

UNIVERSITY OF BIRMINGHAM



Department of Metallurgy & Materials

Irradiation Damage Simulations of Platinum Group Metal Modified Austenitic Stainless Steels for Reactor Core Components

by Ben Palmer

A thesis submitted to the University of Birmingham for the degree of Doctor of Philosophy
Supervised by Dr Brian J Connolly, Dr Mark S D Read and Prof Alessandro Mottura

Abstract

Austenitic stainless steels have been used since the early days of nuclear power and they will play an important role in the construction of Generation III+ (Gen III+) and Generation IV (Gen IV) plant designs. Irradiation of components by neutrons is expensive with relatively few high flux reactors available. The experiments to represent 30-40 years within a reactor core are time consuming and produce radioactive waste as a by-product. Two alternatives to testing by irradiating with a neutron source are either irradiating with a proton beam or by modelling damage events with a computer.

The efficiency and computational power of computers continues to improve. First principles Density Functional Theory (DFT) and molecular dynamics (MD) are computational methods that are now able to give insights into why materials behave the way they do.

1. Proton sources are more widespread, they are relatively cheap and may be focused into a much smaller beam to concentrate the amount of damage per time unit. The radioactive waste may be significant but by carefully selecting the ion energy the electromagnetic repulsion between ions and nuclei reduces transmutation of target atoms whilst maintaining the amount and depth of damage into the material required for testing purposes.

A modified Bateman equation is derived to calculate the radioactivity of a proton irradiated target and a computer program (Activity) is created to implement the calculation using evaluated nuclear reaction cross section data. The program is compared to data measured from the irradiation of an iron sample using the University of Birmingham cyclotron. It is also used to model the radioactivity of a target irradiated to 100 displacements per atom (DPA) at a range of beam energies in order to minimise both the irradiation time and radioactive waste. The results show an increase in target activity of 3 orders of magnitude for a 25MeV proton beam when compared to a 10MeV proton beam.

2. Sensitisation of austenitic steels, by processes such as welding, depletes Cr from grain boundaries removing the corrosion resistant Cr_2O_3 passive layer. Previous (experimental) work has shown that these steels, when doped with Pd or Ru, retain corrosion resistance at the grain boundary[1]. This work takes a step towards investigating whether or not these platinum group metal (PGM)s deplete or saturate at the grain boundary while under irradiation and this may be studied with MD simulations. A computer program is developed to fit interatomic potentials to experimental data and DFT generated data. The properties of face centered cubic (FCC) Fe are computed using DFT. These data along with other properties and DFT generated data are used to derive interatomic potentials (with a dominant ZBL pair for close range collision interactions) for Fe-Pd. These potentials are a step towards MD simulations of the irradiated grain boundaries.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Several Gen III+ nuclear reactors have been proposed for construction in the UK (with Hinckley Point C now under construction) and Gen IV nuclear reactors are being researched and developed. New materials are required to withstand the extreme conditions in and around the core of these reactors. Austenitic stainless steels have been an important structural material in the industry, and may continue to be so, providing the problem of inter granular stress corrosion cracking (IGSCC) can be addressed. Doping these steels with platinum group metal (PGM)s has been seen to reduce IGSCC, but the effects on corrosion resistance are unknown for these steels when irradiated by a radiation field.

1.1 Motivation

I have often questioned the motivation behind this work, and at times it has been a challenge to see the wood through the trees. To help me focus throughout I would restate the motivation and objectives to myself.

Mass produced steels are not perfect repeating crystals, but are made up of small grains. Cr is added to steel to make stainless steel, and this is more resistant to corroding than steel. When steel is in a nuclear reactor, it will have to perform under extreme conditions, such as:

- high temperatures
- strain cause by high pressures and radiation induced swelling
- radiation damage and strains resulting from this damage
- a changing environment, for example radiolysis of water and transmutation of isotope due to radiation
- corrosive environments while in the reactor
- corrosive environments out of the reactor (e.g. fuel cladding in storage)

Radiation damages the steel in a number of ways, including directly knocking atoms out of their place within the lattice structure of the crystalline grains that make up the steel as well as changing the isotopes that make up the steel into new isotopes. An example of the latter is a neutron reacting with an Fe atom, transmuting it into a Co atom.

In time, the radiation damage causes the percentage of Cr at the surface of the grains to drop, and as it falls the steel loses its protection from corrosion at the boundary between the grains it is made of.

This work is divided into two parts.

1.1.1 Part 1: Activity Computer Program

The materials must be tested before being used in a nuclear reactor. One way to do this would be to place samples of the steel into a test reactor. This is expensive and, as a by-product, the steel sample becomes radioactive. It is difficult to create a large number of neutrons, but it is much easier, and cheaper, to create a beam of protons. Protons can be accelerated in a machine such as the Cyclotron at the University of Birmingham.

The damage that protons cause to a sample is not precisely the same as that caused by neutrons, but it is a cheaper alternative and is a trade-off between the cost and results. One side effect that proton irradiation shares with neutron irradiation is the creation of radioactive waste.

The first part of this work investigates exactly how radioactive a sample becomes when irradiated with a proton beam. An existing set of equations, named after Mathematician Harry Bateman, were modified, and a computer program was created to perform the calculation. The user inputs the constituent elements that make up the material, the ratio of these elements, and the irradiation settings. The program then estimates how radioactive the sample will be and the predicted gamma energies.

1.1.2 Part 2: Palladium-Iron Potential

Adding Cr to make stainless steel is not the only way to make a steel resistant to corrosion. Adding metals such as Mo and Pd to steel can increase the resistance to corrosion, but Mo is several hundred times the cost of Fe ore, and Pd is thousands of times as expensive.

Simulating radiation damage using a computer is now a feasible and sensible way to investigate how these materials will be affected by radiation damage, and the simulations may reveal insights that experiments are not able to show, either because they happen on too small a timescale or within the material at too small a length scale.

Key to the simulations success is being able to accurately calculate how the atoms interact with one another. The second part of this work concentrates on deriving a mathematical description of how Pd and Fe atoms interact with one another, which would then allow future simulations of steel with small amounts of Pd added to it.

Radiation causes Cr to be depleted at the grain boundary. If these simulations show that Pd is not depleted, it would suggest that the corrosion resistance is maintained, despite the decrease in the amount of Cr at the grain boundary due to radiation damage.

1.2 Nuclear Power in the UK

1.2.1 Nuclear Fuel

The primary fuel of commercial nuclear power stations since those first built in the 1950s has been U. U-235 is fissile and fissions with a high probability when thermal neutrons are captured, due to its very high thermal neutron fission cross section (fig. 1.1).

Natural U contains a small percentage of fissile U-235 (0.7%) with the rest being its heavier isotope, U-238. U-238 is fissionable, but its fission cross section is much lower than U-235 (fig. 1.1). Many of the design considerations for reactors in operation revolve around the amount of U-235 there is in the fuel and having a sufficient flux of thermal neutrons passing through the fuel to sustain a reaction.

A number of Gen IV designs use the fast neutron spectrum to fission U-238 and other fuels such as Th-232 to generate energy and breed other fissile isotopes. By moving away from U-235, this increases the potential fuel stockpile, but also brings its own set of challenges.

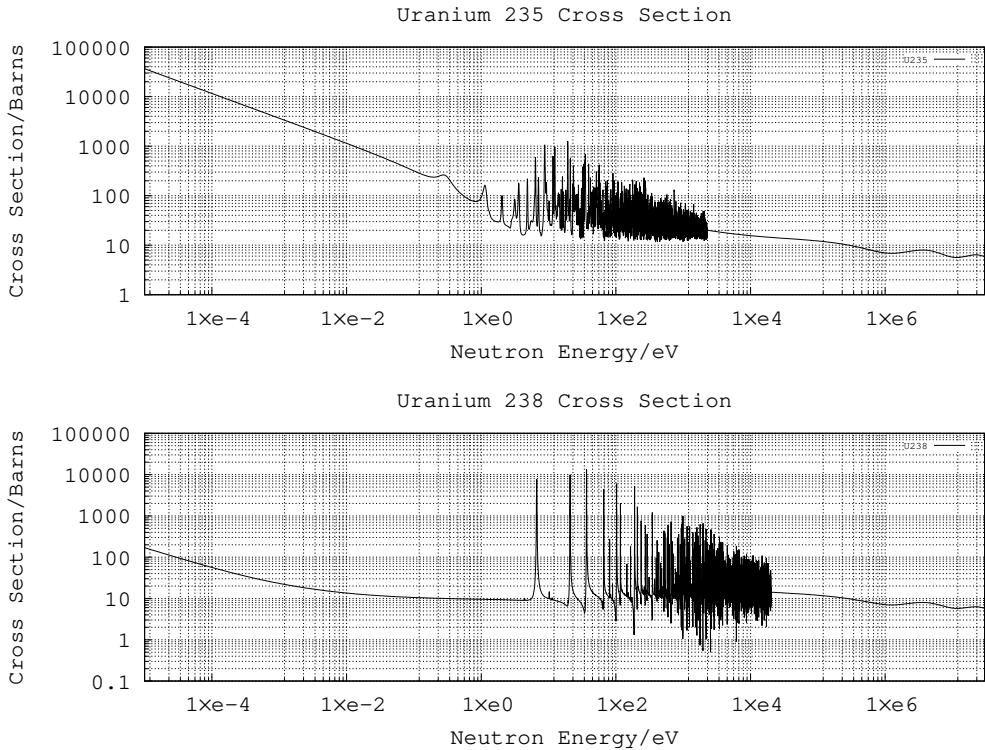


Figure 1.1: U-235 and U-238 Fission Cross Sections

1.2.2 Generation I Reactors

The first generation reactors were primarily prototype reactors. They included graphite moderated reactors, light water and heavy water reactors. Early reactors were designed to produce electricity, but also fissile material for nuclear weapons. With a power output of less than 2MW, the Gen I power station Calder Hall generated much less electricity than a modern nuclear power plant which may produce 1GW per reactor. It did, however, produce Pu for the UK's nuclear weapon programme.

Magnox type reactors were the first used in the United Kingdom. These reactors used natural U as a fuel and were carefully designed to produce energy despite using an un-enriched fuel. Graphite was used as a moderator, and the low neutron capture cross section of the Magnox Mn alloy cladding allowed a nuclear reaction to occur despite the fuel not being enriched.

In all there were 11 Magnox power stations built in the UK with 26 reactors in total. Construction of the first commercial reactor in the UK, Calder Hall, started in 1953 and it operated from 1956 to 2003. All have now shut down with the last, Wylfa in Anglesey, closing in 2015.

1.2.3 Generation II Reactors

Gen II marked the transition from prototype reactors to higher powered commercial reactors. There were a number of designs including the AGR, a graphite core CO_2 cooled reactor, that was implemented in the UK. Light water reactors included the PWR and BWR in the west and the VVER in the USSR. The RBMK was a graphite moderated reactor used in the USSR and the CANDU a heavy water reactor designed and used in Canada.

There are many of these reactors from this period still in operation around the world, and there have been recent implementations of these designs. There are 15 reactors currently operating in the UK, and they are all Gen II reactors. Of these, 14 are AGR and 1, Sizewell B, is a PWR (fig. 1.2).

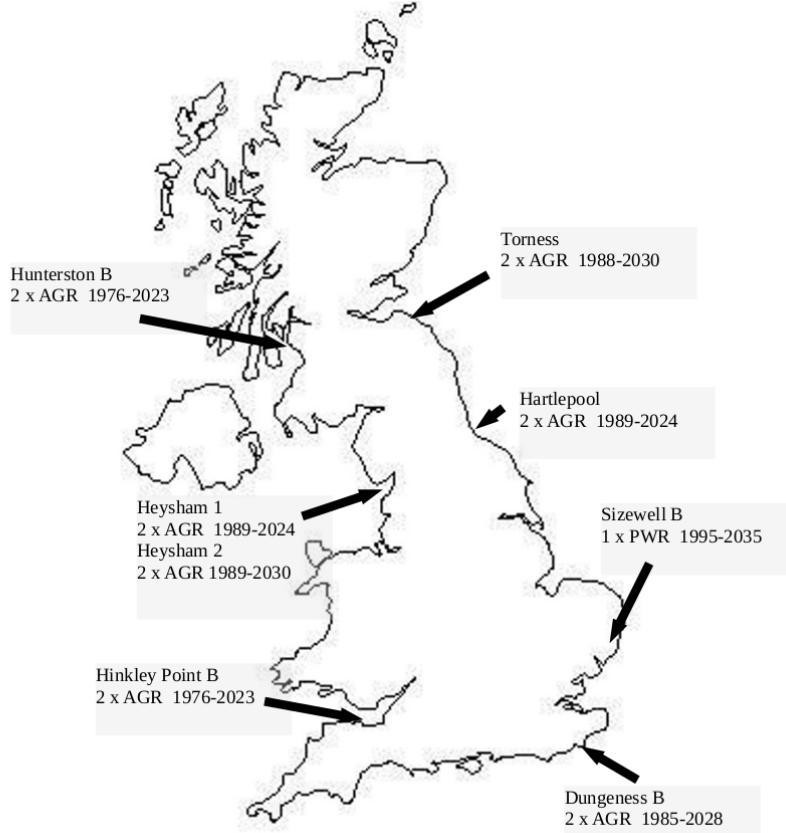


Figure 1.2: Remaining reactor locations in the UK

The planned life span for this generation of reactor was 30-40 years, although this has now been extended to 50-60 years in some cases. The AGR reactors in the UK have been operational for over 40 years, and by the time Hunterston B and Hinkley Point B are planned to be decommissioned, they will have been operational for 46 years. Designed with safety improvements in mind over Gen I reactors, they were also intended to produce more power with less of a focus on producing fissile material for weapons. Unfortunately, this generation of reactors had the worst track record of any with safety, with Three Mile Island, Fukushima and Chernobyl all being Gen II designs.

Gen II+ reactors have been built recently with 18 CPR-1000s being constructed over the last decade in China. The UK is currently looking towards Gen III+ reactors while Gen IV proposals are being researched.

1.2.4 Generation III Reactors

There are no Gen III reactors operating in the UK. The first was built in Japan in 1996 and the type of reactor installed was an Advanced Boiling Water Reactor (ABWR). Bradwell B is a proposed site for the Gen III Hualong One type PWR power plant (HPR1000), and if this goes ahead it would have an expected opening date between 2030-2035.

This generation of reactor was designed to exceed the service life of the previous generation, and generate power for at least 60 years. The safety features have been improved and include increased resilience to external factors, including accidental or intentional aircraft collision into the plant[2].

Gen III+ reactors go further in adding passive safety features, such as natural circulation and gravity coolant feed [3]. Other aims of future generations are to standardise and simplify designs whilst improve the economy over Gen II+ and Gen III.

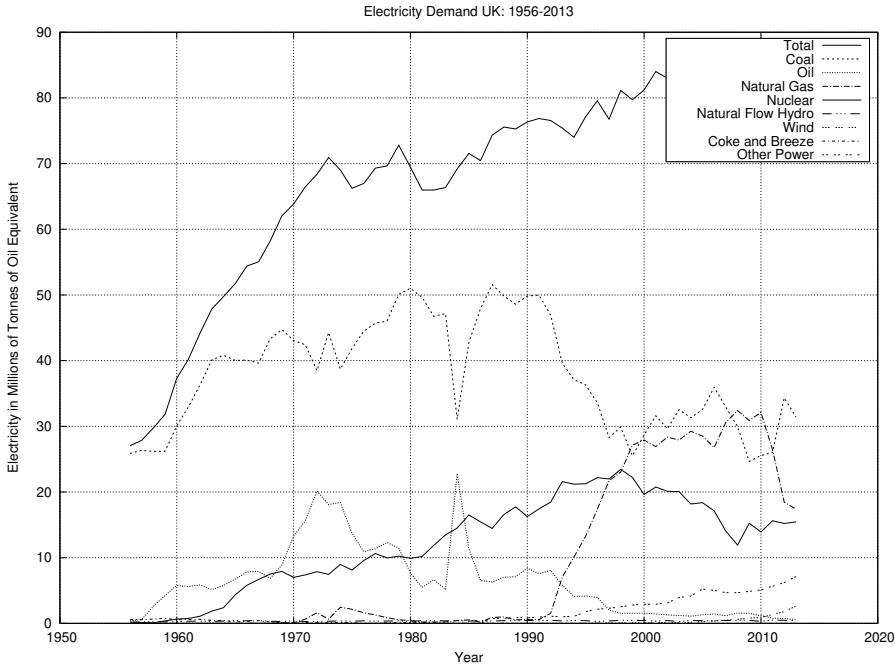


Figure 1.3: Electricity in Millions of Tonnes of Oil Equivalent

1.3 An Approaching Energy Gap for the UK

Since Calder Hall, the first commercial nuclear power plant, opened in 1956, the demand on electrical power generation in the UK has tripled (fig 1.3). There is now a reliance on cheap and clean power from nuclear reactors as these provide a quarter of our electricity. There are sixteen reactors operational in the UK: the Magnox reactor at Wylfa and the fourteen AGR reactors are due to be decommissioned by 2023[4], and the remaining PWR reactor, Sizewell B, is expected to remain operational until 2035[4].

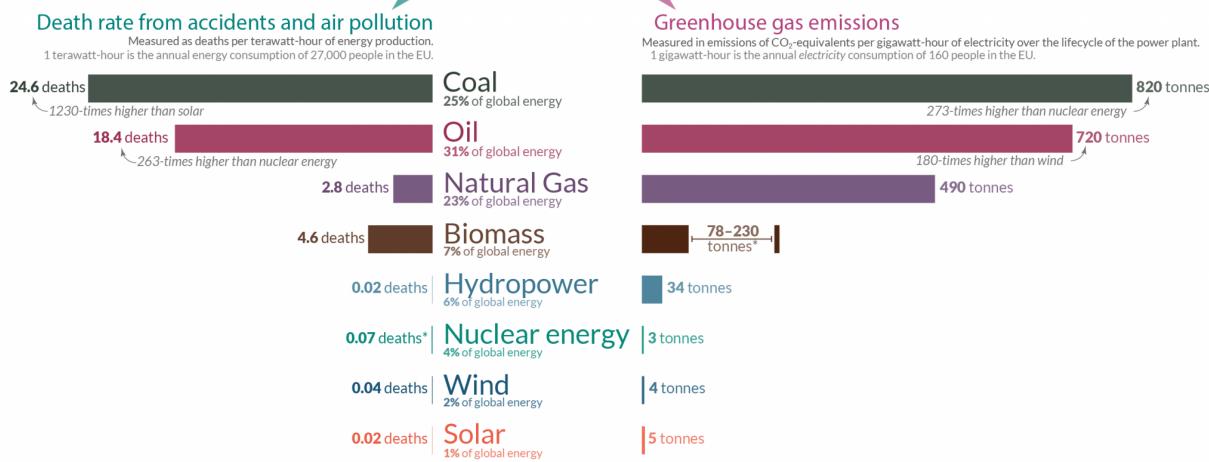
There is an obvious concern that within the next ten years the UK will lose a sizeable proportion of its electricity generation capabilities. There are increasing pressures on countries to reduce their carbon dioxide output, so the gap created by ageing nuclear plants and coal power needs to be filled.

1.3.1 Why choose Nuclear Power?

As a civilization we need energy, whether it's in the form of electricity or stored chemically, and whilst we are becoming more efficient at using that energy, the demand for it will remain (unless something drastically changes our society). There is a choice between burning fossil fuels or bio fuels, using energy from the Sun, wind, ground, oceans or rivers and finally using the energy of the nucleus, whether by fission or fusion.

Each has its drawbacks and advantages. Renewable energies are unreliable; wind power only provides energy when there is a wind, and if the wind is too strong, they must be shut down or risk damage. The Sun is only available for a portion of the day, and the duration and intensity change with the seasons, not to mention the impact of clouds on solar power. Renewable sources are not very good at responding to demand; there isn't a button to turn up the power of the Sun, or increase the velocity of the wind when the national grid demands it. If we were completely reliant on renewable energies, there would either need to be an efficient way to store energy on a large scale, or many more solar, wind, tidal and hydroelectric power stations than would be needed to produce excess energy. Whilst the energy source may be free, turbines, solar panels, dams and so on require maintenance, so an excess of these would be wasteful and costly.

What are the safest and cleanest sources of energy?



Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

Figure 1.4: Safest and cleanest sources of energy [7] - deaths per terawatt hour and tonnes of CO₂ per gigawatt hour

Fossil fuels release carbon dioxide, sulphur dioxide, naturally occurring trace radioactive elements and other pollutants into the atmosphere. These power plants are useful because the fuel is cheap, but the waste is put back into the environment without much processing and the energy produce may be varied to meet the demand of the national grid.

Nuclear power has a bad reputation with the public, in part caused by accidents such as the Windscale fire, Three Mile Island, Chernobyl, and Fukushima. There have been examples of poor handling of nuclear waste in the past, such as the legacy storage ponds at Sellafield, and long term storage in geologically stable areas is something that hasn't been achieved to store the existing waste, let alone waste created in the future.

Modern nuclear power plants are very expensive to build, with the proposed 3.2GWe Sizewell C power plant expected to cost 20 billion or more [5]. When compared to £0.5 billion for the 0.884GWe Carrington CCGT Power Station [6], the initial costs are £0.57 billion per GWe for CCGT compared to £6.25 billion per GWe for Nuclear.

There is an effort around the world of countries aiming to reduce the production of carbon dioxide as a result of burning fossil fuels. When a power plant is operational, the carbon dioxide produced by nuclear power is negligible when compared with gas, oil and coal power plants.

Despite its reputation amongst the public, nuclear is much safer than fossil fuels and is on par with renewable energies. The pollution caused by fossil fuels affects us all and is responsible for many deaths each year, causing more misery than nuclear power ever has (fig. 1.4). Coal and oil are responsible for 350 and 260 times as many deaths as nuclear.

As a species, we have used combustion of chemicals to generate heat for thousands of years, culminating in high efficiency CCGT power stations, with efficiencies above 60%. We are at the beginning of our journey with nuclear power, with many new and innovative designs proposed, where safety to plant workers and the general public are a priority.

Future designs may help to tackle the problematic waste generated in the past, and by using fertile material to produce fissionable elements other than U²³⁵, the amount of fuel available to reactors will increase.

1.3.2 Planned and Under Construction in the UK

Several sites have been acquired with the aim of building new nuclear power stations in the UK. There are five sites and three reactor designs[8]:

- Hinkley Point: two Areva EPRs (EDF Energy), Hinkley Point C under construction
- Sizewell: two Areva EPRs (EDF Energy)
- Wylfa: 2-3 Hitachi ABWRs (Horizon Nuclear Power)
- Oldbury: 2-3 Hitachi ABWRs (Horizon Nuclear Power)
- Sellafield: 3 Westinghouse AP1000s (NuGeneration) ...

Two Gen III+ reactors are under construction in the UK: Hinkley Point C 1 and C 2. These reactors are Areva NP designed Gen III+ EPRs, which are PWRs, and the proposed opening years are 2025 and 2026 respectively.

There are also plans to construct a third plant at Sizewell that, if accepted, will be an EPR reactor. Sizewell C will join the decommissioned Sizewell A and operational Sizewell B, that may operate until 2055.

1.3.3 Proposed Generation III+ Nuclear Power Plant Designs

Areva EPR

The 1.6GW Areva EPR design has four primary loops transferring heat by pressurised water from the reactor to heat exchangers. It requires U enriched to 5% 235U in the form of U Oxide Pellets. The inlet temperature is 568.75K and the outlet temperature is 602.95K.

There are 241 fuels assemblies, each containing 265 fuel rods giving a total of 63865 rods. The fuel rod cladding is made from 316 stainless steel and has an inner diameter of 7.72mm and an outer diameter of 9.68mm. The materials used to construct the control rod drive mechanisms includes 410 stainless steel and 304 stainless steel.

There are two EPR reactors planned for the proposed Sizewell C power plant.

Westinghouse AP1000

The 1.1GW Westinghouse AP1000 design is a pressurised water reactor with two primary loops transferring heat from the reactors to heat exchangers. An enriched UO_2 fuel, up to 5% 235U, is clad in Zirlo (a proprietary Zr alloy).

