionsForTrans.append(ListOfNumberOfCollision[i])
        ListOfVacanciesForTrans.append(ListOfVacancies[i])

    if ListOfEndingMode[i]==3:
        ListOfCollisionsForLack.append(ListOfNumberOfCollision[i])
        ListOfVacanciesForLack.append(ListOfVacancies[i])

plt.ion()
plt.plot([i for i in range(1,len(ListOfDPAPerParticle)+1)],ListOfDPAPerParticle)
plt.title('The average dpa per neutron projectile in the reference volume')
plt.ylabel('average dpa per neutron projectile')
plt.xlabel('Neutron index')
plt.show()

SumVacancies = int(sum(ListOfVacancies))
print('The sum of all vacancies caused by {} neutrons is {}'.format(NumberOfNeutrons,SumVacancies))

resultfile=open("C:/Users/Julien Bouix/Desktop/SRIM-2013/Result Of the neutron Code.txt",'w')

resultfile.write('        FOR {} NEUTRONS FOLLOWING A MAXWELL DISTRIBUTION FOR ENERGY (1 MeV MOST PROBABLE) TRAVELING INSIDE A {} CENTIMETERS THICK LAYER OF {}:\n\n'.format(NumberOfNeutrons,int(10**(-8)*XmaxNoScientific),CompoundName))

resultfile.write("-The average energy of the neutron is {}eV with a standard deviation of {}eV\n".format('{:.2e}'.format(meanlist(ListOfNeutronEnergy)),'{:.2e}'.format(round(ecart_type(ListOfNeutronEnergy),0))))
resultfile.write('------The maximum energy of one neutron is {}\n'.format('{:.2e}'.format(absmaxlist(ListOfNeutronEnergy))))
resultfile.write('------The minimum energy of one neutron is {}\n\n'.format('{:.2e}'.format(minlist(ListOfNeutronEnergy))))

resultfile.write("-The average number of collisions per neutron is {} with a standard deviation of {} collisions \n".format(np.int(meanlist(ListOfNumberOfCollision)),round(ecart_type(ListOfNumberOfCollision),0)))
resultfile.write('------The maximum number of collision caused by a single neutron is {}\n\n'.format(absmaxlist(ListOfNumberOfCollision)))

resultfile.write('-The process ends at {}% due to backscattering, {}% due to transmission, {}% due to the lack of \nneutrons energy to displace more atoms\n'.format(round(100*countlist(ListOfEndingMode,1)/NumberOfNeutrons,0),round(100*countlist(ListOfEndingMode,2)/NumberOfNeutrons,0),round(100*countlist(ListOfEndingMode,3)/NumberOfNeutrons,0)))

resultfile.write('------For backscattering only, the average number of collision is {} with a standard deviation of {}\n'.format(int(meanlist(ListOfCollisionsForBack)),int(ecart_type(ListOfCollisionsForBack))))
resultfile.write('------For transmission only,   the average number of collision is {} with a standard deviation of {}\n'.format(int(
meanlist(ListOfCollisionsForTrans)),int(ecart_type(ListOfCollisionsForTrans))))
resultfile.write('———For lack of energy only, the average number of collision is {} with a standard deviation of {}\n\n'.format(int(meanlist(ListOfCollisionsForLack)),int(ecart_type(ListOfCollisionsForLack))))

resultfile.write("–The average depth penetration of neutrons in the target material is {}cm with a standard deviation of {}cm\n".format(round(meanlist(ListOfDepthPenetration),2),round(ecart_type(ListOfDepthPenetration),2)))
resultfile.write('———The maximum penetration is {}cm\n\n'.format(round(absmaxlist(ListOfDepthPenetration),2)))

resultfile.write('–The average lateral dispersion is {}mm with a standard deviation of {}mm\n'.format(round(meanlist(ListOfLateralDispersion),0),round(ecart_type(ListOfLateralDispersion),0)))
resultfile.write('———The maximum lateral dispersion is {}mm\n\n'.format(round(absmaxlist(ListOfLateralDispersion),2)))
resultfile.write('–The average remaining relative energy of the neutron after all these collisions is {} with a \nstandard deviation of {}\n\n'.format(round(meanlist([ListOfFinalEnergyOfNeutron[i]/ListOfNeutronEnergy[i] for i in range(len(ListOfNeutronEnergy))]),4),round(ecart_type([ListOfFinalEnergyOfNeutron[i]/ListOfNeutronEnergy[i] for i in range(len(ListOfNeutronEnergy))]),4)))

resultfile.write('– TRIM output states that the average of [all the vacancies caused by all PKA of one neutron] is {} with a \nstandard deviation of {}\n'.format(int(meanlist(ListOfVacancies)),int(ecart_type(ListOfVacancies))))
resultfile.write('———The sum of all vacancies caused by {} neutrons is {}\n'.format(NumberOfNeutrons,SumVacancies))
resultfile.write('——————For backscattering only, the average number of [above] is {} with a standard deviation of {}\n'.format(int(meanlist(ListOfVacanciesForBack)),int(ecart_type(ListOfVacanciesForBack))))
resultfile.write('——————For transmission only,  the average number of [above] is {} with a standard deviation of {}\n'.format(int(meanlist(ListOfVacanciesForTrans)),int(ecart_type(ListOfVacanciesForTrans))))
resultfile.write('——————For lack of energy only, the average number of [above] is {} with a standard deviation of {}\n\n'.format(int(meanlist(ListOfVacanciesForLack)),int(ecart_type(ListOfVacanciesForLack))))

resultfile.write('———The maximum average vacancies caused by all the PKA of one neutron is {}\n'.format(int(absmaxlist(ListOfVacancies))))
resultfile.write('———The minimum average vacancies caused by all the PKA of one neutron is {}\n\n'.format(int(minlist(ListOfVacancies))))

resultfile.write('–For neutron irradiation:\n')
resultfile.write('———The total energy loss to bindings per neutron is {}eV on average ({} %) with a standard deviation of {}eV \n'.format(int(meanlist(BINDINGListn)),round(100*sum(BINDINGListn)/sum(
    ListOfNeutronEnergy),2),int(ecart_type(BINDINGListn))))
resultfile.write('———The total energy loss to electrons (ionization)
    per neutron is {}eV on average ({} %) with a standard deviation of
    {}eV \n'.format(int(meanlist(IONIZATIONListn)),round(100*sum(
    IONIZATIONListn)/sum(ListOfNeutronEnergy),2),int(ecart_type(
    IONIZATIONListn))))
resultfile.write('———The total energy loss to phonons per neutron is
    {}eV on average ({} %) with a standard deviation of {}eV\n'.format(
    int(meanlist(PHONONListn)),round(100*sum(PHONONListn)/sum(
    ListOfNeutronEnergy),2),int(ecart_type(PHONONListn))))
resultfile.close()

time2=time.time()

print(time2-time1)
```