Zirlo is an alternative cladding material to 316 stainless steel, and as a cladding material is absorbs fewer neutrons than steel cladding.

Property	316SS Fe 20Cr 8Ni 1Mo	Zirlo (Hill Approximation)
Bulk Modulus (GPa)	164.9 [9]	98.5 [10]
Shear Modulus (GPa)	74.6 [9]	33.2 [10]
Young's Modulus (GPa)	194.3 [9]	89.7 [10]
Poisson's Ratio	0.30 [9]	0.35 [10]

Table 1.1: Bulk, shear, Young's modulus comparison: Zirlo and austenitic stainless steel

The inlet temperature is 552.55K and the outlet temperature is 597.85K which is close to the temperature range of the EPR. The composition of the control rod absorber material includes 304 stainless steel.

1.3.4 Generation IV

Goals of Generation IV Reactors

The GenIV International Forum has put forward four main goals for this next generation of nuclear power[11]:

- sustainability
- safety and reliability
- economics
- proliferation resistance and physical protection

The selection of known materials, and the development of new materials, will play a key part in all four of these goals.

Carnot's Theorem

Whatever the motivation, whether it is to increase profits or to supply energy at greater amounts and for less cost to the consumer, increasing the efficiency of usefully using energy is critical. Solar panels are continually being improved to edge closer to their theoretical limit and wind turbines are being constructed larger and with more advanced materials to extract as much energy from the wind as possible.

In the same way, power plants that use heat are constantly being developed to improve efficiency. In the nineteenth century Carnot showed that the maximum possible efficiency of a heat engine is determined by the difference in temperature between the heat reservoirs.[12]

$$\eta_{max} = 1 - \frac{T_c}{T_h} \quad (1.1)$$

To increase maximum efficiency, the temperature difference should be increased, and this leads to higher temperature reactors. There will be a trade off between the increased temperature, the ability of the materials to withstand the temperature, health and safety considerations, the lifetime of components, the effect on the coolant and more.

The first generation of reactors in the UK were the gas cooled Magnox reactors. With core temperatures of approximately $620K$ [13], the thermodynamic efficiency was relatively poor. This was limited by the MgO cladding, which was in turn selected due to the fuel. Combined cycle gas turbines have much higher temperatures within their turbines, and the temperature of the steam within the secondary steam turbine can reach $850K$ [14], significantly higher than in a Magnox reactor.

The second generation of power plants, particularly in the UK, included the advanced gas reactor that used C_2 as a coolant. This reactor design increased temperatures to over $870K$, and thus the maximum possible thermodynamic efficiency was increased.

Several Gen IV reactors have designed operating temperatures that exceed those of the AGR, and this brings a new set of challenges to overcome.

Fast Reactors

Examples of thermal neutron reactors include Magnox, AGR, LWR and Candu reactors. These reactors contain moderators, designed to slow neutrons, decreasing their energy to thermal temperatures and leveraging the large neutron fission cross section of U-235. The cross section for fast neutrons and U-235 is in the region of 1 barn, but thermal neutrons and U-235 have a fission cross section of over 1,000 barns.

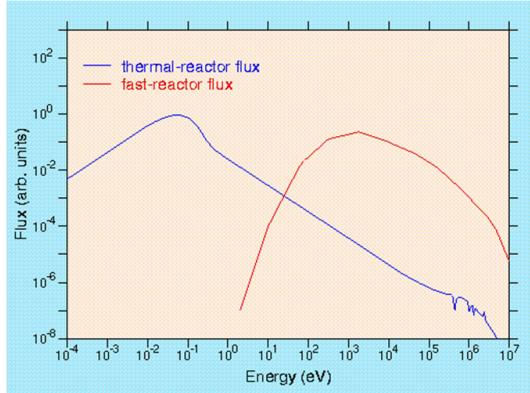


Figure 1.5: Typical Reactor Flux - Thermal and Fast Spectra

Natural U contains 0.7% U-235 and 99.3% U-238[15]. The thermal fission cross section for U-238 is small for thermal neutrons, and can be measured in the microbarn to millibarn range, but it has a much higher fission cross section with fast neutrons with a cross section closer to 1 barn for neutrons with an energy of 1MeV or more. The neutron flux in a fast reactor is spread over a higher energy band than that of a thermal neutron reactor (fig. 1.5).

Fast neutron reactors have been tested and used since the 1950s, and there are several Gen IV reactor designs based on this approach. Given the much larger percentage of U238 in natural U, Fast Neutron Reactors have a larger stockpile of fuel. They also breed fissile isotopes, Pu 239 and 241. This has its advantages and disadvantages. The advantage is clear, as it produces fissile fuel as it runs, but the creation of fissile materials is a concern as these isotopes could be used in nuclear weapons.

Lead-cooled Fast Reactors

Light elements such as the hydrogen in water molecules, or carbon in a graphite stack, allow neutrons to efficiently lose energy. When neutrons scatter inelastically with heavier elements, such as lead, they lose a much smaller percentage of their energy.

The Lead-Cooled Fast Reactor (LFR) uses lead as the coolant. Lead has a low reaction cross section and, because of its mass relative to a neutron, the kinetic energy lost in collisions is low, keeping more neutrons in the fast energy group. As the coolant is a liquid, and the boiling point of lead is over 1,970K, the system will run at atmospheric pressures and any problems associated with void formation due to boiling is removed. The molten lead will act as a gamma shield and it will be chemically less reactive than the coolant in a Sodium-Cooled Fast Reactor (SFR) or Supercritical-Water-Cooled Reactor (SCWR).

There are several disadvantages to using lead as a coolant. The melting point of lead is 600K, so the reactor would need to be heated first for the coolant to become a liquid. This also restricts the lowest temperature and thus the maximum efficiency possible that may be extracted from the system without the coolant forming a solid. The density of lead also poses a problem for the structure needed to support the reactor.

European Lead Fast Reactor (ELSY) is a 600MWe Lead-Bismuth eutectic cooled fast neutron reactor. It has a lower melting point than lead, but there are concerns for the transmutation of Bismuth to Polonium. The 9,000

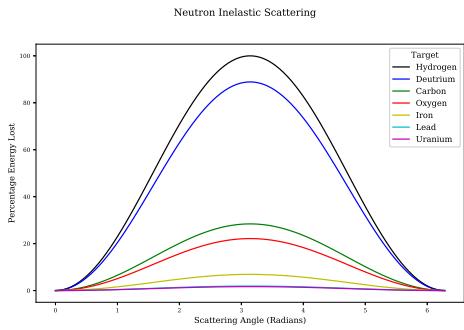


Figure 1.6: Inelastic scattering - scattering angle vs energy loss for a range of target atoms

tonne molten lead coolant and the high flux of fast neutrons[16] will be challenging conditions to overcome.

VHTR

Traditional nuclear power plants, as well as coal and the secondary cycle of CCGT plants, boil water to drive turbines. Very-High-Temperature Reactor (VHTR) designs use thermal neutrons and a fissile fuels. Designs plan to have outlet temperatures of up to 1,270K[17]. With a helium coolant, there are several options.

- directly drive turbines
- heat water to create steam and drive turbines
- use the high temperature to help create hydrogen, and combine with steam driven turbine

Hydrogen may be extracted from high temperature water either by a thermo-chemical, high temperature electrolysis or hybrid process[18], and they all benefit from the very high temperatures provided by the predicted outlet temperatures (fig 1.7).

The Pebble-Bed Reactor (PBR) is a type of VHTR where spherical fuel pellets are held in a hopper shaped reactor. Temperatures within the core may reach 1,770K, and this will pose a challenge for materials scientists.

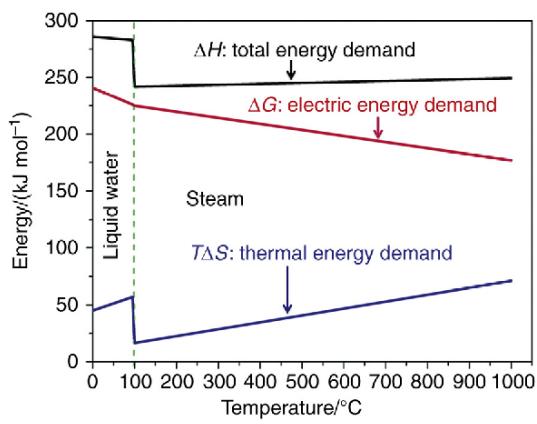


Figure 1.7: Energy demand to produce hydrogen by electrolysis as temperature changes

GFR

The Gas-Cooled Fast Reactor (GFR) design is similar to the VHTR, but rather than use thermal neutrons, it will use fast neutrons. The design will use a ceramic core and have outlet temperatures up to $1120K$. This will give similar options to the VHTR where high temperatures would be used to assist in the production of hydrogen, or it would be used to power a turbine with a secondary steam powered circuit similar to that of a CCGT.

There are challenges ahead, including the development and testing of advanced materials that can withstand these temperatures while under high energy neutron flux. However, unlike LFRs, SFRs and SCWRs, the coolant is chemically inert and isn't transmuted by neutrons removing the risk of the coolant becoming radioactive.

SFR

Memories of small amounts of sodium, kept under oil in school chemistry labs, being carefully cut and prepared ahead of a violent reaction with water, probably come to mind at the first mention of an Sodium-Cooled Fast Reactor (SFR). It may not be the first choice of an element to use as a coolant, but it does have advantages.

Sodium has a low melting point, of just under $370K$, and a boiling point of $1,150$. With an operating temperature of up to $820K$, the reactor would be able to run at atmospheric pressure and without the issues associated with the coolant boiling. Its density is $0.971gcm^{-3}$. Comparing this to the LFR the coolant would be 12 times less dense and this would have advantages when designing the structure.

There have been several SFRs built and operated using this technology, such as the Russian BN-600 reactor. This particular reactor has a sodium primary loop which heats a secondary sodium loop which in turn heats a third loop for water and steam.

SCWR

At $647K$ and a pressure of $22.1MPa$ (approximately 218 atmospheres), water becomes supercritical (fig. 1.8) which is neither a liquid or a gas[19]. The current generation PWR and BWR operate at approximately $501 - 598K$ at $15.2MPa$ [8] and $551 - 560K$ at $7.1MPa$ [20] respectively.

Supercritical water exists above $647K$ and $22.1MPa$, and in this state water has a higher thermodynamic efficiency. The design of the nuclear power plant is also simplified as there is no phase change of the water, so a condenser is not needed. The Supercritical-Water-Cooled Reactor (SCWR) is the only Gen IV reactor design that uses water as the coolant[4]. The economic benefits have already been seen in SCW fossil fuel power stations, and it is incorporated in Gen IV water cooled fast and thermal reactors.

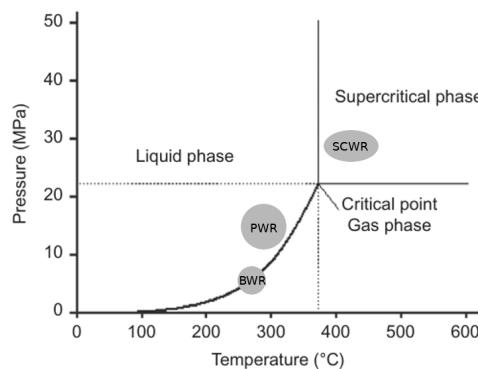


Figure 1.8: Phases of Water: Pressure vs Temperature

There are side effects to using supercritical water. It is an oxidant and is now being considered to “burn” organic waste in future waste processing plants. The combination of supercritical water chemistry and irradiation damage must be considered, as well as higher temperature and pressure, when considering the materials used by a SCWR.

Molten Salt Reactor

MSRs have been operated for over 50 years. Chemically, salts are very stable, and this has obvious safety benefits over reactors like the SFR. The coolants are fluoride salts that have high boiling points which gives the added safety protection of being able to operate at atmospheric pressure, unlike a PWR where a pressure vessel is needed.

The fuel is dissolved into the molten salt. There are no solid fuel rods to place into the core, remove or reprocess. The molten salt is processed on the site to remove poisons, waste and add new fuel.

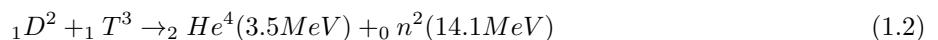
MSRs may use either thermal or fast neutrons and a range of fuels including Th. This is particularly interesting as Th is three times as abundant as U and would increase the fuel available to us to use to generate nuclear power.

A safety feature that takes advantage of the fuel being dissolved in the molten salt are large drain tanks under the reactor. A plug is designed to melt if it reaches a certain temperature and this would quickly drain the molten salt from the reactor into large and cold drainage tanks. The reactivity would drop and the fuel would cool reverting it to a solid salt.

Salts considered in various designs are chlorides, nitrates and fluorides. Corrosion of the reactor due to the molten salts is a concern. Elements that provide protection to corrosion, such as Cr, are prone to dissolution into the molten salt[21]. Where the fuel is suspended in the molten salt as Tristructural Isotropic (TRISO) fuel particles, formation of chromium-carbide precipitates may be an issue.

Experimental Fusion Reactors

Nuclear Fusion is a very attractive technology and could be the answer to all of our energy problems. Much work is being invested in developing this technology and the International Thermonuclear Experimental Reactor (ITER) has been designed to output more energy than is required to start the fusion reaction. The process of fusion combines two isotopes of hydrogen and leaves helium and fast neutrons (eq. 1.2). As neutrons have no charge, they can penetrate shielding causing damage as they lose energy through nuclear interactions. Any atoms they interact with have a chance to capture the neutron and become unstable.



The fast neutron spectrum for fission reactors ranges from a few eV to a few MeV, whereas the neutrons in a fusion reaction have 2-3 times more energy than the most energetic neutrons from fast fission. Engineers must develop materials to construct components that will be resilient to this damage, while having a low reaction cross section and being able to withstand other extreme conditions within the reactor.

Unfortunately, nuclear fusion as a commercial method of creating energy, is often said to be 30 years away, and this has been the case for some time now. It is a lot to ask, recreating a process that occurs within a star, in a safe and steady manner. Once this problem is solved, it could mark a turning point for human civilization.

1.4 Damage to Nuclear Core Components by Neutrons

Radiation damage in a reactor is caused by several projectiles. Fission fragments are heavy, contain charged particles and lose energy close to their source. Gamma rays excite electrons, promoting them to higher energy levels, or ionise the atoms where there is enough energy to eject electrons completely from atoms. They do not have the momentum to knock atoms out of place. Fast neutrons, however, may impart a great deal of momentum to a target atom and, as they are neutral, travel much further into a material than fission fragments. With enough energy, neutrons cause large damage cascades within the material.

One fuel source for many of the Gen III+ and Gen IV reactors is U-235, whether as enriched U or otherwise. The neutron(s) released by the fission of U-235 atoms have a spectra of energy (fig. 1.9) which may roughly be split into four categories: cold (below 0.025eV), thermal (0.025eV), slow and intermediate (above 0.025eV and below 1MeV) and fast (1MeV and above).

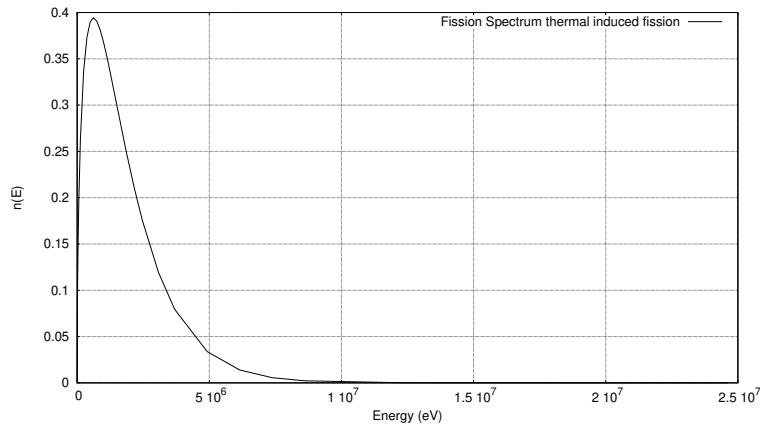


Figure 1.9: Neutron Energy Spectrum from Fission[22]

Thermal and slow/intermediate neutrons cause damage in their own particular way, as they are captured by and transmute the atoms within the target material. Fast neutrons have enough energy to create a great deal of damage, whereby they transfer kinetic energy to an atom in the material that become the primary knock-on atom (PKA) in a cascade of atoms through the material. Neutrons can travel deep into a material, creating many damage cascades through the material until they lose enough energy or pass through the material completely.

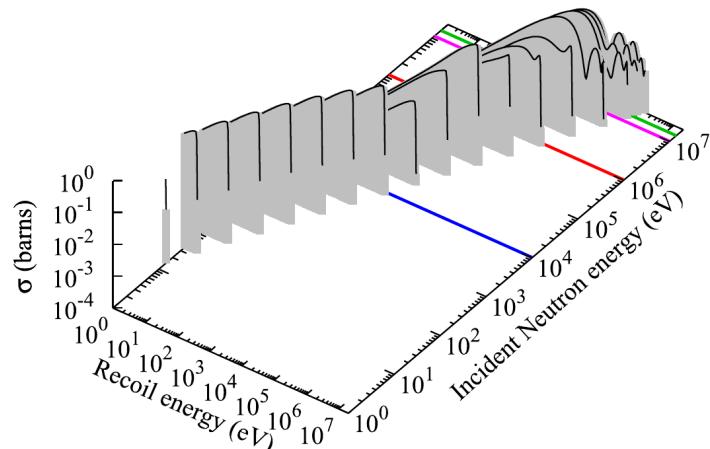


Figure 1.10: PKA energy as a function of neutron energy in Fe[23]

The intensity of neutron energies from the fission of U235 peaks around 1MeV (fig. 1.9). The energy transferred

to PKAs will vary depending on the mass of the target nucleus and scattering angle but, in Fe at least, 1MeV neutrons will create the majority of its Fe PKAs with energies ranging from 1keV to 100keV (fig. 1.10).

1.5 Radiation Damage: Replacing Neutrons with Protons

It is difficult to generate large fluxes of neutrons. When we pick up a cup, press keys on a keyboard, they react because of the electromagnetic force. Neutrons have no net charge, so they cannot be controlled in a similar way. Nuclear reactors are the most common way to create large fluxes of neutrons.

This is an expensive method, and one that may be inconvenient as the sample being tested would need to be placed inside a reactor. Rather than do this, ions may be used to replicate the damage caused by neutrons, but in a more controlled and accessible way.

A major side effect from the process of nuclear fission is the creation of both radioactive fission fragments and radioactive isotopes within the components and structural material of the reactor. Low energy protons are not captured as low energy neutrons would be, due to the opposing electromagnetic force between the proton and nucleus of the target atom. Once the proton energies exceed a few MeV, they have sufficient energy to transmute target nuclei.

Thanks to detailed reaction cross section data files and decay data, the amount of radioactivity induced by both protons and neutrons can be calculated. This will assist in the selection of proton beam parameters to replicate irradiation damage, to the required DPA, whilst keeping the radioactivity at a safe level.

1.6 Simulation Material Damage

Irradiating and testing materials irradiated by both neutrons and protons have a number of drawbacks, including expensive facilities and the creation of radioactive waste. Simulations may be used to help understand the processes on a mesoscopic scale and make predictions on how the material in question may behave inside a reactor under certain specified conditions.

Part of this work focuses on the damage of stainless steel by neutrons. The damage process occurs on time scales and sizes too small to capture experimentally and play back to study. The theories may not be sufficiently well understood, or are often unsolvable with current techniques. Modelling helps bridge the gap between theory and experiment.

1.6.1 Molecular Dynamics

Molecular dynamics models have been used to simulate radiation damage. Damage cascades caused by primary knock-on atom (PKA)s with energies of 10, 20 and 50KeV have been modelled in BCC W[24], whilst cascades caused by PKAs of 5, 10 and 20KeV have been simulated in BCC Fe[25].

Neutrons have a long mean path before interacting with a target when compared to ions. It would be impossible with current and foreseeable technology to simulate a neutron travelling through a material because the volume of the material, and the number of atoms and interactions between atoms, would be so large. Rather than attempt this, it is more productive to focus on individual damage cascades caused by PKAs in order to conserve computing resources.

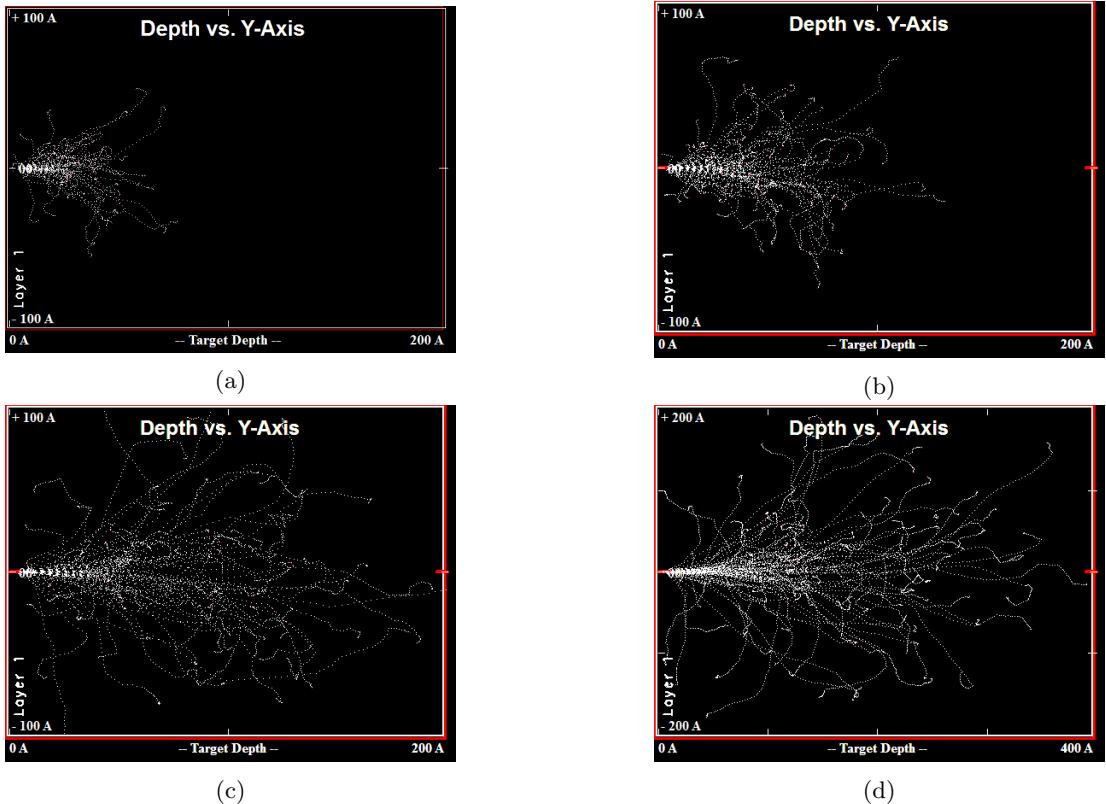


Figure 1.12: (a) 5KeV Fe PKA Damage Cascade in Fe (b) 10KeV Fe PKA Damage Cascade in Fe (c) 20KeV Fe PKA Damage Cascade in Fe (d) 50KeV Fe PKA Damage Cascade in Fe

Molecular Dynamics simulations require interatomic potentials to govern how the atoms in the simulation interact with one another. Typical simulation sizes range from thousands of atoms to millions of atoms.

Fe has a density of approximately 8.5×10^{-2} atoms per cubic angstrom, and this leads to simulation sizes between almost one million atoms and fifty million atoms for cascades with PKAs ranging from 5KeV to 50KeV (fig 1.12, table 1.2).

PKA Energy (KeV)	Volume (Ang^3)	Atoms
5	1.0×10^6	9.0×10^5
10	6.0×10^6	5.4×10^6
20	8.0×10^6	6.9×10^6
50	6.4×10^7	5.5×10^7

Table 1.2: Approximate simulation sizes by PKA energy

The interatomic potentials used to simulate a damage cascade must be able to model the very close separations during the collisions event. For very close separations, an exponential potential such as the Ziegler-Biersack-Littmark (ZBL) potential may be used, and for other separations a standard potential such as the embedded-atom method (EAM) may be used, in particular for metals.

There have been attempts to create potentials based on Physics, but the nuances are still poorly understood and there aren't any models that can be applied accurately to any material type under any set of circumstances. A number of recent interatomic potentials, including a selection of transition metals, have been fit to match the experimental data rather than the underlying physics.

1.6.2 Density Functional Theory

In quantum mechanics, the Schrödinger equation and the wavefunction of a system describes all that can be known about that quantum system. Unfortunately, it is very difficult to compute when there are just several electrons in the system. Many atoms in a system makes this an impossible task with current technology.

Density Functional Theory (DFT) replaces the many electron wavefunction with a single electron wavefunction. Rather than use the positions of all the electrons, which would result in 3^n parameters, the charge density, which varies in space with just 3 parameters, is used.

DFT allows for the calculation of the forces between atoms and the energy of that collection of atoms by replacing the many parameters to give the coordinates of many electrons with the three coordinates for the charge density. From the forces the optimum (relaxed) basis vector and atom positions may be computed, and from the energy of multiple configurations other properties are computable (bulk modulus, elastic constants, etc).

1.6.3 Interatomic Potentials for Molecular Dynamics

Interatomic potentials help to describe the interaction between atoms. Originally they were used for pairs of atoms and a collection of atoms on a pair by pair basis, but in the 1980s the complexity of potentials increased as they started to include the effects of the background electron density that the atoms are embedded in. This lead to the Finnis-Sinclair (FS) and embedded-atom method (EAM) type potentials.

Whilst DFT is a purer form of computational materials science, more closely linked to physics, it is computationally intensive and suitable for small systems of hundreds to thousands of atoms. By using interatomic potentials to recreate the forces and energies that would be computed by DFT, the systems may be increased in size to thousands to millions of atoms, and these calculations may be run over a time frame with hundreds or thousands of time steps.

There are a wide range of potentials, and these are typically selected based on the type of material and the required accuracy. Simple pair potentials may still be used, but angularly dependent potentials have been developed for materials with covalent bonds, and metals use FS and EAM potentials. The potentials are made up of a set of functions, and these functions change from derivation to derivation.

Chapter 2

Background: Ionizing Radiation

There are a wide range of radiation sources and types of ionising radiation. They interact differently with matter, depending on the type of radiation and its energy. Damage may be caused with the radiation transferring energy directly to the target. The target may be transmuted or the surrounding, for example the creation of Cobalt-60 in steel in the nuclear industry. The environment may be changed chemically, and one example of this is the radiolysis of water.

2.1 Radiation Types Relevant to This Work

2.1.1 Introduction

There are three types of radiation that are useful to discuss in this work, neutrons, ions and Ggammas, and two of these (neutrons and protons) are of particular interest in this work. Each of these are capable of ionizing the atoms they interact with, given sufficient energy. Higher energy and momentum radiation it is able to directly damage materials, or the DNA structure of organic life, that it interacts with. Lower momentum ionizing radiation may change the water chemistry of the environment or within the cells of organic lifeforms that is detrimental to life.

2.1.2 Protons and other Ions

Charged particles interact with matter through the Coulomb interaction. As a charged particle passes through matter, it may interact with both the nucleus and electrons of an atom. A sufficiently energetic ion may lose kinetic energy to electrons by either raising the electrons to higher energy levels in their atoms, or by removing electrons from atoms altogether, causing the ionization of those atoms.

Ions may also lose kinetic energy to the nucleus of an atom through elastic scattering and, where the atom is in a crystal structure, through knocking atoms in the material out of their lattice positions. Inelastic scattering is also a possibility, leaving the target nucleus in an excited state.

Knock on atoms and electrons with enough kinetic energy that have been removed from atoms (delta rays), continue the irradiation of the material while they have the kinetic energy available to do so.

A large proportion of the energy of a charged projectile is lost to the electrons of the target material. There is a chance, depending on the energy and type of charged projectile (and the cross section of the target nucleus), that the charged particle will overcome the coulomb potential and be captured by the nucleus.

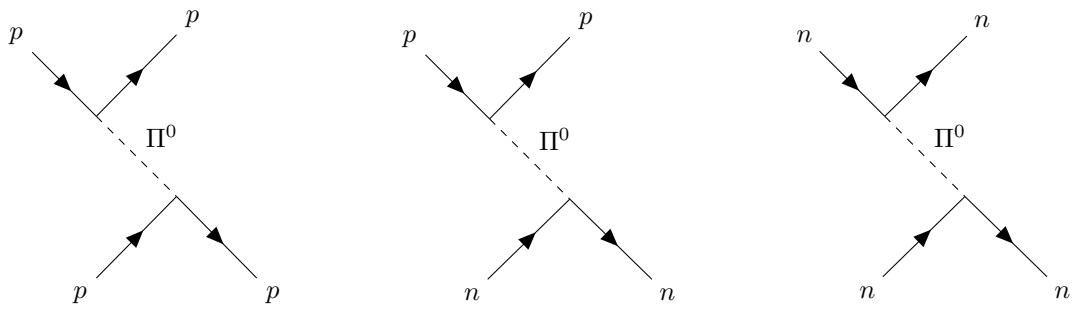


Figure 2.1: Direct interaction p-p [27] Figure 2.2: Direct interaction p-n [27] Figure 2.3: Direct interaction n-n [27]

This may result in a stable nucleus or an unstable nucleus. If it is unstable, there is a probability that it will decay releasing energy in the form of photons, nucleons or electrons. The initial ion irradiation creates sources of further radiation within the target material.

2.1.3 Neutrons

Neutrons interact with matter differently to that of protons, ions and other atoms, as the neutron has no overall charge. Neutrons do have a magnetic moment and experience a weak interaction with unpaired electrons by a dipole-dipole interaction. Neutrons also interact with the nucleus of atoms, and this is dependant upon the target atom and the energy of the projectile neutron. The energy of neutrons may be grouped as shown in table 2.1, and as mentioned earlier, thermal neutrons are of interest to current reactor designs, and fast neutrons are of interest to several future designs.

Name	Energy Range	Velocity/ms ⁻¹	Wavelength Ang
Cold	0-0.025 eV	$0.0 - 2.2 \times 10^3$	> 1.8
Thermal	0.025 eV	2.2×10^3	1.8
Epithermal	0.025-0.4 eV	$2.2 \times 10^3 - 8.8 \times 10^3$	0.5-1.8
Cadmium	0.4-0.6 eV	$8.8 \times 10^3 - 1.1 \times 10^4$	0.4-0.5
Epicadmium	0.6-1.0 eV	$1.1 \times 10^4 - 1.4 \times 10^4$	0.3-0.4
Slow	1-10 eV	$1.4 \times 10^4 - 4.4 \times 10^4$	0.09-0.3
Resonance	10-300 eV	$4.4 \times 10^4 - 2.4 \times 10^5$	0.02-0.09
Intermediate	300 eV - 1 MeV	2.4×10^5	$2.9 \times 10^{-4} - 0.02$
Fast	1-20 MeV	$1.4 \times 10^7 - 6.1 \times 10^7$	$6.5 \times 10^{-5} - 2.9 \times 10^{-4}$
Relativistic	> 20 MeV	$> 6.1 \times 10^7$	$< 6.5 \times 10^{-5}$

Table 2.1: Neutron Categories by Energy Range [26]

The nuclear force holds nuclei together by exchanging pions. It is 137 times stronger than the electromagnetic force, but because of the mass of the pion (approximately 130-140MeV) it only acts over a short range of approximately 1fm. The nuclear force is attractive but has a short-range repulsive core.

For nuclear reaction and damage calculations, the interaction of neutrons with matter is represented by a cross section (section 4.4.3). Cross sections are used to calculate the probability of a certain interaction occurring and are dependent upon the target nucleus, how much of the material the pass through and the energy of the neutron.

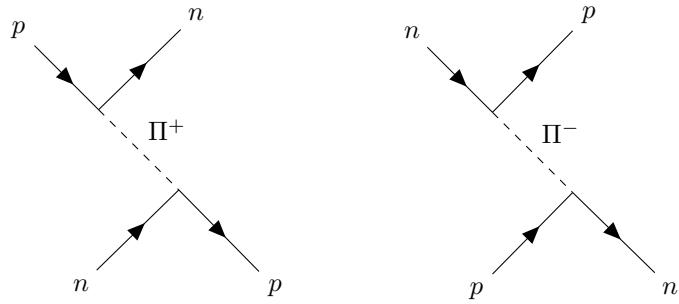


Figure 2.4: Exchange interaction n-n [27]

Figure 2.5: Exchange interaction p-n [27]

2.1.4 High Energy Photons

The electromagnetic spectrum classifies photons based on their energy and/or source, but visible light, x-rays, gamma rays and so on are all the same elementary “particle”. During the early years of Quantum Mechanics, the relationship between the energy and wavelength of a photon was discovered: the Planck-Einstein relation (eq. 2.1).

$$E = hf \quad (2.1)$$

Pair Production

The energy of photons in the database used for this work ranges from 1keV up to almost 10MeV. There are several ways high energy photons will interact with the atoms of a target material. The rest mass of an electron is 511keV. If the photon energy is greater than 1.02MeV, i.e. there is at least enough energy to create an electron and positron, there is a chance that the photon will create an electron-proton pair.

$$hf = (m_e + m_p)c^2 + T_e + T_p \quad (2.2)$$

The creation conserves energy and mass, with excess energy carried away as the kinetic energy of the particle pair. The charge before the creation is zero, as the photon is neutral, and the charge after is also zero, with the -1 of the electron and +1 of the positron cancelling out. Angular momentum is also conserved; the photon is a spin 1 Boson and, as electrons and positrons are Leptons they have half integer spin, adding up to 1. Finally, momentum is not conserved in a vacuum, and this is why pair production occurs in the coulomb field of a nucleus. The nucleus carries away excess momentum, fulfilling this conservation law (fig. 2.6).

Compton Scattering

An incident photon with enough energy may interact with the electron of an atom, and if it transfers enough kinetic energy it will eject the electron from the atom (fig. 2.7). A lower energy photon is also created that carries away the remainder of the energy, but also linear momentum, as both energy and momentum must be conserved.

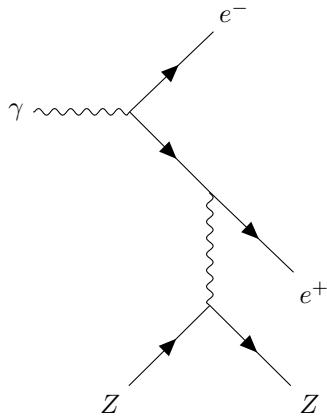


Figure 2.6: Pair production Feynman diagram

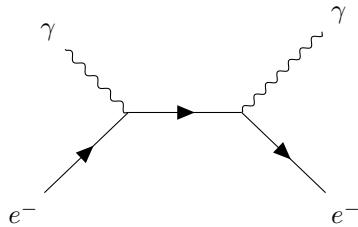


Figure 2.7: Compton Scattering Feynman diagram

Photoelectric Effect

Photons of sufficient energy interact with and eject electrons from the surface of a metal (fig. 2.8). At lower energies, the photons have no effect on the electrons. Ejected electrons have a maximum kinetic energy equal to the difference between the energy of the interacting photon and the workfunction, $T_{max} = h\nu - \phi$.

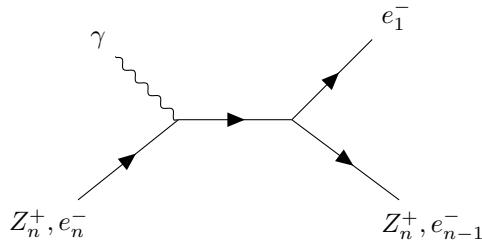


Figure 2.8: Photoelectric Effect Feynman diagram

Coherent Scattering

Lower energy, non-ionizing photons, have insufficient energy to eject electrons (fig. 2.9). Coherent scattering is the interaction in which photons change direction, scattering, without losing energy.

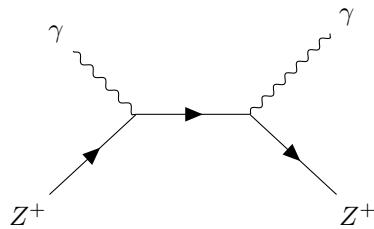


Figure 2.9: Coherent Scattering Feynman diagram

2.2 Effects of Radiation on Organics

2.2.1 Dose: A Multitude of Units and Definitions

There is a distinction between how radioactive a source is and how much radiation is absorbed by an organic due to that source. There are different measures of radioactivity and its effects, and they may be summarised very briefly as follows:

- Radioactivity of a source - decay events per second (Bq) (also measured in Curie, 3.7×10^{10} decays per second)
- Absorbed dose of any target - energy absorbed per unit mass (Gy) (equivalent of Joules per Kilo)
- Equivalent dose absorbed by tissue/organ - type of radiation is important (Si)
- Effective dose on tissue/organ - how the equivalent dose affects the specific tissue/organ (Si)

One joule of ionising radiation absorbed by a person may not seem a lot of energy, but the average annual doses are so small they are measured in millisieverts per year. A dose of just 5 Sieverts is enough to kill 50% of people within 1 month (table 2.2).

Exposure	Dose (mSv)
Dental X-ray	0.005
Chest X-ray	0.02
UK average annual dose	2.7
Whole body CT scan	10
Nuclear industry employee annual exposure	20
Acute radiation effects	1,000
50% lethal dose within 1 month	5,000

Table 2.2: Irradiation dose comparison in mSv

2.2.2 Levels of Radioactivity from Neutron Irradiated Components

In work by Rodenas and Verdu a MCNP5 simulated neutron flux environment and a thermal flux of $2.5 \times 10^7 n/cm^2 s$ a stainless steel cylinder (20mm diameter and 70mm depth) was irradiated for 10 hours[28]. The sample was composed of 65% Fe, 19% Cr, 2.0% Mn, 1.0% Si by weight as well as small amounts of C, Ti, P and S. After 10 minutes cooling, it was calculated to have an activity of over 1.0×10^7 Bq and, by several orders of magnitude, it was predominantly due to the creation of Mn^{56} .

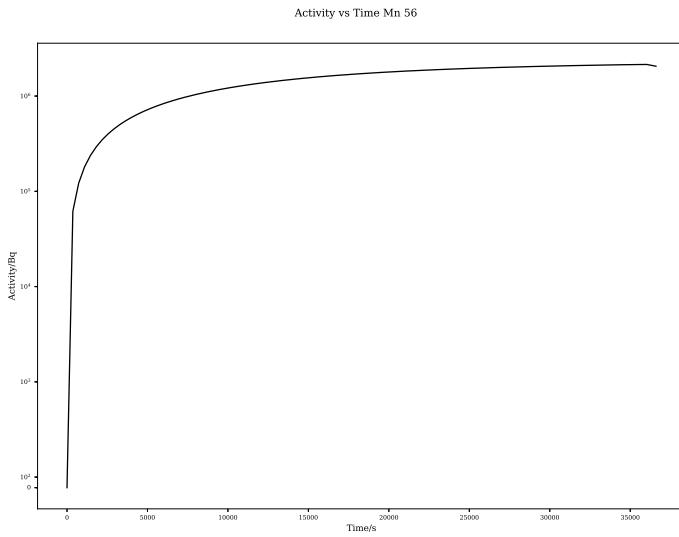


Figure 2.10: Predicted Mn 56 activity after neutron irradiation

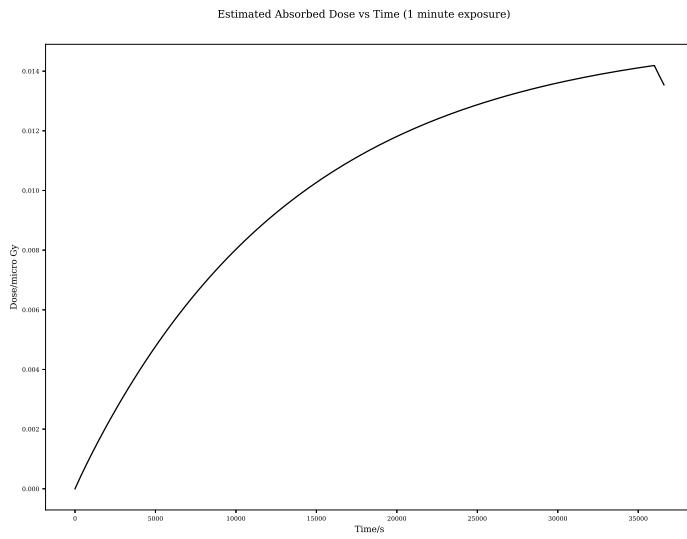


Figure 2.11: Steel Irradiation Gamma Dose 2.5×10^7 neutrons per cm squared per second

The maximum cross section of Mn^{55} (n, γ) Mn^{56} is 2.0 barns and the estimated reaction rate at this flux is just over 2.0×10^6 . As the half life of Mn^{56} is short compared to the irradiation time (a half life of 2.5 hours) the creation rate of Mn^{56} and decay rate equalise, saturating the sample with Mn^{56} .

A simple neutron activation code was developed, which will be discussed in more detail in this work, and the predicted Mn^{56} activity vs time was calculated (fig. 2.10). The overall gamma dose, for 1 minute of exposure to an 80kg person standing 2m from the source was also calculated (fig. 2.11). The calculated activity for Mn^{56} was less than that predicted by Rodenas and Verdu [28], but it was within an order of magnitude. It predicts a total activity after cooling of 2.3×10^6 Bq with almost 90% of this due to Mn^{56} .

The flux in the within the flux trap of the HFIR is much higher, with high energy neutrons having a flux of 1.0^{14} neutrons per cm squared per second (fig. 2.12). A second calculation was performed for this increased flux, also irradiated over 10 hours (fig. 2.13). The total activity increased from 2.3×10^6 Bq to 7.1×10^{12} Bq and the overall dose per minute was much higher, between 40 and 50 milligrey per minute over the first hour of cooling. These are dangerous levels of radiation.

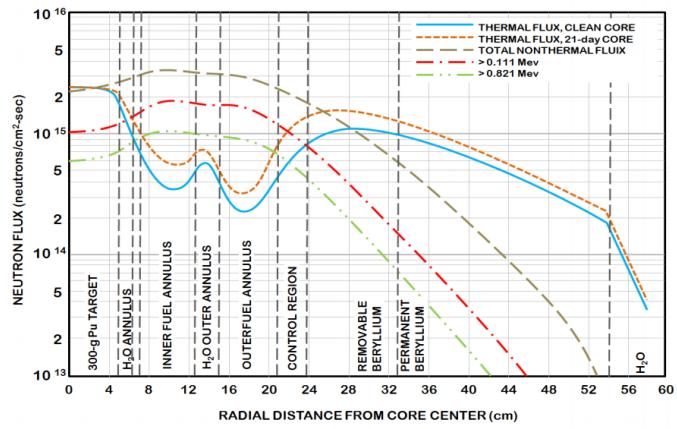


Figure 9: Neutron Flux at the MidPlane of HfIR (at 85MW)

Figure 2.12: High Flux Isotope Reactor Neutron Flux

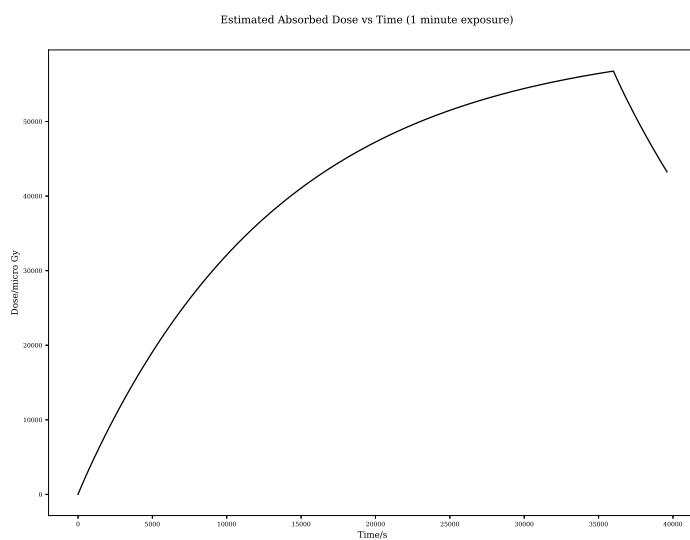


Figure 2.13: Steel Irradiation Gamma Dose 1.0×10^{14} neutrons per cm^2 squared per second

Damage to Organics by Radiation

Life evolved on Earth from single cell to multicelled organisms such as humans. Prokaryotic cells are single celled organisms that do not have a nucleus, an example being bacteria. Our cells are eukarytic cells, and these have a nucleus that contains the genetic information. Millions of cells die in our body every second, and our body replaces these by replicating living cells. During the replication stage, the DNA within the nucleus is copied.

DNA is a polymer that is carefully constructed from six components. Stretched out, it is approximately 2m in length and approximately 25 angstrom wide, and it is neatly contained within the cell nucleus which is on average just 6 microns across. The structure of DNA is the well known double helix. The sides of the helix are made from alternating sugars and phosphates, while the “rungs” of the ladder like structure are pairs of either thymine and adenine or cytosine and guanine. These pairs are covalently bonded to the sugar-phosphate sides, and by hydrogen bonds to each other.

Mitosis is the part of the cell cycle where the DNA within the nucleus is replicated and the cell splits into two. There are repair mechanisms, but if the DNA beyond repair, the cell may die through apoptosis, or it may replicate in an uncontrolled way leading to cancer.

Direct damage is caused when ionising radiation alters or destroys sections of DNA through a direct collision, for example a neutron colliding with and knocking out an atom in the DNA strand.

Our cells are predominantly water, so there is a higher probability of the radiation colliding with water molecules than DNA directly. The chemistry of the contents of the cell is changed, for example by the creation of free radicals, and these lead to damage of DNA as an indirect result of radiation passing through the cell.

2.3 Damage Cascades in Metals

When a metal is damaged by radiation, there are microscopic defects created and removed in a very short space of time. The result of many months or years of damage is complex and will be discussed in more detail in chapter 3.

The cohesive energy of atoms are measured in eV (table. 2.3), whereas the energy of incoming neutrons or fission fragments are measured in MeV. Radiation in the form of electrons, protons and heavy ions will lose energy to electrons in the target as they pass through. They may also lose energy to atoms within the lattice, knocking that atom out of place and creating a PKA. Due to its neutral charge and magnetic moment, neutrons have a very weak interaction with charged particles. They primarily lose energy directly with the nucleus of atoms within the target, also creating a PKA.

Element	Cohesive Energy/eV
Aluminium	-3.36
Iron	-4.32
Palladium	-3.91
Platinum	-5.77
Zirconium	-6.32

Table 2.3: Cohesive Energies [29]

The PKAs create a damage cascade, and their mean recoil energy is dependent upon the energy and type of radiation. After each cascade, there will be a period of time where the material quenches, with a recombination of some interstitials and vacancies. The PKA creation and cascade thermal spike cover a time range of 10^{-18} s

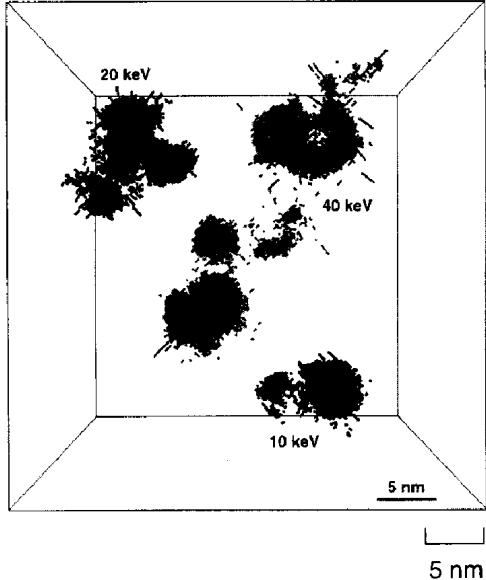


Figure 2.14: Interstitials in the cascades of 10keV, 20keV and 40keV displacement cascades [31]

to 10^{-13} s, with the quenching phase lasting in the region of 10^{-11} s. However, the cumulative effect of radiation damage on a component within a reactor core may only become apparent after months or years.

Projectile	Mean Recoil Energy
1 MeV Electrons	60eV
1 MeV protons	200eV
1 MeV heavy ions	5keV
1 MeV neutrons	35keV

Table 2.4: Average recoil energy of Nickel PKAs[30]

The immediate aftermath of interstitials created three damage cascades is shown in fig. ???. Damage cascades have been the subject of a number of studies and due to their short time scale and complexity, involving thousands to millions of atoms, are studied with computer simulations.

Chapter 3

Background: Austenitic Steels and Nuclear Power

Austenitic stainless steels have played an important role in the nuclear industry. They have a good resistance to corrosion and behave well at moderately high temperatures, having a good resistance to creep. They have been used within the primary circuit and secondary circuit of power stations, as cladding for fuel pellets, in the steam dryers, pre-heater tubing, core structurals, primary piping and more. Ferritic steel is magnetic and has a BCC structure, whereas austenitic steel is non-magnetic with a FCC structure, with its high Nickel content stabilising this phase. An area of concern is the susceptibility of austenitic stainless steels to IGSCC.

3.1 Stainless Steel

3.1.1 Introduction

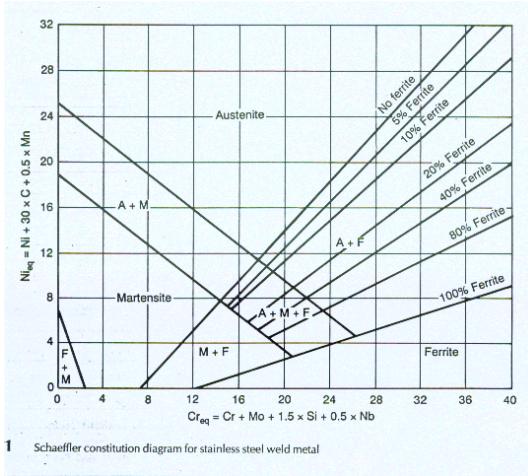
Stainless steel is a relatively new material, having first been developed and refined from the 1800s to the early 1900s, then being defined as a steel with at least 10.5% Cr in 1911. The addition of Cr to this level causes the formation of a passive protective layer of chromium oxide.

Fe is alloyed with Cr, Nickel and Carbon in varying quantities to make Stainless Steel. Depending on the application, other elements, such as Molybdenum, may be added to enhance the properties of the steel.

3.1.2 Grades of Stainless Steel

While the criteria that qualifies an alloy as Stainless Steel is containing Fe, Mg and at least 10.5% Cr, there are many grades that have variety of properties and atomic structures. The composition influences the structure, resistance to corrosion, yield strength, whether it is magnetic or not, and more.

Both Fe and Cr are BCC at standard temperatures and pressures, but adding austenite stabilisers such as Ni, Mn or C, the structure of the steel can be altered from BCC to FCC. Balancing the proportion of these elements changes the phase of the steel, and this may be represented as a Schaeffler diagram (fig. 3.1) or a ternary phase diagram (fig. 3.2).



1 Schaeffler constitution diagram for stainless steel weld metal

Figure 3.1: Steel Cr-Ni Schaeffler Diagram

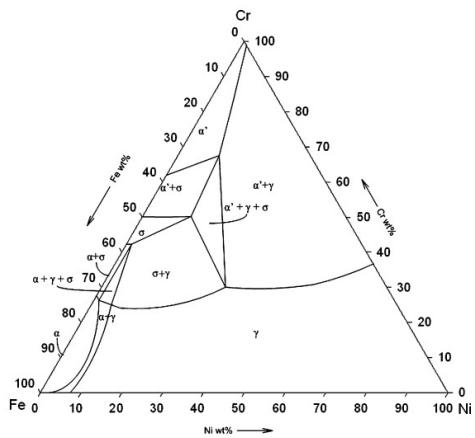


Figure 3.2: Iron-Chromium-Nickel Phase Diagram[32]

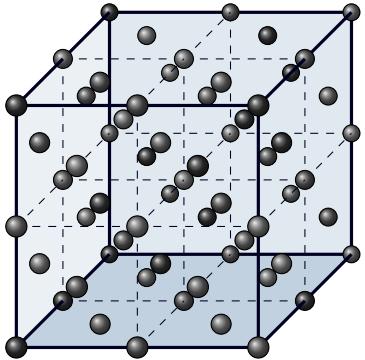


Figure 3.3: BCC structure of ferritic stainless steel

Ferritic Stainless Steel

At room temperatures and pressures, Fe exists in the alpha phase, which is a BCC crystal. It is energetically favourable for the magnetic moments of the Fe atoms to align with one another ferromagnetically. Ferritic stainless steels also have the natural alpha phase crystal structure of pure Fe at room temperature which is BCC (fig 3.3). These steels are magnetic and may be hardened by cold working. They are less corrosion resistant than austenitic stainless steels. Two common examples of this grade of steel are American Society of Engineers codes 405 and 430.

Austenitic Stainless Steel

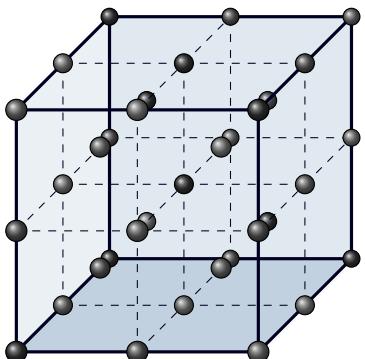


Figure 3.4: FCC structure of austenitic stainless steel

Austenite is a FCC allotrope of Fe (fig 3.4), and austenitic stainless steels are useful in many applications,

including a structural material for nuclear plant components, due to their resistance to corrosion. In addition to 10.5 wt% or more Cr they require an austenite stabilising element such as carbon and/or nickel to be added.

Two examples of such steels are ASME codes 304 and 316. Both have a high Cr content, in the region of 18-20%, which is in excess of the minimum passive film requirement of around 10-11%. The natural structure of such an Fe Cr alloy would be BCC, however 304 and 316 Steels contain Nickel (approximately 8% and 10% respectively) which is an austenite stabiliser, and this is responsible for the FCC structure of the steel. The 316 grade contains a minimum of 2% Mo to improve its resistance to corrosion.

Ferritic and martensitic have a better resistance to thermal or swelling shock than austenitic [33], but the corrosion resistance of austenitic steels is a deciding factor when choosing a steel for an application where corrosion resistance is important.

Martensitic Stainless Steel

Martensitic stainless steels have a FCC structure at high temperature, but when heat treated take on a BCC structure. This allows martensitic, unlike austenitic and ferritic, to be hardened by heat treatment. These steels are magnetic, they contain more than 10.5% Cr but have a much lower Ni than austenitic grades, if any. They are corrosion resistant, but the corrosion resistance of austenitic steel is typically better. Two examples of such steels are ASME codes 410 and 431. A comparison of properties between common grades of each steel are detailed in table 3.1.

Grade	Code	HRB	Proof strength (MPa)	UTS (MPa)	Density (kgm^3)	E (GPa)
Austenitic	304	80	230	540-750	7900	200
Austenitic	316	79	240	530-680	8000	200
Ferritic	405	75	250	400-600	7700	220
Ferritic	430	85	280	450-600	7700	220
Duplex	255	32	550	750-1000	7800	200
Martensitic	410	90	205	600	7700	215
Martensitic	420	95	345	700	7700	215

Table 3.1: Common grades of ferritic, austenitic, duplex and martensitic stainless steel and their properties. Hardness using the HRB, ultimate tensile strength (UTS) and Young's modulus E.

3.2 Austenitic Steels in Nuclear Reactors

3.2.1 The Use of Austenitic Steels in Reactors

Due to their excellent strength and corrosion resistance, austenitic steels have been widely used in nuclear reactors. The cheaper to manufacture 304SS and more corrosion resistant 316SS have been particularly popular choices of steel. There are drawbacks, when compared to other alloys including ferritic and martensitic steels, and two in particular are IGSCC and swelling.

3.2.2 Atom Damage Cascades

Fission fragments carry away the majority of the energy released during fission, with an approximate energy of 170MeV per ^{235}U fission event. The fragments are large and have a highly charged positive nucleus that stops rapidly due to the Coulomb interaction with the nuclei of surrounding atoms. Electrons released by beta decay travel a greater distance; they lose energy by interacting with electrons, but may also knock atoms out of place. Neutrons travel further still and lose energy by interacting directly with the nuclei of the surrounding material.

When an incoming projectile hits an atom (the primary knock-on atom (PKA)), it is knocked out of its place and loses energy by colliding with other atoms and to the electrons in the metal. Depending on the energy of the PKA, this causes a damage cascade. Atoms knocked out of their regular position in the crystal lattice leave vacancies behind. The displaced atoms either recombine with vacancies, become interstitial atoms or diffuse to defect sinks. A number of complex problems arise as a result of this damage, and this can take many years and high damage doses to appear.

3.2.3 Damage Rates

The safety of a Nuclear reactor is the primary concern; it must generate power, but it must be safe. It must also be cost efficient. Gen II PWR fuel assemblies were expected to withstand several DPA[34]. In BWRs the materials are designed to be irradiated by $10^{22} n/cm^2/s$, experiencing approximately 7 DPA over the lifetime of the components[35]. Components in PWRs are irradiated by up to $10^{22} n cm^{-2}s^{-1}$, and are expected to operate up to an irradiation dose of 70 dpa[35]. To remain cost efficient, components in Gen IV reactors will be expected to operate safely up to even higher doses or irradiation damage (fig 3.5). Components in sodium-cooled fast reactors will be expected to withstand up to 200 DPA over their lifetime to meet the requirements of cost effectiveness and durability[34].

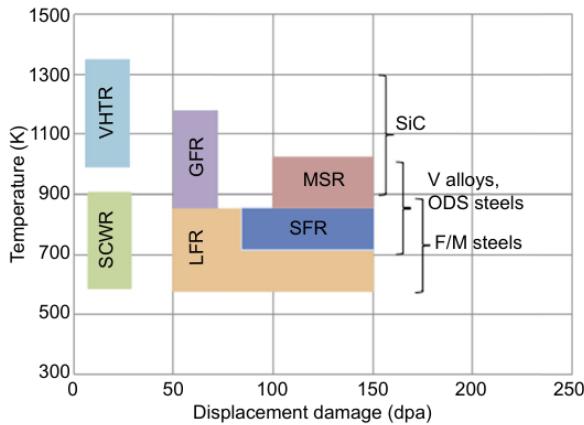


Figure 3.5: Expected DPA during component life time and operating temperatures[34]

Neutrons released during the fission of U235 range from 0 to 14 MeV (fig 1.9), and the energy transferred to a target atom depends on the scattering angle of the neutron and the mass of the target nucleus.

$$\alpha = \left(\frac{m_N - 1}{m_N + 1} \right)^2$$

$$E_N = E_n \left(1 - \frac{(1 + \alpha) + (1 - \alpha) \cos\theta_C}{2} \right) \quad (3.1)$$

In a collision, the amount of energy transferred to a target atom will depend on both the scattering angle of neutron recoiling from the target and the mass of the target nucleus. If the neutron does not change direction, a small amount of energy will be transferred, but if it bounces back at 180 degrees it will transfer the maximum amount of energy. A neutron will lose more energy per collision with light atoms (hydrogen, helium, carbon), but much less energy per collision with larger atoms (iron, molybdenum, lead).

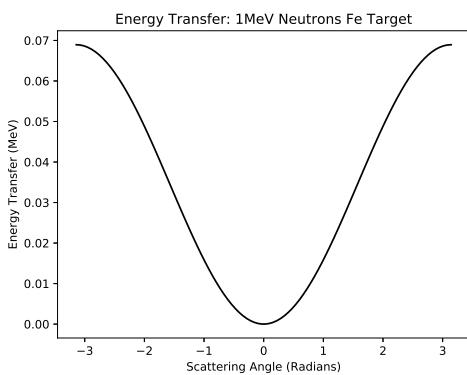


Figure 3.6: Scattering angle probability of neutrons

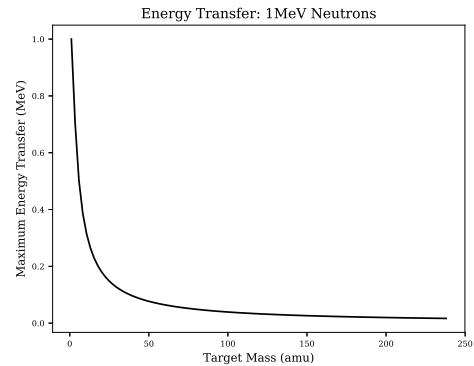


Figure 3.7: Energy transfer as a function of target mass

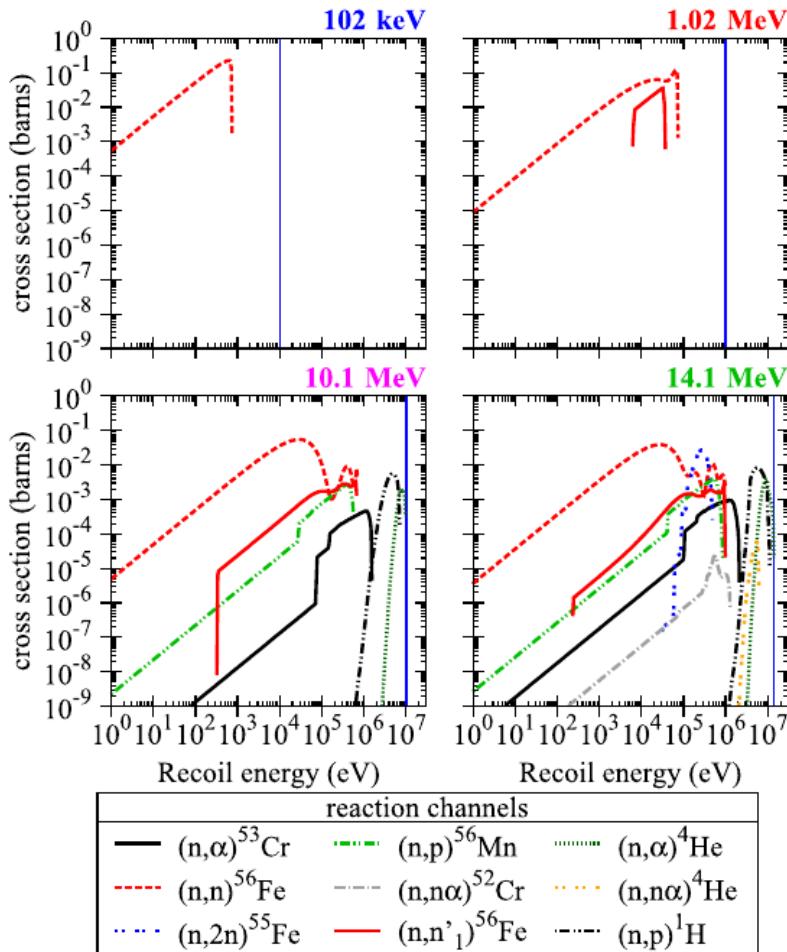


Figure 3.8: Recoil energies for neutrons at 102keV, 1.02MeV, 10.1MeV and 14.1MeV[23]

The energy of Fe PKAs as a result of neutron irradiation has been calculated for an Fe target[23] (fig. 1.10), and those created by 1MeV neutrons may have an energy up to 100keV. As nuclear reactions are also possible between the neutrons and the target nuclei, there is an inelastic $(n,n)^{56}Fe$ channel as well as other elastic scattering channels where the post collision particles are different to the pre collision particles (fig. 3.8).

3.2.4 Swelling

Swelling results from damage to the crystalline structure of the metal, and is formed through small voids being created in the crystal. As vacancies are created by radiation damage, they precipitate into small voids. The temperature range for such swelling is primarily from 670K to 870K.

Materials have been known to swell to a 100% increase in their original volume at these intermediate temperatures under irradiation[36]. Swelling is sensitive to the total damage amount, the temperature and composition of the alloy. An increase in Ni decreases swelling as does the addition of Zr. Small amounts of P and Mo (0.02% and 0.5-1.0% respectively) cause the swelling of the alloy to increase, however higher concentrations of either cause swelling to decrease[37].

As a component swells its properties change. By definition the volume changes, but so does the elastic moduli. Not only this, but the swollen component may stress itself and surrounding components, and this may lead to SCC.

In the EBR-II structural 304SS was irradiated to 20DPA. At a temperature of approximately 650K the steel's volume increased by 2% due to swelling[38]. With components in future reactors expected to withstand ten times this damage, swelling is a concern. However, increased temperatures and a careful balance in the composition of future alloys will help to reduce this.

3.2.5 Radiation Induced Segregation

The diffusion of atoms within an alloy has an impact on the characteristics of the metal [39]. When radiation causes point defects, these interstitials also diffuse parallel to thermal solute diffusion. At low temperatures, the atoms are unable to diffuse at an appreciable rate; the mobility of vacancies are low[30] and there are an excess of vacancies due to radiation damage, and this leads to recombination of defects. At high temperatures, there is a higher concentration of thermal defects[35]. This increases the defect recombination rate and reduces migration of defects to sinks, such as grain boundaries.

The melting temperature range of 304 and 316 stainless steel are approximately 1695-1722K and 1644-1672K respectively. Gen II reactors, such as Sizewell B, have an operating temperature of several hundred degrees centigrade. The inlet temperature for the Sizewell B PWR reactor is 566K[40] and the outlet temperature is 597K[40]. Operating at approximately 35% the melting point of the steel this is, unfortunately, a good temperature for radiation induced segregation (RIS) to occur.

The Kirkendall effect (KE) concerns the diffusion of one material into another and vice versa at an interface. In the original experiment performed by Kirkendall and Smigelskas, brass (Cu and Zn) was sandwiched between Cu and left at a temperature of over 1050K for almost two months. Using Mo as a marker, it was discovered that Zn diffused out of the brass faster than the copper diffused into the brass. The flux of atoms results in a flux defects and a shrink in volume of brass[41].

The inverse Kirkendall effect (IKE) is driven by an external force, such as irradiation. Irradiation of the material causes a flux of defects and, inverse to the KE, this causes a flux of atoms. For an austenitic stainless steel the diffusion coefficients in order of size are $D_{Cr} > D_{Fe} > D_{Ni}$ [42], so Cr will diffuse away from grain boundaries and other sinks, followed by Fe. As Ni is slowest to diffuse, this will be enriched. An additional mechanism that is thought to contribute to RIS is interstitial flux where undersized atoms as interstitials are preferential[42].

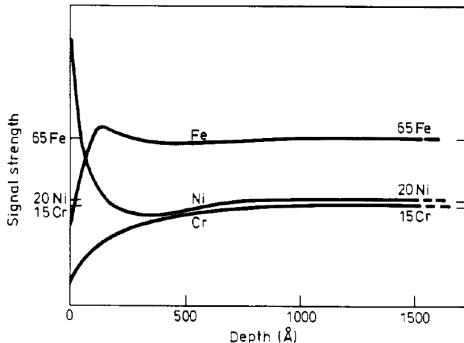


Figure 3.9: Depletion of cr, fe and enrichment of ni at the surface[43]

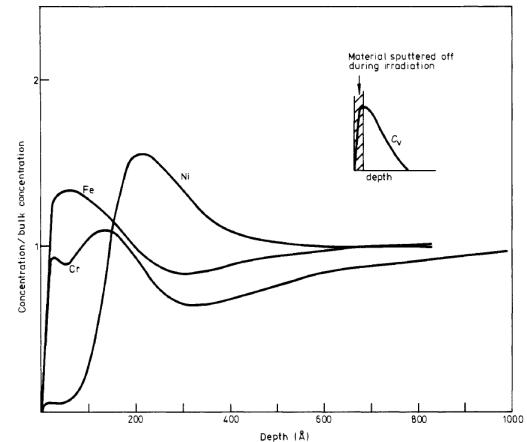


Figure 3.10: Concentration profile after 46DPA irradiation with 75 keV Ni ions[44]

Experimental work by Johnston et al irradiated steel, containing 15% Cr and 20% Ni, with 4MeV Ni ions. It was irradiated to a damage dose of 8DPA at 950K. Some material will have been sputtered from the surface, but in the remaining material Cr, and Fe, were depleted. Ni, however, was enriched (fig. 3.9).

In work by Marwick a similar alloy with 14% Cr and 15% Ni was irradiated to a much higher dose of 46 DPA with lower energy 75KeV Ni ions. In this instance the damage occurs in a much thinner layer of the surface that was approximately 30nm thick (fig. 3.10). Conversely to Johnston et al, in this thin layer there is a depletion of Ni and an enrichment of Cr in the first 20nm or so. However, between approximately 25nm and 60nm there is a similar depletion of Fe and Cr as well as an enrichment of Ni.

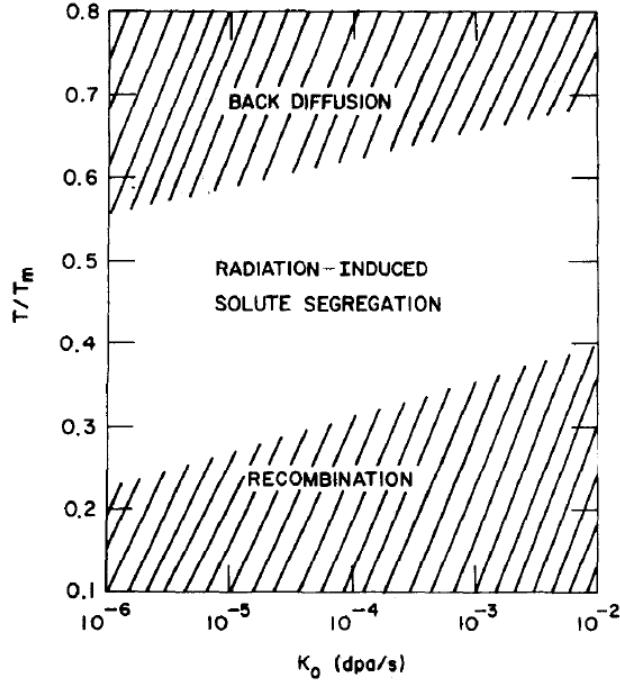


Figure 3.11: Temperature and DPA dependence of RIS[45]

The underlying mechanisms of RIS are dependent on both the temperature (relative to the melting point of the material) and the amount of damage (fig. 3.11). The result of RIS is a breakdown in the passive protection of Cr, which leads to IGSCC under the correct conditions.

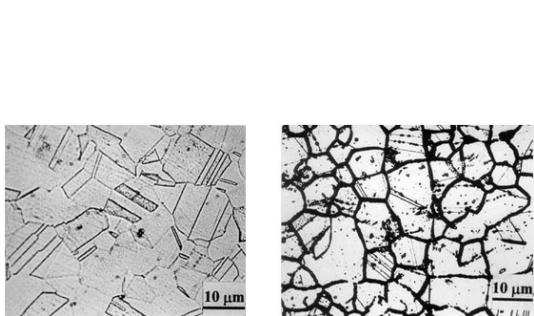


Figure 3.12: Formation of chromium precipitates at the grain boundary [50]

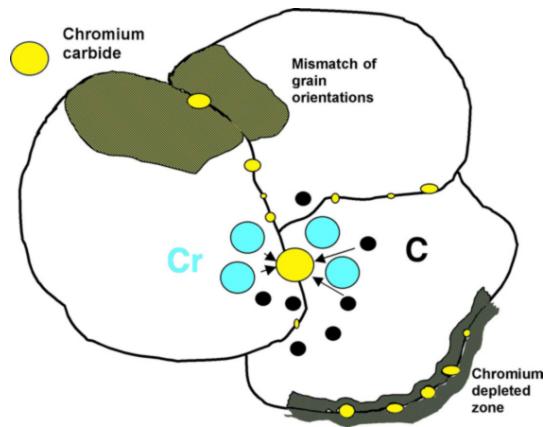


Figure 3.13: Sensitization of an alloy: chromium carbide precipitation [51]

3.3 Corrosion Resistance of Austenitic Stainless Steels

3.3.1 Passive Film Protection of Chromium

The addition of Cr improves the resistance of stainless steel to corrosion by orders of magnitude. A very thin passive oxide layer, 20-30 angstroms in width, forms when the surface is exposed to an environment containing oxygen. The protective layer is self repairing and, if the surface is damaged, the oxide layer reforms in the presence of oxygen.

When the steel is in an environment containing oxygen, electrons within the metal tunnel through the surface. As an oxidizing agent, the nearby oxygen atoms readily accept the electrons. A strong electric field forms between positive ions within the metal and the negatively charged oxygen atoms [46].

Fe at the surface of the forming oxide layer is preferentially dissolved away over Cr [47] and Cr ions within the oxide layer have a lower mobility than Fe ions [48]. Ni remains in place within the alloy, and this leads to an enrichment of chromium oxide in the passive layer. Once the layer is thick enough to reduce the electric field across it, the formation of the layer stops. As mentioned previously, this is at a layer thickness of 2-3nm for stainless steel.

The Cr_2O_3 layer may be formed by heating the steel to 770K[49]. However, at higher temperatures, the steel begins to lose its protective layer due to sensitization.

3.3.2 Sensitization and Passive Film Removal

Steel by definition is Fe alloyed with varying small percentages of C. C changes the property of the alloy, and one example of this is an increase in hardness over pure Fe. At elevated temperatures, 670K to 1020K, the added Cr within the steel forms precipitates of Fe-Cr carbides $(\text{FeCr})_{23}\text{C}_6$ at the grain boundaries, reducing the percentage of Cr at the grain boundary and removing the layer of passive protection.

The sensitization of the steel can be reversed. By heating the steel further, to approximately 1,320K to 1,420K, the steel is solution annealed dissolving the carbides back into the steel.

3.3.3 Addition of Molybdenum

A common austenitic stainless steel is 304SS and this relies heavily on the passive film due to a high content of Cr, ranging from 17.5% to 20% for grades 304, 304L and 304H. A similar stainless steel, 316, has slightly less Cr, slightly more Ni and 2-3% Mo, an element not present in 304 stainless steel.

Pure Mo has a BCC crystal structure, and it is a ferrite former. However, high proportions of austenite former such as Ni and N force 316 stainless steel to keep its FCC structure despite the addition of Mo. Adding it to austenitic steels improves their strength and resistance to creep at high temperatures.

Steels such as 316SS are known to have better corrosion resistance in chloride rich environments. Where at least 2% Mo has been added it improves resistance to pitting and crevice corrosion resistance[52]. The mechanism by which Mo helps to protect against corrosion is still unclear. It is possible the addition helps to reduce the breakdown of the passive film protecting the steel, but it may also be the case that Mo promotes the repair of the passive film [53].

3.4 Irradiation Assisted Stress Corrosion Cracking

Damage due to irradiation was first identified in high stress stainless steel components in a reactor, such as bolts, springs, and fuel elements[54][55], and later being found in lower stress austenitic stainless steel components[55].

Irradiation of water causes the radiolysis of water, changing the chemistry of water and creating hydrogen, radicals and other ions that increase the corrosiveness of the environment (fig 3.14). The radiation also damages the material directly, changing its properties. A form of IASCC to which high nickel alloys, including austenitic steels, are particularly susceptible to is inter granular stress corrosion cracking.

3.5 Inter granular stress corrosion cracking

Stress corrosion cracking may be trans granular (through the grains) or inter granular, at the grain boundary. IGSCC is a particularly prominent failure mechanism for austenitic stainless steels. For IGSCC to occur, there must be a material that is susceptible to this form of cracking, an environment that is corrosive to the material as well as stress (fig. 3.15).

Stress may come from a high pressure environment or residual stresses due to welding. In a nuclear reactor the atomic structure may be under stress due to damage caused by neutrons passing through the steel, or due to the swelling of the steel on the macroscopic scale. If Cr is depleted at the grain boundary, protection to corrosion due to the passive layer is lost. This, coupled with the knowledge that austenitic stainless steels are susceptible to IGSCC, completes the three requirements and, over time, components of this material in these conditions will eventually fail to IGSCC.

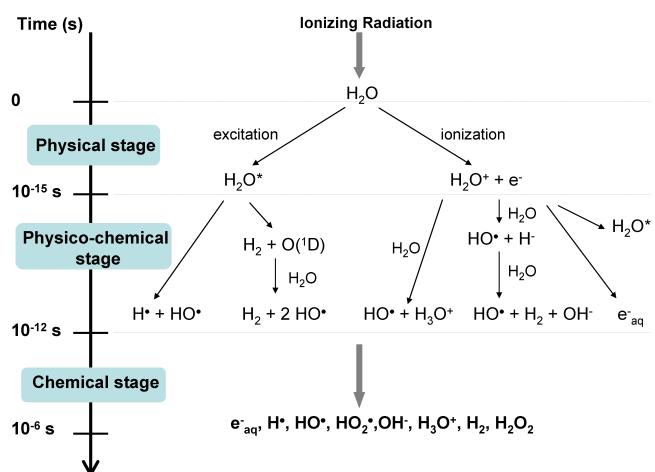


Figure 3.14: Changing water chemistry: radiolysis of water[56]



Figure 3.15: Three requirements for IGSCC to occur

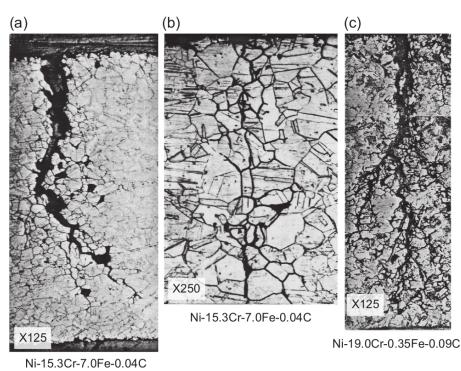


Figure 3.16: Inter Granular Stress Corrosion Cracking in Nickel Alloy[57]

IGSCC has not only been a defect in stainless steel. High Ni content alloys, such as Alloy 600 (Inconel: Ni-72, Cr-17, Fe-10), have been known to suffer from IGSCC since the very early days of nuclear energy, in particular with the prototype S1W reactor, prototype for the first nuclear powered submarine, the USS Nautilus.

After the war, there was a drive to develop a nuclear powered navy for the US. There are obvious benefits in replacing conventional power in vessels with nuclear, in particular for submarines. This was pushed by Admiral Rickover and the route to building the USS Nautilus began. A number of available Fe-Cr-Ni alloys were considered for the construction of the reactor (fig. 3.17).

Prototypes developed for the USS Nautilus experienced stress corrosion cracking in the Inconel Alloy 600 components. H. Coriou replicated this damage to Inconel in the laboratory, holding a sample at 620K in deoxygenated water for 3 months[57]. Further studies showed that IGSCC became a particular issue to alloys with a high Ni content in pure water, commencing with a content of approximately 70% Ni. In a weak salt water solution (0.1% NaCl) IGSCC occurred as with the same alloy in pure water, but TGS SCC occurred in similar alloys with a low Ni content (less than 15%) (fig. 3.18).

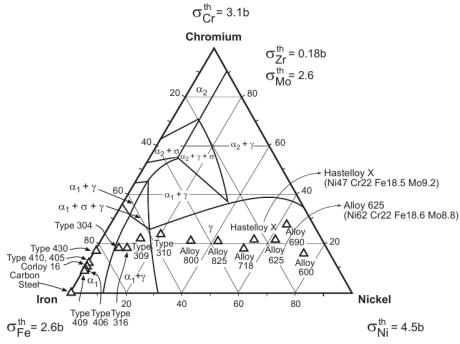


Figure 3.17: Alloy choices for early LWRs

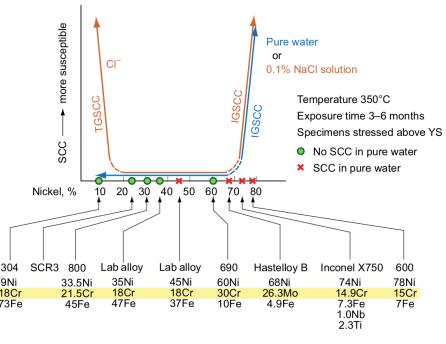


Figure 3.18: Nickel, IGSCC and TGS SCC

3.5.1 IGS SCC in Light Water Reactors

In LWRs a major factor in corrosion of components is the water chemistry and electrochemical corrosion potential that they operate in. Higher levels of oxygen dissolved in the water leads to an increase in the corrosion potential[36]. Other factors within LWRs include depletion of Cr due to either heat treatment, welding or irradiation.

The primary circuit of a BWR includes components such as a reactor pressure vessel, piping that leads out of the containment building, fuel and fuel assembly. 300 series steels, including 304 and 316, are used within the primary circuit, including the piping and control rod absorbers. In BWRs the purity of the water has been addressed, and the addition of hydrogen to the water has reduced cracking of components[57].

Higher nickel content steels such as the 600 series alloys are used in PWRs, and components that are made from these alloys include steam generators and pressure vessels. In the AP-1000 design, the control rod drive mechanism at the reactor coolant pressure boundary are made from 304, 304L, 304LN, 316, 316L and 316LN steel. Higher nickel content 690 is used for penetration into the pressure vessel[58]. As illustrated by Coriou's work in the 1950s and 1960s, higher nickel content steels are susceptible to stress corrosion cracking.

3.5.2 IGS SCC and Advanced Gas-cooled Reactors

The first generation of nuclear reactors in the UK were Magnox type reactors. They used un enriched Uranium 0.7% U235 as fuel which was contained within magnesium oxide cladding due to its low absorption cross section. A major drawback was the relatively low operating temperature of 630K.

Carnot's theorem shows that the maximum amount of energy from an engine is dependent on the difference between the hot and cold reservoirs (section 1.3.4). The ambient temperature of the power station will vary a small amount with the seasons. The practical way to improve the maximum possible efficiency is to increase the engine (reactor) temperature.

$$\eta_{max} = 1 - \frac{T_c}{T_h} \text{ where the temperature is measured in absolute units} \quad (3.2)$$

AGR's were designed to run at much higher temperatures of 920K. By increasing the temperature, the maximum possible thermal efficiency was increased from just over 50 percent to almost 70 percent. To withstand higher temperatures, the magnesium oxide cladding used in the earlier Magnox reactors was replaced with stainless steel. To counteract the higher neutron absorption cross section of the cladding, the fuel was enriched up to 3.5% U²³⁵.

The cladding holds the fuel elements together and at the required location inside the reactor whilst it is consumed in the reactor. The cladding also performs several functions once the fuel has been spent. It holds the fuel elements together and acts as a primary containment for the spent fuel[59].

During its lifetime as a fuel element within the reactor, the cladding has been heated and irradiated. By the process of RIS the Cr at the grain boundary can drop to 10% concentration[60]. The C rich environment within the reactor, due to the CO₂ coolant, provides yet another mechanism to deplete Cr at the grain boundary by the formation of Fe-Cr carbides (FeCr)₂₃C₆.

This loss of protection at the grain boundary, stress due to the role of the component within the reactor, the changing environment due to radiation damage and the material's susceptibility lead to IGSCC.

3.6 Austenitic Stainless Steels in Gen III+ and Gen IV Reactors

The AP1000 is an advanced PWR. As a Westinghouse reactor, the fuel cladding will be their trademarked Zirlo alloy. Inconel will be used for the steam generator and heat exchanger with carbon steel used for the construction of the pressure vessel. The control rod absorbers, however, will be constructed from 304SS[61].

Areva designed the EPR and this will use 316SS as the fuel cladding material. It will also use the slightly less corrosion resistant 304SS (forged) in parts of the control rod drive mechanisms[62].

The perforated upper core plate, between the reactor core and the inlet/outlet/control rod mechanisms, is also made from austenitic stainless steel. The pressuriser is constructed from ferritic steel, but its internals are clad with austenitic stainless steel to protect the structure from the coolant.

Gen IV designs also include the use of austenitic stainless steels. Experimental fast reactors in many countries, including the US, UK, Japan, France, India and China have used either 304SS, 316SS or both in their construction.

Austenitic steels are more corrosion resistant than ferritic or martensitic steels. They do expand more as a result of being irradiated, but they are stronger at higher temperatures. Many of the Gen IV reactors operate at much higher temperatures than existing reactors and the components will be expected to resist higher doses of radiation damage throughout their lifetime.

GFRs are expected to use austenitic steels. Due to their chemical compatibility with sodium they are also expected to be used in SFRs. As well as being used in the reactor core, these steels will also be used outside the core. For example, SFRs in France will use austenitic steels in the secondary circuit, primary pump, internal heat exchanger and more[63].

3.7 The Addition of Platinum group metals to Stainless Steel

Water purity is a concern for light water reactors. The fewer contaminants there are to begin with, the better. Even so, the radiolysis of the purest water will still create a steady supply of corrosive molecules (fig. 3.14). Adding hydrogen to the water reduces the electrochemical corrosion potential of the water, but if too much is added it will combine with the radioactive nitrogen-16 created by the $^{16}O(n, p)^{16}N$ reaction to form $^{16}NH_3$ [64].

Cathodic modification is a technique used to improve corrosion resistance, and one way to do this for stainless steel is to add small amounts of PGMs[65]. By adding such metals, the anodic reaction may be reduced. PGMs also act as a catalyst in the reduction and removal of O₂ and H₂O₂ from the water[64].

The increased corrosion resistance is dependant upon the amount of PGM added, but a higher percentage does not necessarily mean better protection. This is important for two reasons, to find the optimum amount for the

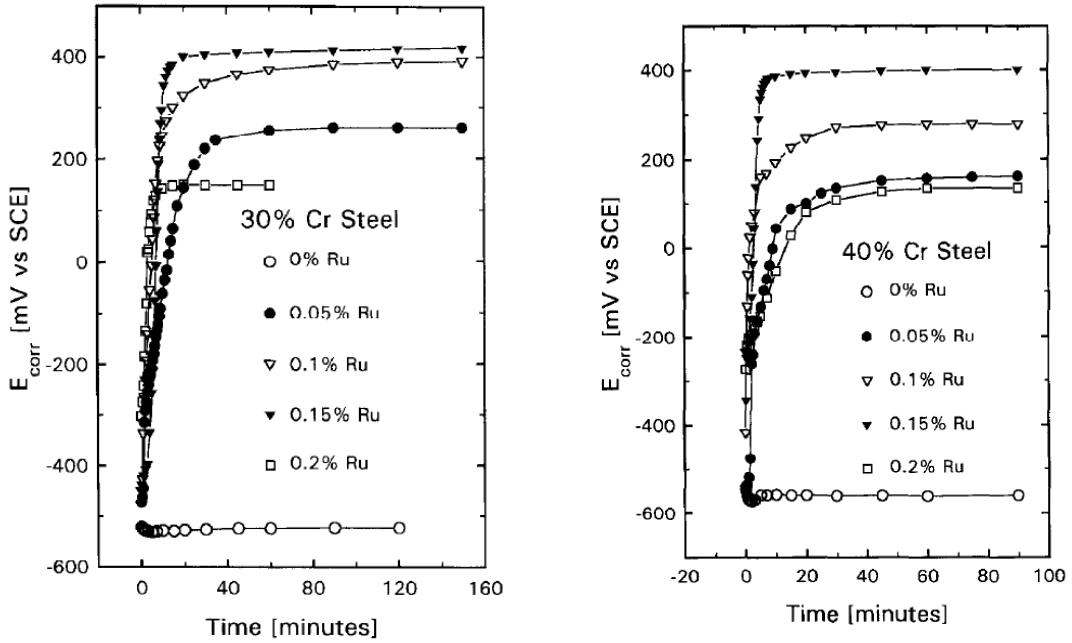


Figure 3.19: Addition of Ru to Fe30Cr and Fe40Cr steels - open current corrosion potential variation with time in 10% sulphuric acid[65]

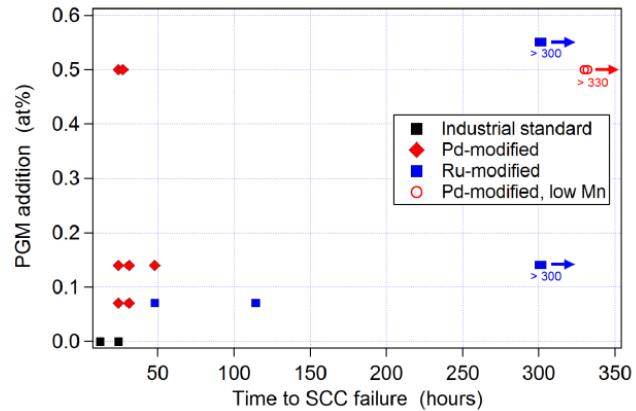


Figure 3.20: Failure times for 304SS, Pd doped 304SS (with and without Mn) and Ru doped 304SS in polythionic acid at ambient temperature[1]

sake of corrosion resistance, and to keep the price down as the additions are so expensive, even in such small amounts.

It was found that an addition of 0.15% Ru (out of a selection of 0%, 0.05%, 0.10%, 0.15% and 0.20%) was optimum for both Fe30Cr and Fe40Cr steel (fig. 3.19)[65]. Other PGMs may be used for cathodic modification, including Pd, Ir and Pt, but the cost of the metal must be weighed against the possible gains.

Connolly et al recently added Ru and Pd to 304SS samples that were sensitized at 920K for 24 hours[1]. After sensitization, the steel was analysed by TEM and Cr-rich carbides were found at grain boundaries. Without the passive protection due to Cr, the samples were exposed to polythionic acid to promote SCC. The Ru doped steels performed well, but the Pd doped steels showed no improvement over regular 304SS. Further analysis of the samples by TEM reveals that the low percentage of Mn in the steel allows the formation of Pd-Mn precipitates at the grain boundary[1]. These precipitates form during the sensitization procedure and thus negate the protection by cathodic modification at the surface of the metal by removing the Pd.

With a low Mn 304SS (less than 0.05% Mn), varied amounts of Pd have been tested. With the reduction in Mn

the corrosion resistance improves greatly over regular 304SS. Palladium remains at the surface, as the amount of Mn is too low for appreciable amounts of Pd to be lost to precipitate formation during sensitization. No failures have been observed with either the Ru or Pd (low Mn) samples for 330 hours exposure to polythionic acid at ambient temperatures (fig. 3.20) [1].

The surface of the metal does not need to have a full coating of PGM but it should be equally spaced across the surface. Oxidant in the layer at the surface will be consumed at the locations of PGMs and the change in concentration at those points will naturally cause diffusion of more oxidant to those locations[64].

The PGMs may be coated on the surface of the metal or alloyed to the metal. The first method is the more cost efficient but if a crack does form, or if a layer is removed, it will expose the regular alloy underneath. If the PGM is alloyed, there will always be protection, unless a mechanism is removing the PGM from the surface. This could be through precipitation as a carbide, or it could be due to radiation causing the metals to segregate.

Reaction	Product Halflife
$^{54}Cr(n, \gamma)^{55}Cr$	3.5 mins
$^{64}Ni(n, \gamma)^{65}Ni$	2.5 hrs
$^{55}Mn(n, \gamma)^{55}Mn$	2.6 hrs
$^{104}Ru(n, \gamma)^{105}Ru$	4.4 hrs
$^{108}Pd(n, \gamma)^{109}Pd$	13.7 hrs
$^{196}Pt(n, \gamma)^{197}Pt$	19.9 hrs

Table 3.2: Radioactive products and their half lives for PGM doped 304SS

Adding elements to metals, even in small amounts, may be problematic as certain elements have high neutron reaction cross sections and transmute into radioactive isotopes. ^{55}Cr is a concern for all neutron irradiated stainless steels, but with a very short half life it quickly cools and becomes less of a concern, with ^{65}Ni becoming the dominant source of activity after 10 hours. Adding Mn to steel that is in a reactor is known to result in an increase in activity due to the $^{55}Mn(n, \gamma)^{56}Mn$ reaction (section 2.2).

The addition of PGMs alters the activity of the steel following neutron irradiation and with just a 1% addition there is an increase in activity comparable to that of the 10% Ni transmuted to ^{65}Ni . This is due to reactions that create ^{105}Ru , ^{109}Pd and ^{197}Pt for Ru, Pd and Pt respectively (table 3.2).

With such small additions required for cathodic modification and the short half lives of the radioactive isotopes, there isn't a concern due to neutron activation over regular 304SS, and much less of a concern than that of 304SS containing Mn. More data computed using the TENDL library is available in appendix ??.

Chapter 4

Background: Proton Activation and Radioactive Decay

Where proton or deuteron radiation is used to emulate neutron damage, there is still an associated risk of transmuting atoms in the target material, creating radioactive isotopes. An equation was derived to calculate the radioactivity of each element in the decay chain. It takes into account a source term for each isotope in the decay chain, as well as any branching factors.

Once derived, the equation was used in a computer code that, coupled with cross section data from the TENDL library and ion transport data from the SRIM ion transport code, predicts the radioactivity of a sample (chapter 7).

4.1 Ion Sources

4.1.1 LINAC

Since the development of the first linear accelerator (LINAC) in the 1940s, their modern day versions have become some of the most powerful accelerators in the world. The longest LINAC, Stanford Linear Accelerator Center, is 3.2km in length and it accelerates electrons and positrons at energies of up to 50GeV. Several LINACs for protons include the 800MeV LINAC component of the ISIS neutron source in Oxfordshire, and the 800MeV LINAC used by the Spallation Neutron Source at Oak Ridge National Laboratory.

The accelerator is constructed of several tubes, connected alternately to opposite terminals of a high frequency alternating current supply. As protons enter the first tube, a negative voltage is applied. As the protons reach the gap between the first and second tube, the polarity is reversed. The positive charge that is now applied to the first tube pushes the protons forward as the negative charge on the second tube pulls the protons forward. This process is repeated along the length of the accelerator, with the sections increasing in length due to the increase in velocity of the protons.

4.1.2 Cyclotron

Cyclotrons are reasonably compact and cost effective. The largest current cyclotron, TRI-University Meson Facility, is located in Canada and is able to output protons with energies over 500MeV. It is relatively large, weighing 170 tons. There are approximately 350 cyclotrons[66] around the world today.

The University of Birmingham cyclotron is more compact. The protons it accelerates are in the 8-40MeV range and the projectiles are not just restricted to protons (table 4.1).

Projectile	Energy (MeV)	Maximum Current (micro A)
proton	8-40	60
deuteron	8-40	30
He^{2+}	8-53	30

Table 4.1: University of Birmingham Cyclotron Ion Beams

The moderate size, energy range, availability and cost of these Cyclotrons make them ideal candidates for damaging targets with light ions, rather than neutrons from a neutron source.

The Scanditronix MC40 is an isochronous cyclotron. As the ions are accelerated to higher and higher velocities, the effects of special relativity become appreciable.

$$v = c \sqrt{1 - \frac{E_{rest}^2}{E_V^2}} \quad (4.1)$$

The cyclotron has two Dees and these are D shaped hollow electrodes. Ions enter at the center of the cyclotron and are held in the Dees as they are accelerated. The magnetic field is varied to bend the path of the ions from a circle into a round cornered triangle.

4.1.3 Synchrotron

Two of the most well known accelerators are Synchrotrons: the Large Hadron Collider at Conseil Europen pour la Recherche Nuclaire, and the now retired Tevatron at Fermilab. These are typically large machines and there are approximately 70 around the world[67], approximately a fifth the number of Cyclotrons.

Synchrotrons are used for storing high energy particles in a continuous loop, either to generate light through magnetobremssstrahlung, or for high energy collisions. Cyclotrons are more commonly used for lower energy (tens of MeV) projectiles, in materials science for damaging materials and for medical use (creating radioactive isotopes for use in hospitals).

4.2 Neutron Sources

4.2.1 High-Flux Neutron Reactors

There are 250 or so research reactors in 55 countries [68], and a number of these are used for materials research. In the UK, there is only one remaining research reactor, and this is the Neptune pool type reactor at Rolls Royce[69]. In Cadarache, France, the Jules Horowitz reactor is under construction and this is being built specifically as a materials testing reactor. It will be crucial in researching new materials for use in upcomming Gen IV nuclear power stations [68].

High Flux Isotope Reactor, Oak Ridge

The High Flux Isotope Reactor, at the Oak Ridge National Laboratory in America, is an 85MW research reactor that provides testing space within the reactor as well as a number of neutron beam lines. The reactor uses

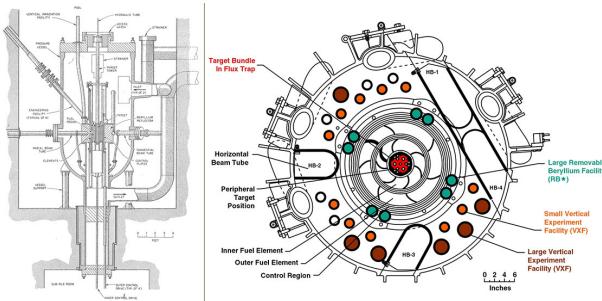


Figure 4.1: A cross section of the HFIR [71]

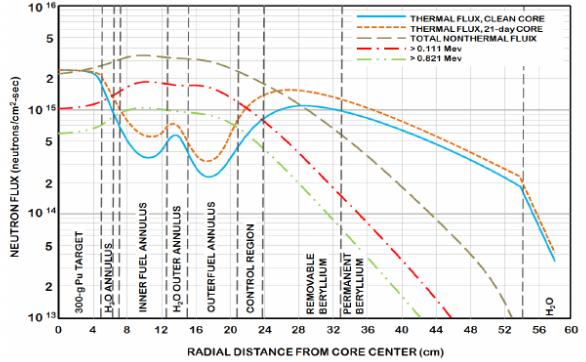


Figure 4.2: Neutron Flux in the HFIR while at 85MW [72]

highly enriched Uranium as its fuel source and is scheduled to operate at 100% capacity for 161 days per year, in cycles of approximately 23 days. A 30cm surround of Beryllium is used to reflect neutrons, and in the reactor core there is a high flux of thermal neutrons at a rate of 2.3×10^{15} neutrons $cm^{-2}s^{-1}$ [70].

NIST Center for Neutron Research

The National Bureau of Standards Reactor (fig. 4.3) was designed as a research reactor with a power output of 40MW and a neutron flux of approximately 1.0×10^{15} neutrons $cm^{-2}s^{-1}$. It uses highly enriched Uranium as fuel and is both moderated and cooled by heavy water.

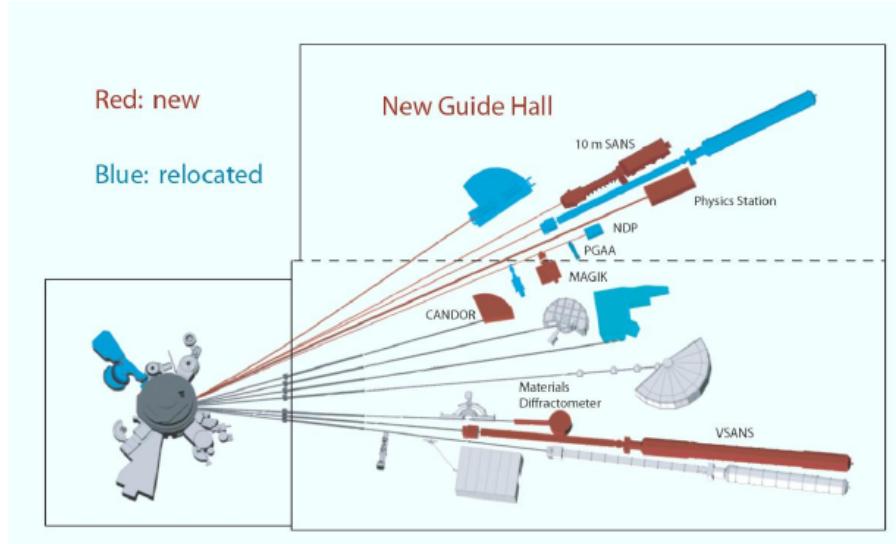


Figure 4.3: NBSR building layout[73]

4.2.2 Spallation

Neutrons cause fission in nuclear reactors, but spallation sources require protons and high mass targets. The energy of the protons required is a magnitude greater than that of the Scanditronix MC-40 Cyclotron at the University of Birmingham, with spallation source accelerators having a range from 500MeV to over 1GeV.

ISIS is a neutron spallation source located in Harwell, Oxford. Protons are accelerated in a LINAC to 0.37c before being fed into a 163m circumference synchrotron. The protons are then accelerated to 0.84c before being projected, in bunches, at a tungsten target. The impact with the tungsten releases neutrons[74].

Oak Ridge National Laboratory also has a neutron spallation source, the SNS. A LINAC accelerates a negatively charged proton (a hydrogen atom with two electrons) up to just below 0.9c. It is stripped of the electrons and enters an accumulation ring as a proton. More protons are accumulated until the beam in the accumulator is diverted at a mercury target. The collision between the mercury target and the protons releases approximately 20 neutrons per collision, and these neutrons pass through beam lines to the experiments.

The source creates a pulse of neutrons from the 50 ton mercury target 60 times per second. The protons are accelerated by the LINAC to between 2.5MeV and 1.0GeV and this results in neutrons with energies almost as high as the proton projectiles. A spectrum of neutron energies is produced with each pulse, although the neutrons may be moderated at the end of the beam lines to slow the neutrons down.

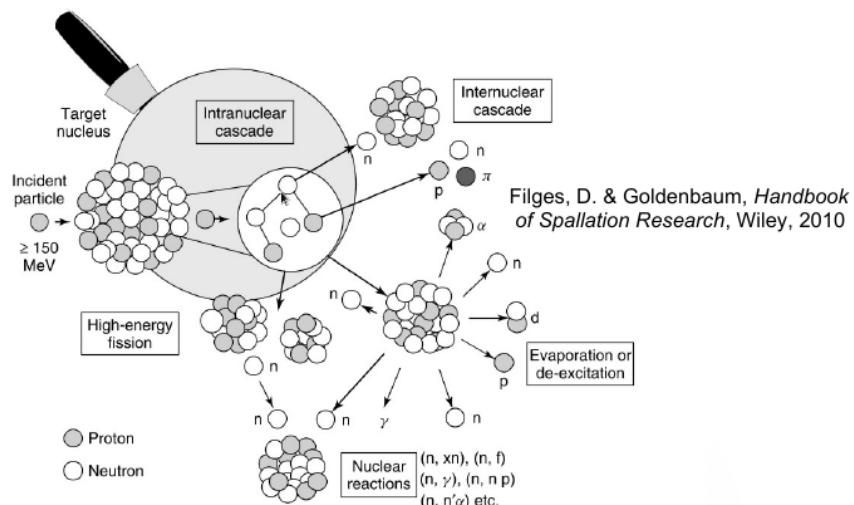


Figure 4.4: An example of a neutron spallation source where high energy ions collide with heavy atoms[75]

4.3 Source Review

Source	Cost	Projectile	Flux/Current	Energy
Scanditronix MC-40	1 million	Proton	3.7×10^{14}	8-40MeV
HFIR[72]		Neutron	3.0×10^{15}	Full Spectrum
HFIR[72]		Thermal Neutron	Over 2.0×10^{15}	Thermal
HFIR[72]		Fast Neutron	2.0×10^{15}	$> 0.111\text{MeV}$
HFIR[72]		Fast Neutron	1.0×10^{15}	$> 0.821\text{MeV}$
SNS[76]	\$1.4 billion	Neutron	Average 1.2×10^{13}	Full Spectrum
ISIS spallation source[76]		Neutron	Average 4.0×10^{13}	Full Spectrum

Table 4.2: Examples of neutron sources, energies and flux, with the MC-40 as a reference for protons

The compact proton source that is the cyclotron is compared to neutron sources. It is much smaller and cheaper in comparison and the energy of the projectile may be controlled, whereas a spectrum of energies of neutrons is produced by reactors and spallation sources. Faster neutrons will cause more damage to materials in Gen IV reactors giving the cyclotron an advantage, being able to focus in just one energy range. The three mentioned isotope sources are national facilities that cost much more than a cyclotron. Access is shared between many researchers whereas a cyclotron, such as the Scanditronix MC-40 at the University of Birmingham may have a beam dedicated to a particular task.

4.4 Ion Irradiation to Investigate Neutron Damage

Neutrons emitted during the fission of Uranium-235 have an energy spectra in the intermediate to fast range, with a peak at 1MeV, and a large proportion in the 1MeV to 10MeV range (fig. 4.5). The higher energy neutrons are more of a concern to this work as higher energy neutrons, on colliding with atoms within the target material, cause large damage cascades.

As discussed earlier in this chapter, there are a number of methods available to create both high energy neutrons and ions, but each has its own set of advantages and disadvantages. At the University of Birmingham high energy ions are created using a cyclotron.

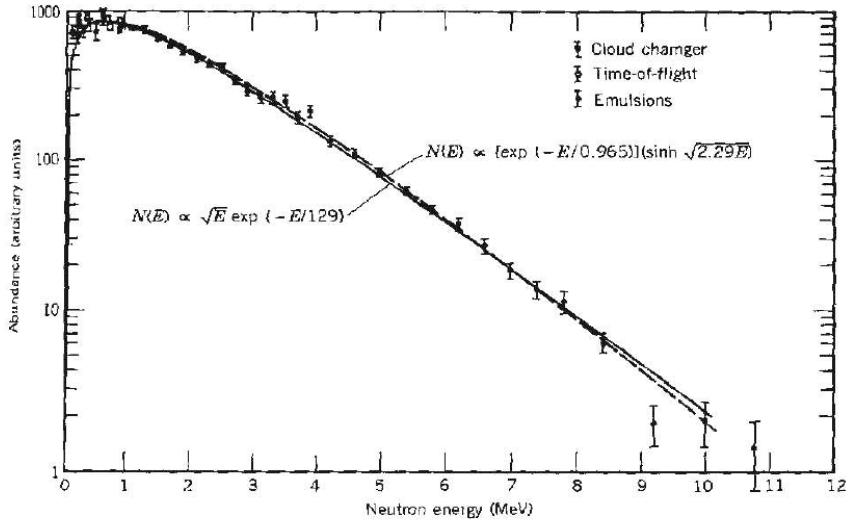


Figure 4.5: Neutron Spectra from the Fission of U235[77]

4.4.1 Ion Irradiation at the University of Birmingham

The Scanditronix MC-40 Cyclotron is used at the University of Birmingham to create a beam of protons or other light ions. The energies of these ions are typically between 10 MeV and 60 MeV with beam currents ranging up to 50 microamps (3.1×10^{14} protons per second). Target materials are irradiated by this cyclotron for a number of reasons, including purposely creating radioactive isotopes for the nearby Queen Elizabeth Hospital, investigating ion irradiation damage and emulating neutron irradiation.

The Cyclotron is usually used to create radioactive isotopes for medical use, but an additional beam line has been devoted to material science investigations into radiation damage. While the creation of radioactive isotopes is desired in some cases, material being tested for radiation damage should preferably have low levels of radioactivity.

It is expensive to arrange the irradiation of target materials by high energy neutrons sources, whereas it is relatively inexpensive to irradiate using an ion beam on the MC-40 Cyclotron. The energies can be controlled, and a set dose at a single energy, or a range of energies, can be precisely deposited into the target material. The reaction cross section for neutrons also cover a much larger range including lower energy projectiles, something Coulomb repulsion reduces for ions (fig. 4.6).

The Activity code discussed in this work was developed to calculate the activity of a target material irradiated

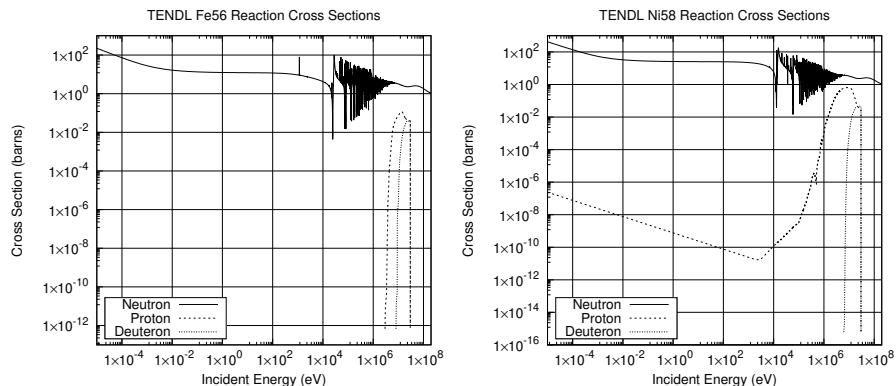


Figure 4.6: Neutron, proton and deuteron cross sections for Fe56 and Ni58

by a proton beam. It has been developed in Fortran and uses data from the TENDL-2013 proton cross section database, SRIM ion transport code and Nuclear Data Services radioactive decay database. A second version was also developed using Python and a more recent TENDL database.

4.4.2 Transmutation of Nuclei by Neutrons and Protons

Neutron Activation

The fission of Uranium-235 atoms results in neutrons with a varied spectrum of energies. The neutrons will bounce around inside the reactor losing energy quickly to light atoms within moderators and coolants, such as water. At lower energies the neutrons may be captured by the nuclei of atoms they interact with. This creates a new isotope which may or may not be stable.

Proton Activation

Considering a simplified nuclear potential well, energetic protons approaching a nucleus may overcome the Coulomb potential barrier. They are captured by the nucleus and held within the potential well by the strong nuclear force. This process may leave the nucleus in an excited and unstable state, depending on the input energy of the proton and configuration of nucleons. The process is probabilistic, and the average chance of a reaction (the microscopic cross section) may be measured as a function of the projectile, projectile energy and target, either experimentally or by optical model potential calculations. The reaction rate is calculated from the microscopic cross section using the following equation:

$$R = \frac{J}{Q} \cdot n_p \cdot \sigma \cdot 10^{-28} \delta t \quad (4.2)$$

- R Reaction Rate (reactions per second)
- J Beam current (A)
- n_p Number density of target (atoms per cubic metre)
- σ microscopic reaction cross section (barns)
- Q projectile charge e.g $1.602177 \times 10^{-19} C$ for a proton
- δt target thickness (m)

4.4.3 Nuclear Reaction Cross Sections

Reaction Cross Sections

The type of reaction for an individual reaction cannot be determined, but the probability of that reaction happening may be measured and future events predicted. This data may be gathered experimentally, or it may be calculated using various models.

TALYS and the Optical Model Potential

In the standard model, protons and neutrons are composed of quarks, held together by the strong nuclear force. The nucleus of an atom is also held together by the strong nuclear force that on such small separations overwhelms the electromagnetic force of the protons with one another.

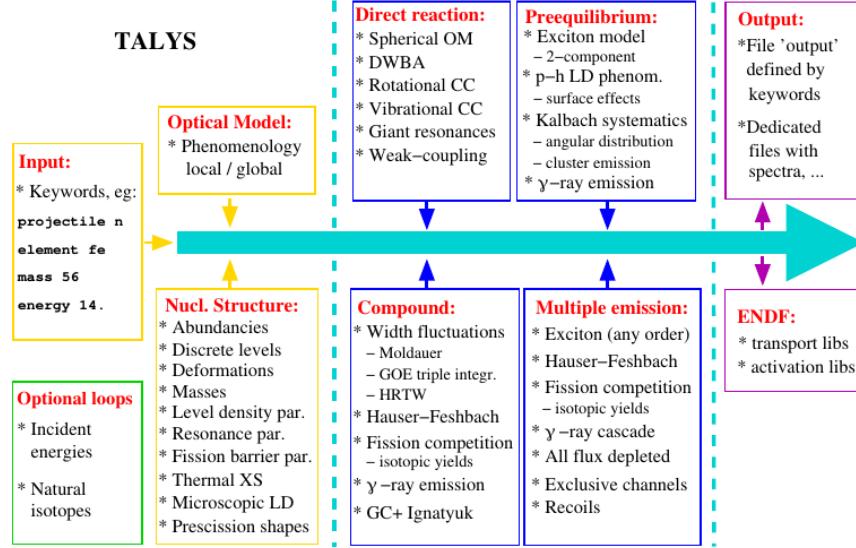


Figure 4.7: TALYS work flow [79]

This is a complicated system, not to mention the excited states that nuclei may occupy. The interaction of a projectile (proton, neutron or a heavier ion) with the nucleus is a challenge to model.

Initially, reaction data for protons and neutrons were gathered through experimentation. In the 1950s, Feshbach et al put forward a simple model for nuclear reactions between the nucleus and neutrons, and this model was restricted to 0-20MeV neutrons. The form of the potential used was a complex function[78].

$$\begin{aligned} V &= V_0(1 + i\zeta)r < R \\ V &= 0r > R \end{aligned} \tag{4.3}$$

The potential in equation 4.3 has the parameters $R = 1.4 \times A^{\frac{1}{3}}$, $V_0 = 42\text{MeV}$ and $\zeta = 0.03$ where A is the atomic mass of the target atom.

Considering the complexity of the system being modelled, this simple model was very successful. Over the years since, the models used have become more complex and parameters used have been fit to an increasing amount of experimental data.

The Talys code uses a range of models (figure 4.7). These include the optical model which is solved using the Equations Couples Itrations Squentielle (ECIS-06) code of Jacques Raynal, implemented as a subroutine within Talys, and this is accurate up to 180MeV[79]. Whilst the Talys code has been extended up to 1GeV, it is experimental in the 180MeV to 1GeV range. Fortunately, this work only requires nuclear reaction cross section data up to approximately 100MeV as the current ion source under consideration produced ions with energies up to 60MeV.

The Talys code has potentials for protons and neutrons, but it also has potentials for deuterons, tritons, helium-3 and alpha particles. The potentials are discussed in detail in the Talys manual[79]. This extension to heavier ions may be useful for this work as the University of Birmingham cyclotron is capable of accelerating deuterons and He^{2+} ions.

Comparing experimental data to the Talys data for $Fe54(p, \gamma)Co55$ shows good agreement between 1MeV and 10MeV[4.8]. There was insufficient experimental data from this source, but the Talys generated data ranges from

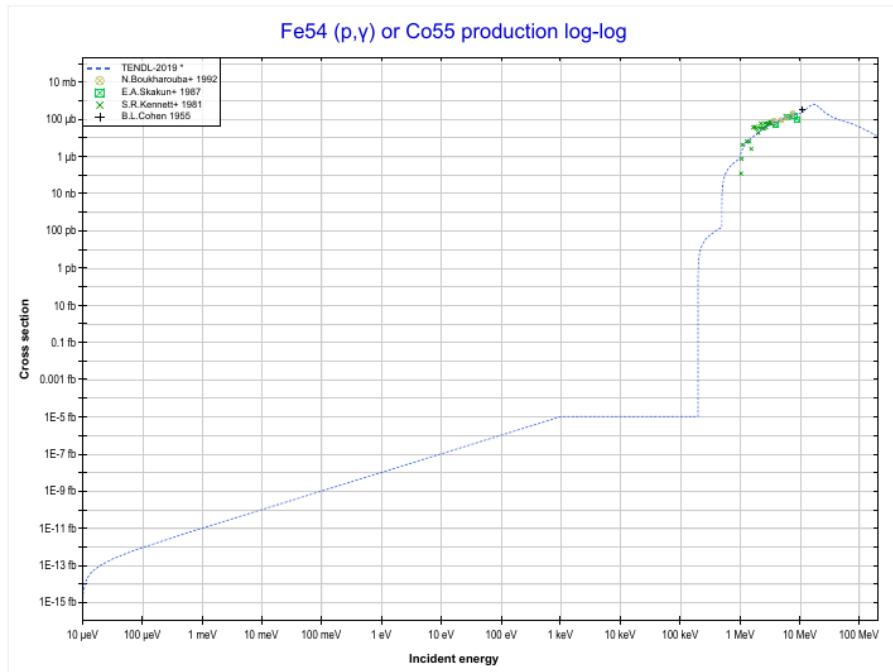


Figure 4.8: TALYS Fe54-Co55 cross section comparison with experiment [80]

micro eV up to 100MeV+ (fig. 4.8).

PADF

The Proton Activation Data File was released in 2007 and contained nuclear reaction data for 2355 target nuclei, ranging from Magnesium (12) to Radon (86) with proton energies up to 150MeV. The file contains the sum of all individual reactions as well as certain yields, and the data was generated using the TALYS and ALICE/ASH codes, as well as experimental data from Exfor[81].

Listing 4.1: A sample of the Iron-56 PADF data file

```
1 Proton Activation Data File          528 1451 12
2 ****
3
4 Authors: C.H.M.Broeders, U.Fischer, A.Yu.Konobeyev, L.Mercatali 528 1451 15
5
6
7 Data for PADF file were obtained using the TALYS code [1] for 528 1451 18
8 target nuclei with half-life more than 10 min and using the 528 1451 19
9 ALICE/ASH code [2] for nuclei with 1 sec < T1/2 < 10 min and 528 1451 20
10 available experimental data        528 1451 21
11 MAT numbers are taken according to JEFF-3.1/RDD      528 1451 22
12
13 Evaluation for Fe-56 (stable) : TALYS code          528 1451 24
14 Experimental data used for the correction of calculated 528 1451 25
15 excitation functions are taken from Refs.[3-12]       528 1451 26
16
17
18 File contains                      528 1451 28
19
20 MF=3 MT=5                          528 1451 30
21 Sum of all individual reaction cross-sections (Total 528 1451 32
22 reaction cross-section)           528 1451 33
23
24 MF=6 MT=5                          528 1451 34
25 Yields of nuclei in nuclear reactions including 528 1451 35
26 n, p, d, t, He-3, He-4 and photon production    528 1451 37
```

TENDL 2019 Data Files

The cross section data for protons and neutrons is available to download in ENDF format files. The data files used by this work are TENDL data files, and these are created by a combination of different nuclear models and experimental data. The TENDL nuclear reaction simulation code provides the calculated data.

The nuclear reaction files are rather large, and they all follow a standard format.

Listing 4.2: Sample TENDL File

```
1 2.605600+4 5.545443+1      0      0      0      02631 3 2   1
2 0.000000+0 0.000000+0      0      0      1      462631 3 2   2
3      46      2
4      2631 3 2   3
5 1.000000+3-9.920042-7 1.000000+6-9.920042-7 2.000000+6-2.167241-42631 3 2 4
6 3.000000+6-7.609736-3 4.000000+6-3.232345-2 5.000000+6-5.975554-22631 3 2 5
7 6.000000+6-5.149510-2 7.000000+6-3.685484-2 8.000000+6-5.542173-22631 3 2 6
8 9.000000+6-9.978765-2 1.000000+7-1.599116-1 1.100000+7-2.206871-12631 3 2 7
9 1.200000+7-2.743084-1 1.300000+7-3.165007-1 1.400000+7-3.471231-12631 3 2 8
10 1.500000+7-3.674350-1 1.600000+7-3.788282-1 1.700000+7-3.825634-12631 3 2 9
11 1.800000+7-3.799978-1 1.900000+7-3.725944-1 2.000000+7-3.617662-12631 3 2 10
12 2.200000+7-3.342054-1 2.400000+7-3.031296-1 2.600000+7-2.709398-12631 3 2 11
13 2.800000+7-2.388019-1 3.000000+7-2.077513-1 3.500000+7-1.398614-12631 3 2 12
14 4.000000+7-8.801994-2 4.500000+7-5.013441-2 5.000000+7-2.349373-22631 3 2 13
15 5.500000+7-5.480233-3 6.000000+7 6.189980-3 6.500000+7 1.332313-22631 3 2 14
```

```

15  7.000000+7 1.727774-2 7.500000+7 1.906833-2 8.000000+7 1.943711-22631 3 2 15
16  9.000000+7 1.783081-2 1.000000+8 1.499871-2 1.100000+8 1.211197-22631 3 2 16
17  1.200000+8 9.641120-3 1.300000+8 7.717424-3 1.400000+8 6.318436-32631 3 2 17
18  1.500000+8 5.367555-3 1.600000+8 4.771939-3 1.800000+8 4.312985-32631 3 2 18
19  2.000000+8 4.445323-3          2631 3 2 19

```

4.4.4 Radioactive Decay

Radioactive decay is the random change in nucleons or energy state of an unstable nucleus. It is impossible to predict when a single nucleus will decay, but the decay of a collection of nuclei is statistical in nature. The radioactivity and number of unstable nuclei at time t can be predicted using the decay constant, λ , for the radioactive isotope. This constant is defined as follows:

$$\lambda = -\frac{N'(t)}{N(t)} \quad (4.4)$$

The number of radioactive nuclei $N(t)$ at time t is given by the following equation, where $N(0)$ is the starting number of nuclei:

$$N(t) = N(0) \exp(-t\lambda) \quad (4.5)$$

The activity $A(t)$ of the radioactive nuclei is predicted at time t by using the following equations, where $N'(t)$ is the change in amount of nuclei with respect to time:

$$A(t) = -N'(t) = \lambda N(t) \quad (4.6)$$

$$A(t) = \lambda N(0) \exp(-t\lambda) \quad (4.7)$$

4.4.5 Saturation Activity

As a radioactive isotope is created by reactions between target atoms and projectiles (protons, neutrons, deuterons) it will begin to decay. The amount of the isotope will continue to increase until the decay rate equals the reaction rate for the creation of the isotope.

For a single isotope:

$$\frac{dN}{dt} = \frac{J}{Q} \cdot n_\rho \cdot \sigma \cdot 10^{-28} \delta t - \lambda N \quad (4.8)$$

$$N(t) = \frac{\frac{J}{Q} \cdot n_\rho \cdot \sigma \cdot 10^{-28} \delta t}{\lambda} (1 - \exp(-\lambda t)) \quad (4.9)$$

The saturation time for a given percentage may then be calculated directly from the decay constant.

$$t = \frac{\ln(1 - (p/100))}{-\lambda} \text{ where } p \text{ is the saturation percentage} \quad (4.10)$$

4.4.6 Decay Constants: Joint Evaluated Fission and Fusion File (JEFF) 3.3 Data File

JEFF 3.3 is an evaluated data file[22], meaning it has been evaluated by a relevant authority. The quality of the data may affect the health of the public and industry workers, so it must be evaluated. This particular file

is managed by and available through the Nuclear Energy Agency (NEA).

It is a collection of many data files. Released in 2017, it also contains several files for incident gammas, protons, deuterons, tritons, helium-3 and alphas from the TENDL 2017 data file.

The files that will be important for this work are the ENDF-6 radioactive decay data files only. The nuclear reaction cross section data will come from Talys and various iterations of the TENDL libraries. The decay data held in the JEFF 3.3 file includes isotope data, masses, half lives, branching factors, continuous and discrete gamma data.

4.4.7 Bateman Equation for Radioactive Decay

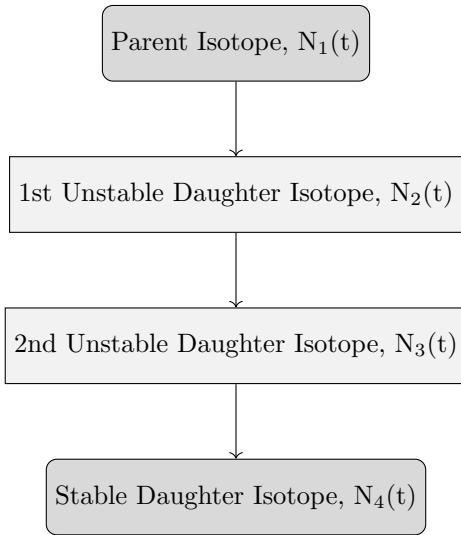


Figure 4.9: An example decay chain from an unstable parent isotope, through unstable daughter isotopes ending with a stable daughter isotope.

The English mathematician Harry Bateman derived an equation (eq. 4.11) to calculate the amount of each isotope in a decay chain, illustrated in Figure 4.9, at time t.

$$N_n(t) = \sum_{i=1}^{i=n} \left(\left(\prod_{j=i}^{j=n-1} \lambda_{(ij+1)} \right) \sum_{j=i}^{j=n} \left(\frac{N_{i0} \exp(-\lambda_j t)}{\prod_{p=i, p \neq j}^{p=n} (\lambda_p - \lambda_j)} \right) \right) \quad (4.11)$$

When a radioactive isotope decays, there may be more than one mode of decay, and this leads to branching factors. Pb-214 only decays via beta decay to Bi-214, giving a branching factor of 1.0, whereas Bi-214 has a 99.979% chance of decaying to Po-214 by beta decay and a 0.021% of emitting an alpha particle and decaying to Tl-210 (branching factors of 0.99979 and 0.00021 respectively) [22].

When a target material is irradiated, there is a source term for transmuted nuclei due to the irradiation. The daughter isotopes of these transmuted isotopes will also be affected by the irradiation and will transmute further, giving a source term for each daughter isotope as a result of the irradiation. Sources for each isotope in the decay chain, and branching factors between a parent isotope and its daughter isotope/s must be accounted for.

4.5 Simulating Ion Irradiation with SRIM and TRIM

A package of ion transport codes, SRIM, is freely available to download and use to investigate the transport of ions through matter. It includes tools to calculate the stopping range of a given ion in a material as well as TRansport In Matter, a code used to calculate the energy and position of an ion through a target.

TRIM does not take into account the structure of a target. It may be layered in the calculation perpendicular to the beam, but beyond that the target is treated as amorphous. It will give the same result for BCC iron as it would for FCC iron, providing the density remains the same. The density and whether or not the target layer is a gas or solid are options that must be configured when setting up a calculation. Each ion history is tracked through the same target; the target never changes. TRIM is ignorant to the structure and therefore damage to the structure that would accumulate over time, and to any swelling and change in composition or density.

The interaction of the ion with the target material is split into two parts: nuclear loss and electronic loss. The ion is tracked through the target until either its energy drops below a set threshold (10eV) or it leaves the target (fig 4.10). The code then moves on to the next ion history.

In a projectile-nucleon interaction, the magic formula (eq. 4.12)[82] is used to determine the scattering angle Θ of the ion and target nucleus, as well as the amount of energy transferred. The azimuthal angle is selected randomly by multiplying 2π by a random float [0.0, 1.0].

$$\begin{aligned}
 \cos\left(\frac{\Theta}{2}\right) &= \frac{B + R_c + \Delta}{R_0 + R_C} \\
 B &= \frac{p}{a} \\
 R_0 &= \frac{r_0}{a} \\
 R_C &= \frac{\rho}{a} \\
 \Delta &= A \frac{R_0 - B}{1 + G} \\
 A &= 2\alpha\epsilon B^\beta \\
 G &= \gamma \left((1 + A^2)^{\frac{1}{2}} - A \right)^{-1} \\
 \alpha &= 1 + C_1\epsilon^{-\frac{1}{2}} \\
 \beta &= \frac{C_2 + \epsilon^{\frac{1}{2}}}{C_3 + \epsilon^{\frac{1}{2}}} \\
 \gamma &= \frac{C_4 + \epsilon^{\frac{1}{2}}}{C_5 + \epsilon^{\frac{1}{2}}}
 \end{aligned} \tag{4.12}$$

where p is the impact parameter

where a is the screening length

where r_0 is the distance of closest approach

where ϵ is the reduced energy

The potential between the projectile and target is calculated at the distance of closest approach, r_0 . It is a Coulomb type potential (eq. 4.13) than includes the universal screening function (eq. 4.14). This potential will appear again, although occasionally with differing parameters, in chapter 5.

$$V(R) = \frac{Z_1 Z_2 e^2}{a R} \Phi(R) \tag{4.13}$$

$$\Phi(R) = 0.1818 \exp(-3.2x) + 0.5099 \exp(-0.9423x) + 0.2802 \exp(-0.4028x) + 0.2817 \exp(-0.2016x) \tag{4.14}$$

The recoil nuclei are also followed through several generations until their energies fall below that set for either the surface binding energy or displacement energy (3-6eV and 15-30eV respectively)[82].

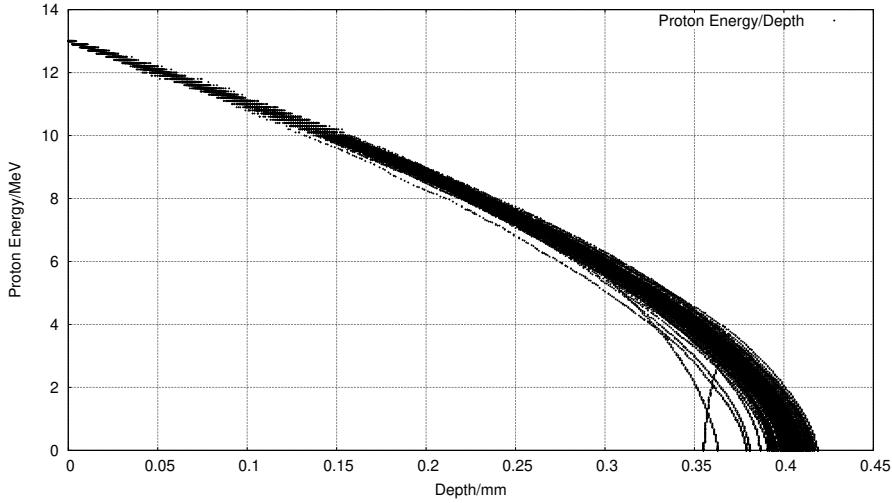


Figure 4.10: One hundred simulated 13MeV proton energy loss curves in Fe simulated with SRIM [83]

The free flight path between large collisions is calculated based on the projectile energy, type and the target material. The path length between collisions is randomly generated by taking the maximum free path and multiplying by a random float [0.0, 1.0]. Smaller projectile-nucleus interactions are not individually calculated, but the average energy that would have been lost is calculated and applied.

The ion continuously loses energy to the electrons in the material. TRIM calculates this over the free flight path, L (eq. 4.15).

$$\Delta E_e = L N S_e(E)$$

where $S_e(E)$ is the electronic stopping
and N is the atomic density of the target

(4.15)

The target damage is a combination of the incoming ion, the primary knock on atoms and their damage cascades. The user has the option of running a full damage cascade that plots the entire cascade, or a quicker calculation where the cascade is treated as just a point. The total number of displacements is the sum of vacancies and replacement collisions, and the vacancies is sum of interstitials with the number of atoms that leave the target.

One file that SRIM creates is of importance to the Activity code, and that is the trajectory file that contains the energy and x,y,z co-ordinate data points for simulated ions moving through matter. Figure 4.10 shows the trajectory of one hundred 13MeV protons entering and passing through an Iron target, and it is this set of data points (together with the cross section database) that the Activity code uses to calculate the reaction rates for the transmutation of nuclei in the target. At higher energies, the ions slow as they lose energy due to electronic stopping, but as the ion energy drops the mechanism of loss through nuclear collisions becomes important. The spreading of ion depths at lower energies is a result of the higher momentum transfer during nuclear collisions, as can be seen in Figure 4.10.

Chapter 5

Background: Interatomic Potential Fitting

In order to model Fe and Pd with Molecular Dynamics codes, an interatomic potential is needed. This will require experimental data and data generated by first principles calculations. The simplified model will consist of just Fe and Pd, but pure Fe does not take the FCC structure that it does when alloyed with Nickel in Austenitic Stainless Steel.

The properties of theoretical pure FCC Fe are estimated using Density Functional Theory that solves the many body Schrödinger Equation. While it is impossible to solve this equation exactly, with our current technology and knowledge, it is possible to solve approximately by making several approximations and by using the Hohenberg-Kohn theorem.

Many bodey potentials such as the FS and embedded-atom method are often used to model metals. The force matching method is used to fit the potentials to data. Optimization algorithms are split into global and local, with an example of a global being simulated annealing and a local being BFGS.

5.1 Experiment, Modelling and Theory

5.1.1 Introduction

We do not have a complete understanding of the world around us, so it may be that experiment is the “best” method of testing a material, as by its nature it will obey the rules of physics. As far as we know, these do not change from one place to another. A piece of steel used in an experiment is never going to have precisely the same composition and structure as similar steel use in a reactor, but it should be close enough.

The scanning electron microscope has a magnification of approximately 1 million times and they are able to generate a picture of the surface of a specimen using a beam of electrons. The transmission electron microscope has the ability to magnify a specimen further, with a resolution of less than 1 angstrom. The trade off is that a it covers a smaller area of the specimen than a SEM and the sample must be prepared in thin slices.

The composition of an irradiated sample may be tested by a number of methods including neutron activation analysis and secondary-ion mass spectrometry. Scattering techniques, such as x-ray or neutron diffraction, can give an insight into the structure of a material, but they are statistical methods and do not give detailed measurements of individual atoms or defects.

There are draw backs to experiments, several of which are particular to this work. Even with our advances in technology, there are still errors introduced into any experiment. When dealing with smaller and smaller volumes of atoms quantum effects will also need to be considered, but this is perhaps a difficultly in our inability to visualise what happens on such small scales.

Radiation damage experiments, whether by neutrons or ions, will result in radioactive waste and the target must be safe to handle to be examined. The collision event happens up to a micron or more into the material. They occur on a very short timescale and within small volumes of space at random points along the path of the projectile, so the events would be very hard or impossible to track as they occur with current technology.

Current theories have been developed and modified over hundreds of years, and in the last 100 years Quantum Mechanics has been introduced to better describe physics on a small scale. It is a well proven theory that has been very successful. Unfortunately, using the theory exactly for a large ensemble of atoms and electrons is impossible with our current level of technology.

Computer models use theory, and a number of approximations, to simulated reality. These models may need to be fine tuned, to better fit both the theory and experiment, but they allow us to do what is not possible with either experiment or theory. It is possible to simulate an individual damage cascade and watch it in fine detail spatially and temporally. As computer power increases, the models used become more complex and the simulation boxes and time spans grow.

This chapter discusses the background required to link the more accurate DFT to MD by deriving interatomic potentials based on experiment and calculated data. By doing this, the concentrations at the grain boundary for an iron-palladium alloy may be modelled following irradiation damage.

5.1.2 Simulating Materials on a Variety of Scales in Time and Space

Taking a 1 micron grain in a steel as an example, it would contain tens of billions of atoms. There are a wide range of modelling programs available (fig. 5.1).

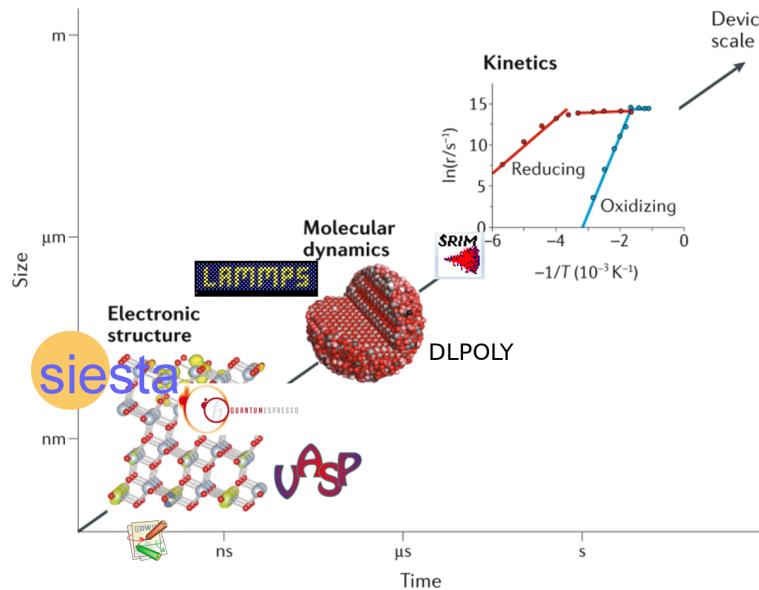


Figure 5.1: Time and Size Scales for Computer Packages [84]

The properties of metals may be approximated using small crystals, and this is ideally suited to DFT calculations. Simulations of small ensembles of atoms over time are now also possible using DFT molecular dynamics. Larger simulations containing thousands to millions of atoms, that are able to simulate single damage cascades are suitable for MD simulations.

5.2 Properties of Crystals

5.2.1 Introduction

There are seven crystal classes (table. 5.1) and 14 Bravias lattices (section 5.6.5), although this work is only concerned with cubic and orthorhombic crystals.

Class	Lengths	Angles
Cubic	$a = b = c$	$\alpha = \beta = \gamma = 90^\circ$
Hexagonal	$a = b, c$	$\alpha = \beta, \gamma = 120^\circ$
Rhombohedral	$a = b = c$	$\alpha = \beta = \gamma \neq 90^\circ$
Tetragonal	$a = b, c$	$\alpha = \beta = \gamma = 90^\circ$
Orthorhombic	a, b, c	$\alpha = \beta = \gamma = 90^\circ$
Monoclinic	a, b, c	$\alpha = \beta = 90^\circ, \gamma \neq 90^\circ$
Triclinic	a, b, c	$\alpha, \beta, \gamma,$

Table 5.1: Seven Crystal Classes

Metals are rarely single crystals, but are made from many crystals separated by grain boundaries. However, the knowledge of the microscopic crystals translates well to the properties of the metals on a macroscopic scale.

This work does not attempt to calculate properties based on a collection of many crystals, but on a single crystal with less than a thousand atoms. The crystals are then assumed to be infinite in size, due to periodic boundary conditions.

5.2.2 Equation of State

The equation of state of a material relates either the pressure on that material as a function of the volume, or the energy of a sample of a material to the volume. This not only allows one to predict the energy or pressure at a certain volume, but also the minimum energy, relaxed volume and the bulk modulus.

Bulk Modulus

The bulk modulus of a material is defined as the bulk stress of a sample divided by the bulk strain on that sample and several example values are given in table 5.2. It is also the inverse of the compressibility of that material, which means that materials with a higher bulk modulus are less compressible than those with a lower value.

$$B_0 = -V \frac{\partial P}{\partial V} \quad (5.1)$$

$$B_0 = V \frac{\partial^2 E}{\partial V^2} \quad (5.2)$$

Material	Bulk Modulus/GPa
Aluminium	70
Iron (BCC)	110
Stainless Steel 18-8	163

Table 5.2: Example bulk modulii for three metals

Murnaghan Equation of State

Hooke's law implies a linear relationship between stress and strain. In practice, where a pressure is applied to a material, the application of Hooke's law is limited[85]. Muraghan derived a new equation to improve upon formulae developed in the 1930's, using compression data from high pressure experiments.

$$P(V) = \frac{B_0}{B'_0} \left(\left(\frac{V_0}{V} \right)^{B'_0} - 1 \right) \quad (5.3)$$

As pressure is the negative derivative of the internal energy of the system with respect to change in volume, $p = -(\partial E/dV)$, and the equation can be integrated and written in terms of the energy, volume, bulk modulus and its derivative[86].

$$E(V) = E_0 + \frac{B_0 V}{B'_0} \left[\left(\frac{V_0}{V} \right)^{B'_0} \frac{1}{B'_0 - 1} + 1 \right] - \frac{B_0 V_0}{B'_0 - 1} \quad (5.4)$$

Birch-Murnaghan Equation of State

Several years after Murnaghan's equation, Birch developed the equation of state further upon the experimental data provided by work from Bridgman in high pressure physics. For cubic symmetry, the description of free energy now includes third order terms in the strain components[87].

$$P(V) = \frac{3B_0}{2} \left[\left(\frac{V_0}{V} \right)^{\frac{7}{3}} - \left(\frac{V_0}{V} \right)^{\frac{5}{3}} \right] \left[1 + \frac{3}{4}(B'_0 - 4) \left(\left(\frac{V_0}{V} \right)^{\frac{2}{3}} - 1 \right) \right] \quad (5.5)$$

The energy-volume relationship may again be constructed[86].

$$E(V) = E_0 + \frac{9V_0 B_0}{16} \left[\left[\left(\frac{V_0}{V} \right)^{\frac{2}{3}} - 1 \right]^3 B'_0 + \left[\left(\frac{V_0}{V} \right)^{\frac{2}{3}} - 1 \right]^2 \left[6 - 4 \left(\frac{V_0}{V} \right)^{\frac{2}{3}} \right] \right] \quad (5.6)$$

In order to fit the Birch-Murnaghan, the first step is to fit a second order polynomial to the energy-volume data. This may be achieved using least-squares fitting with a vandermonde matrix. The coefficients from this polynomial may then be used to calculate reasonable value for E_0 , V_0 and B_0 ; sane starting values of B'_0 are between 1 and 10, and the code takes a starting value of 2[88].

$$\begin{aligned}
E(V) &= c_0 + c_1 V + c_2 V^2 \\
V_0 &= -\frac{c_1}{2c_2} \\
E_0 &= c_2 * V_0^2 + c_1 V_0 + c_0 \\
B_0 &= 2c_2 V_0 \\
B'_0 &= 2
\end{aligned} \tag{5.7}$$

Newton Gauss is then used to minimise E_0 , V_0 and B_0 while $B'_0 \in 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0$

$$[J^T J] P = J^T R \tag{5.8}$$

The parameters with the lowest residual square sum are returned.

5.2.3 Voigt Notation

Where a tensor is symmetric, Voigt notation is used to simplify how the tensor is written. It reduces a second order tensor such as stress or strain, represented by a 3x3 matrix, to a 6 row matrix. It also reduces a fourth order tensor, such as the compliance or stiffness tensor (represented by a 3x3x3x3 matrix), to a 6x6 matrix.

$$\vec{A} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} A_{11} \\ A_{22} \\ A_{33} \\ A_{23} \\ A_{13} \\ A_{12} \end{bmatrix} \text{ if } \vec{A} \text{ is symmetric} \tag{5.9}$$

5.2.4 Elastic Constants

Stress and strain are both related by either the stiffness tensor C , filled with the elastic constants, the compliance tensor S and this is the inverse of the stiffness tensor. Strain is a measure of the deformation of a material and by definition has no units. In one dimension $\epsilon = \frac{\delta l}{l_0}$ where l_0 is the unstrained length and δl is the change in length. The Generalized Hooke's law relating these tensors allows the computation of the Cauchy stress if the elastic constants are known and an input strain is provided.

$$\vec{A} = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} & \epsilon_{13} \\ \epsilon_{21} & \epsilon_{22} & \epsilon_{23} \\ \epsilon_{31} & \epsilon_{32} & \epsilon_{33} \end{bmatrix} = \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \epsilon_{23} \\ \epsilon_{13} \\ \epsilon_{12} \end{bmatrix} \text{ where } \epsilon_{ij} \equiv \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{5.10}$$

$$\vec{\sigma}_{ij} = \vec{C}_{ijkl} \vec{\epsilon}_{kl} \tag{5.11}$$

$$\vec{\epsilon}_{ij} = \vec{S}_{ijkl} \vec{\sigma}_{kl} \quad (5.12)$$

$$\epsilon_{ij} = \begin{bmatrix} \epsilon_1 = \epsilon_{11} = \epsilon_{11} \\ \epsilon_2 = \epsilon_{22} = \epsilon_{22} \\ \epsilon_3 = \epsilon_{33} = \epsilon_{33} \\ \epsilon_4 = \epsilon_{23} = \epsilon_{32} \\ \epsilon_5 = \epsilon_{13} = \epsilon_{31} \\ \epsilon_6 = \epsilon_{12} = \epsilon_{21} \end{bmatrix} \quad (5.13)$$

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{45} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{55} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix} \quad (5.14)$$

The number of independent elastic constants changes depending on the symmetry of the crystal. The tensors in Voigt notation are given for cubic, orthorhombic/orthotropic, monoclinic and triclinic in appendix ??.

5.2.5 Calculating Elastic Constants for a Cubic Crystal

Cubic crystals are the most simple class with primitive unit cells having three orthonormal basis vectors. Four of the most common variants of the cubic crystal are the simple cubic, body centered cubic, face centered cubic and zincblende. Due to symmetry, a cubic crystal has only three elastic independent constants; in Voigt notation these are C_{11} , C_{12} and C_{44} .

Applying two strains to a cubic crystal [89] coupled with the calculation of the Bulk Modulus, as already discussed, allows the three independent elastic constants to be calculated.

First, the bulk modulus may be calculated either by finding the second derivative of the energy wrt volume at the relaxed volume, $B(V) = -VP'(V) = VE''(V)$ [89], or by fitting the Birch-Murnaghan equation of state.

$$\epsilon_{(C_{11}-C_{22})} = \begin{bmatrix} \delta & 0 & 0 \\ 0 & -\delta & 0 \\ 0 & 0 & \delta^2/(1-\delta^2) \end{bmatrix} \quad (5.15)$$

Second, a volume conserving orthorhombic strain (eq. 5.15) may be applied to the crystal [89]. The relation between the relaxed energy and that under strain is symmetric about $E(0)$ and is given by $E(\delta) = E(-\delta) = E(0) + (C_{11} - C_{12})V\delta^2 + O[\delta^4]$ [89]. V is the volume of the relaxed unit cell and $E(0)$ is the minimum energy. By fitting a polynomial to the energy-strain data, the coefficient for the quadratic term is equal to $(C_{11} - C_{12})V$.

$$\epsilon_{(C44)} = \begin{bmatrix} 0 & \frac{\delta}{2} & 0 \\ \frac{\delta}{2} & 0 & 0 \\ 0 & 0 & \delta^2/(4 - \delta^2) \end{bmatrix} \quad (5.16)$$

Third, a volume conserving monoclinic strain (eq. 5.16) is applied and the relation between the relaxed energy, strained energy and the elastic constant C_{44} is $E(\delta) = E(-\delta) = E(0) + \frac{1}{2}C_{44}V\delta^2 + O[\delta^4]$. Similarly, fitting a polynomial to the strain-energy data points will calculate the value of C_{44} .

Finally, the relation between the bulk modulus, C_{11} and C_{12} , $B_0 = (C_{11} + 2C_{12})/3$ allows the computation of the individual constants eq. 5.17 (this relationship is only for cubic crystals).

$$\begin{aligned} C_{ortho} &= C_{11} - C_{12} \\ C_{11} &= \frac{3B_0 + 2C_{ortho}}{3} \\ C_{12} &= \frac{3B_0 - C_{ortho}}{3} \end{aligned} \quad (5.17)$$

5.2.6 Calculating Elastic Constants for Orthorhombic Crystal

The DFT work here includes Pd and Fe. The natural arrangement of Pd atoms in a pure sample are FCC within a cubic crystal. Pure iron at room temperature is BCC, but this work is interested in austenitic stainless steel where the structure of atoms in the alloy are FCC. When modelling FCC iron using DFT with a non-polarized calculation, the crystal favours a cubic crystal with the atoms fixed in the FCC positions. When a spin-polarized calculation is computed, with magnetization along the z-axis, the crystal becomes slightly tetragonal with one side being 5% larger than the other two.

$$C_{ij} = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{44} \end{bmatrix} \quad (3 \text{ independent values}) \quad (5.18)$$

$$C_{ij} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{13} & C_{23} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \quad (9 \text{ independent values}) \quad (5.19)$$

Once the relaxed crystal basis vectors and atom positions have been determined for the orthorhombic crystal, nine strains are applied to the crystal [90] in order to calculate the nine independent elastic constants.

As a strain is applied to a crystal its energy can change. The volume may also change, although it may be held constant. The relationship between the energy and the strain σ may be represented as a Taylor expansion of the internal energy in powers of the strain tensor[90]. The equation used by Ravindran et al is in a slightly different notation to that used by Mehl[91]. It includes a constant xi to account for the symmetry of σ and the use of Voigt notation.

$$E(V, \sigma) = E(V_0, 0) + V_0 \left(\sum_i \tau_i \xi_i \sigma_i + \frac{1}{2} \sum_{ij} c_{ij} \sigma_i \xi_i \sigma_j \xi_j \right) + O(\sigma^3)$$

where σ is the strain
and τ is the stress
if the index k of ξ is $\in 1, 2, 3$ then $\xi_k = 1$
if the index k of ξ is $\in 4, 5, 6$ then $\xi_k = 2$

(5.20)

The first three strains applied to the orthorhombic crystal are non-volume conserving. They strain the crystal along each axis and allow the calculation of C_{11} , C_{22} and C_{33} .

$$D_1 = \begin{bmatrix} 1 + \delta & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.21) \quad E(V, \sigma) = E(V_0, 0) + V_0 \left(\tau_1 \sigma + \frac{c_{11}}{2} \sigma^2 \right) \quad (5.22)$$

$$D_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 + \delta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.23) \quad E(V, \sigma) = E(V_0, 0) + V_0 \left(\tau_2 \sigma + \frac{c_{22}}{2} \sigma^2 \right) \quad (5.24)$$

$$D_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 + \delta \end{bmatrix} \quad (5.25) \quad E(V, \sigma) = E(V_0, 0) + V_0 \left(\tau_3 \sigma + \frac{c_{33}}{2} \sigma^2 \right) \quad (5.26)$$

Volume conserving monoclinic distortions (eq. 5.27, eq. 5.29, eq. 5.31) are used to calculate the C_{44} , C_{55} and C_{66} elastic constants (eq. 5.28, eq. 5.30, eq. 5.32).

$$D_4 = \begin{bmatrix} \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & 0 \\ 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} \\ 0 & \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \quad E(V, \sigma) = E(V_0, 0) + V_0 \left(2\tau_4 \sigma + 2 \frac{c_{44}}{2} \sigma^2 \right) \quad (5.27)$$
(5.28)

$$D_5 = \begin{bmatrix} \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} \\ 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \quad E(V, \sigma) = E(V_0, 0) + V_0 \left(2\tau_5 \sigma + 2 \frac{c_{55}}{2} \sigma^2 \right) \quad (5.29)$$
(5.30)

$$D_6 = \begin{bmatrix} \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ 0 & 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \quad E(V, \sigma) = E(V_0, 0) + V_0 \left(2\tau_6 \sigma + 2 \frac{c_{66}}{2} \sigma^2 \right) \quad (5.32)$$

(5.31)

Finally, three orthorhombic strains (eq. 5.33, eq. 5.35, eq. 5.37) that conserve the volume, by altering the strain along each axis, are used to calculate the C_{12} , C_{13} and C_{23} elastic constants (eq. 5.34, eq. 5.36, eq. 5.38).

$$D_7 = \begin{bmatrix} \frac{1+\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & 0 \\ 0 & \frac{1-\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ 0 & 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \quad E(V, \sigma) = E(V_0, 0) + V_0 \left((\tau_1 - \tau 2\sigma + \frac{1}{2}(c_{11} + c_{22} - 2c_{12})\sigma^2) \right) \quad (5.34)$$

(5.33)

$$D_8 = \begin{bmatrix} \frac{1+\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & 0 \\ 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ 0 & 0 & \frac{1-\sigma}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \quad E(V, \sigma) = E(V_0, 0) + V_0 \left((\tau_1 - \tau 3\sigma + \frac{1}{2}(c_{11} + c_{33} - 2c_{13})\sigma^2) \right) \quad (5.36)$$

(5.35)

$$D_9 = \begin{bmatrix} \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & 0 \\ 0 & \frac{1+\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ 0 & 0 & \frac{1-\sigma}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \quad E(V, \sigma) = E(V_0, 0) + V_0 \left((\tau_2 - \tau 3\sigma + \frac{1}{2}(c_{22} + c_{33} - 2c_{23})\sigma^2) \right) \quad (5.38)$$

(5.37)

5.2.7 Stability Conditions

Whilst the elastic constants may be computed using first-principles calculations, there are a series of assumptions and approximations made in order to solve first-principles calculations in a reasonable amount of time. There must be a sanity check, as it is no use to compute the elastic constants from DFT to find the crystals are unstable.

For a cubic crystal with three independent elastic constants and zero stress, the Born elastic stability criteria apply[92]. The bulk modulus should be positive $(C_{11} + 2C_{12})/3 > 0$, the tetragonal shear should be positive $C_{11} - C_{12} > 0$ and the shear modulus should be positive $c_{44} > 0$.

For an orthorhombic crystal, there are additional conditions given there are nine independent elastic constants (eq. 5.39)[93].

$$\begin{aligned} C_{11} > 0, C_{22} > 0, C_{33} > 0, C_{44} > 0, C_{55} > 0, C_{66} > 0 \\ (C_{11} + C_{22} - 2C_{12}) > 0, (C_{11} + C_{33} - 2C_{13}) > 0, (C_{22} + C_{33} - 2C_{23}) > 0 \\ (C_{11} + C_{22} + C_{33} + 2C_{12} + 2C_{13} + 2C_{23}) > 0 \end{aligned} \quad (5.39)$$

5.2.8 Computing Values from Elastic Constants

The bulk modulus, as noted earlier, is a measure of the effect of strain on stress. While it may be calculated by taking the second derivative of energy with respect to volume, or by fitting an equation of state, it may also be calculated using the elastic constants of a material or the compliance constants.

For cubic crystals the elastic constants may be used to calculate the bulk modulus, tetragonal shear, shear modulus and Cauchy pressure (eq. 5.40).

$$\begin{aligned} B_0 &= \frac{C_{11} + 2C_{12}}{3} = \frac{5C_{11}}{9} \text{ Bulk Modulus} \\ C' &= \frac{C_{11} - C_{12}}{2} \text{ Tetragonal Shear} \\ &\quad C_{44} \text{ Shear Modulus} \\ p_c &= C_{12} - C_{44} \text{ Cauchy Pressure} \end{aligned} \tag{5.40}$$

For an orthorhombic crystal, the Voigt bulk modulus (eq. 5.41) and Reuss bulk modulus (eq. 5.42) may be calculated from the stiffness and compliance matrix respectively. These two values are then averaged to give the bulk modulus (eq. 5.43)[90].

$$B_V = \frac{1}{9} (C_{11} + C_{22} + C_{33} + 2(C_{12} + C_{13} + C_{23})) \tag{5.41}$$

$$B_R = (S_{11} + S_{22} + S_{33} + 2(S_{12} + S_{13} + S_{23}))^{-1} \tag{5.42}$$

$$B = 0.5(B_V + B_R) \tag{5.43}$$

Calculation by this method is particularly useful for this work, where the FCC iron crystal is orthorhombic, and not cubic (section ??).

Elastic constants are also used to calculate the shear modulus G . Where the crystal is cubic, there are several elastic constants that can be used (eq. 5.44)[90]. Both the Voigt (eq. 5.45) and Reuss (eq. 5.45) values are calculated and averaged to give the value of G (eq. 5.47).

$$G = C_{44} = 0.5(C_{11} - C_{12}) = \frac{C_{11}}{3} \tag{5.44}$$

$$G_V = \frac{1}{15} (C_{11} + C_{22} + C_{33} - C_{12} - C_{13} - C_{23}) + \frac{1}{5} (C_{44} + C_{44} + C_{55}) \tag{5.45}$$

$$G_R = \frac{15}{4(S_{11} + S_{22} + S_{33}) - 4(S_{12} + S_{23} + S_{23}) + 3(S_{44} + S_{55} + S_{66})} \tag{5.46}$$

$$G = 0.5(G_V + G_R) \quad (5.47)$$

The Young's modulus E as a scalar is a measure of how easily the length an isotropic material is changed under tension or compression, and the computed elastic constants for a cubic crystal may be used to calculate E (eq. 5.48).

$$E = \frac{9B_0G}{3B_0 + G} \text{ where } B_0 \text{ is the bulk modulus and } G \text{ is the shear modulus} \quad (5.48)$$

The Poisson ration is a measure of how much a material changes in the direction perpendicular to a strain applied transversely.

$$\nu = -\frac{d\epsilon_{perp}}{d\epsilon_{transverse}} \quad (5.49)$$

The bulk modulus may be calculated from the equation of state or elastic constants, and the shear modulus may be calculated from the elastic constants. In turn, the Poisson ratio is calculated from these.

$$\nu = \frac{3B_0 - 2G}{2(3B + G)} \quad (5.50)$$

The ratio may be used as another data point in order to fit a potential, or at the very least as a check to compare to the known value once a potential has been derived.

5.2.9 Correlation of Melting Temperature in Metals with Elastic Constants

There is a correlation between the melting temperature of metals and their elastic constants detailed in eq. 5.51 and fig. 5.2) [94].

$$T \approx 598.0 + 6.66 \times (C_{11} + C_{22} + C_{33}) - 0.003 \times (C_{11} + C_{22} + C_{33}) \quad (5.51)$$

Where the temperature is in K, and the elastic constants are in GPA

The correlation values between the temperature and C_{11} are 0.848 and 0.780 for the Pearson and Spearman's correlation respectively. An accurate temperature will not be possible to predict, but it will act as a further sanity check within the computer codes developed in the results section of this work.

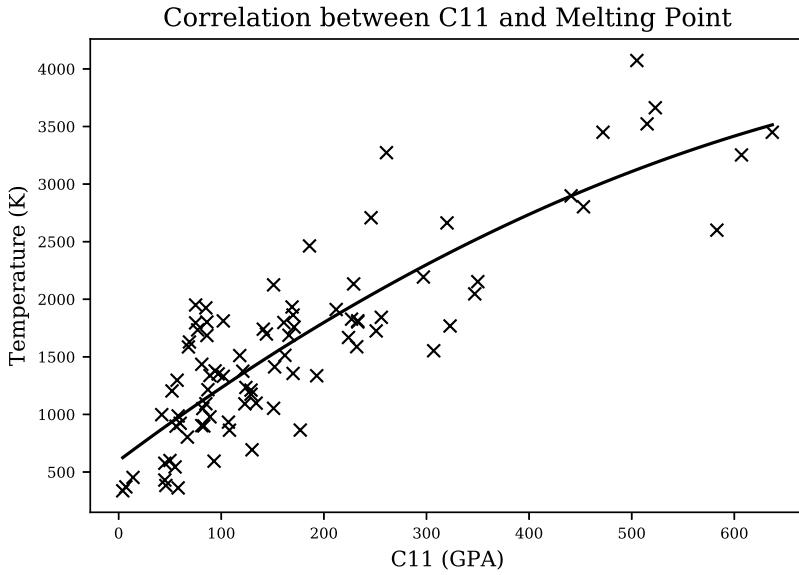


Figure 5.2: Correlation between C_{11} and temperature

5.3 Interatomic Potentials

5.3.1 Introduction

An interatomic potential, as used in this work and Molecular Dynamics computer codes, is a function or a set of functions that describe the energy and force between atoms. Simpler functions represent the potential energy between pairs of atoms only, but more complicated functions have been used in molecular dynamics since the 1980s that also attempt to represent the many body nature of materials, which applies in particular to metals.

5.3.2 Pair Potentials

A pair potential only considers individual pairs of nearby atoms, one pair at a time, and does not consider the effect of any other nearby atoms. Where an alloy is being modelled, there will be a pair potential function for each element and element combination; 1 for a single element, 3 for a two element alloy, 6 for a three element alloy and so on.

Lennard-Jones

The Lennard-Jones potential was proposed in the 1920s and has both an attractive term and a repulsive term; the $(r_m/r)^{12}$ becomes much larger than the $(r_m/r)^6$ term at close distances, and this mimics the coulomb repulsion as two atoms are pushed closer together. At larger separations, the attractive term dominates.

$$V(r) = e \left(\left(\frac{r_m}{r} \right)^{12} - 2 \left(\frac{r_m}{r} \right)^6 \right) \quad (5.52)$$

Lennard-Jones Potential

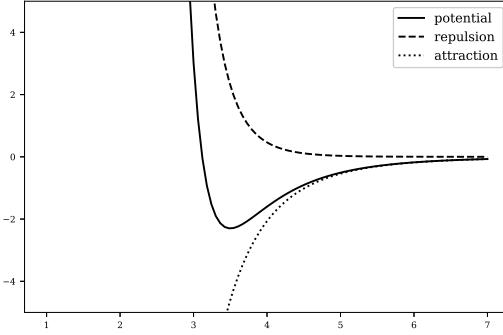


Figure 5.3: Lennard-Jones Potential

Morse Potential

The Morse potential was proposed five years after the Lennard-Jones potential. It also has an attractive and repulsive term, but it uses the exponential function rather than 6th and 12th powers.

$$V(r) = \exp(-2a(r - re)) - 2\exp(-a(r - re)) \quad (5.53)$$

Morse Potential

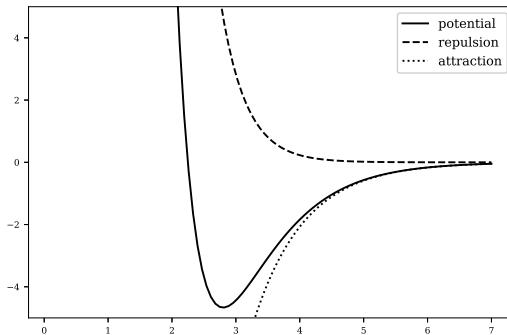


Figure 5.4: Morse Potential

Buckingham Potential

The Buckingham Potential consists of a repulsive and attractive part. As the separation decreases, the attractive term dominates and the overall function tends towards negative infinity. This shouldn't be problematic when gasses or solids are being modelled alone, but when collisions and damage cascades are being modelled, the separation between atoms may be much smaller than that in a typical simulation, ending with these atoms being pulled together (eq. 5.54 and fig. 5.5).

$$V(r) = A\exp(-Br) - \frac{C}{r^6} \quad (5.54)$$

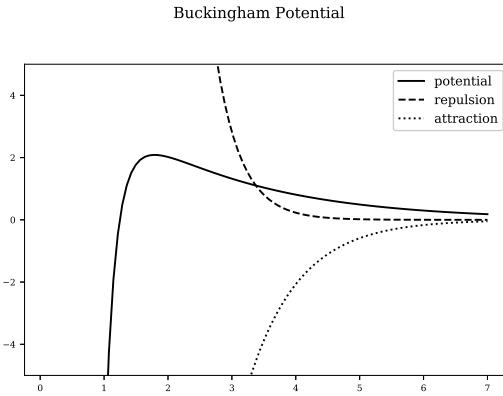


Figure 5.5: Buckingham Potential

Ziegler-Biersack-Littmark Universal Potential Function

It became clear, while experimenting with a number of existing potentials and molecular dynamics programs, that at the very least modifications to those potentials would need to be made for small atom separations. A simulation to model a projectile failed early on due to the projectile's proximity to target atoms, resulting in the MD code returning an error as the separation was out of the range of the potential.

The Ziegler-Biersack-Littmark potential, between an atom of charge Z_i and Z_j is set out in the SRIM manual[82].

$$V_{zbl}(r_{ij}, Z_i, Z_j) = \frac{1}{4\pi\epsilon_0} \frac{Z_i Z_j}{r_{ij}} \phi\left(\frac{r_{ij}}{a_{ij}}\right) \quad (5.55)$$

where $\epsilon_0 = 8.85419 \times 10^{-12}$

The parameters of the Universal screening potential function are as follows:

$$\phi(x) = 0.181e^{-3.2x} + 0.5099e^{-0.9423x} + 0.2802e^{-0.4029x} + 0.02817e^{-0.2016x} \quad (5.56)$$

where $a_{ij} = \frac{0.8854a_0}{Z_i^{0.23} + Z_j^{0.23}}$

and $a_0 = 0.529$ angstrom

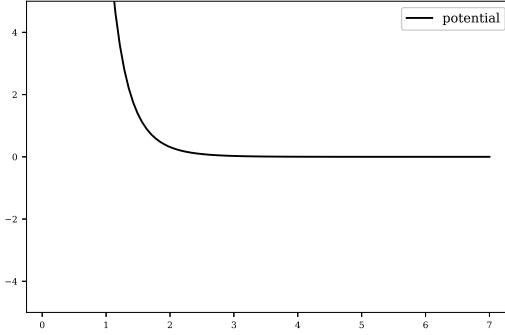


Figure 5.6: ZBL universal screening potential

5.3.3 Many Body Potentials

This work is focused on metals, and two popular, and closely related potentials, will be discussed in this section.

Finnis-Sinclair

The Finnis-Sinclair potential was published in 1984[95] and it introduced both a pair potential and an embedded term to take into account the cohesive energy dependent on the local electron density. The pair term represents the repulsion between the atoms whereas the embedding functional glues the atoms together in the solid.

The tight-binding that this potential is based on is represented by the functional: square root of the density. The potential for a single element model consists of a pair function, a density function and a tight-binding functional.

$$\begin{aligned}
 u^{A,A} &= \sum_{i \neq j}^N(r) v^{A,A}(r_{ij}) \\
 \rho_i^A(r) &= \sum_{j,j \neq i}^N \phi(r_{ij}) \\
 F^A &= \sum_i^N \rho_i
 \end{aligned} \tag{5.57}$$

Embedded-atom method

The embedded-atom method was also invented in the 1980s. It was developed with metals in mind, and in many ways is a more flexible variant of the Finnis-Sinclair potential. It too has a pair and density function, but the functional of the EAM potential is not restricted to a square root.

$$\begin{aligned}
 U_{EAM} &= \frac{1}{2} \sum_{i=1}^N \sum_{j \neq i}^N V_{ij}(r_{ij}) + \sum_{i=1}^N F[\rho_i] \\
 \text{where } \rho_i &= \sum_{j=i, j \neq i}^N \rho_{ij}(r_{ij})
 \end{aligned} \tag{5.58}$$

Elastic properties were not reliably calculated from pair functions alone[96], but the advent of Finnis-Sinclair and EAM type potentials changed this.

$$A_{ij} + F'(\vec{\rho})V_{ij} = 0$$

where

$$\begin{aligned} A_{ij} &= \frac{1}{2} \sum_m \phi'_m a_i^m a_j^m / a^m \\ V_{ij} &= \frac{1}{2} \sum_m \rho'_m a_i^m a_j^m / a^m \end{aligned} \quad (5.59)$$

$$C_{11} = (B_{11} + F'(\vec{\rho})W_{11} + F''(\vec{\rho})(V_{11})^2) / \Omega_0$$

$$C_{12} = (B_{12} + F'(\vec{\rho})W_{12} + F''(\vec{\rho})(V_{11})^2) / \Omega_0$$

$$C_{44} = (B_{12} + F'(\vec{\rho})W_{12}) / \Omega_0$$

where

$$\begin{aligned} B_{ijkl} &= \frac{1}{2} \sum_m (\phi''_m - \phi'_m/a^m) a_i^m a_j^m a_k^m a_l^m / (a^m)^2 \\ W_{ijkl} &= \frac{1}{2} \sum_m (\rho''_m - \rho'_m/a^m) a_i^m a_j^m a_k^m a_l^m / (a^m)^2 \end{aligned} \quad (5.60)$$

where

$$\phi''_m = (d^2\phi(r)/dr^2)_{r=a^m},$$

$$\rho''_m = (d^2\rho(r)/dr^2)_{r=a^m}.$$

For homonuclear cubic crystals, the lattice constant may be calculated by the equilibrium condition in eq 5.59. The three independent elastic constants, C_{11}, C_{12}, C_{44} , may also be calculated at equilibrium from eq 5.60. There are notable consequences in removing either the pair function or embedding functional. If $V_{ij}(r_{ij})$ is removed, $F'[\vec{\rho}] = 0$ and this gives a shear modulus of 0 ($C_{11} = C_{12}, C_{44} = 0$)[96]. If $F[\vec{\rho}]$ is removed the Cauchy discrepancy becomes zero ($C_{12} = C_{44}$)[96]. Both of these situations are generally untrue, and both functions are required for a good potential for a metal.

Two band embedded-atom method

There are several variations of the potential, and one of particular interest to us is the two band embedded-atom method that has two electron density and embedding energy terms. This formalism was originally developed to model Caesium[97], and the transition of electrons between S and D bands under pressure, but it has been modified to apply to alloys.

Caesium changes its structure as pressure is applied. The first change is from BCC to a more compact FCC structure, but it then compresses further as electrons move from the S-band to the more compact D-band, reducing the size of the atom.

The bond energy may be described with eq. 5.61 where N_1 and N_2 are the capacities of each band, n_{i1} and n_{i2} are the electrons in each band of the i^{th} atom. E_{prom} is the energy required to promote an electron from band 1 to band 2.

$$U_{\text{bond}} = \sum_i \frac{W_{i1}}{2N_1} n_{i1} (n_{i1} - N_1) + \sum_i \frac{W_{i2}}{2N_2} n_{i2} (n_{i2} - N_2) + E_{\text{prom}} \quad (5.61)$$

A Finnis-Sinclair type EAM potential was used by Ackland and Reed in this work for both bands. The parameters fitted in Ackland and Reed's work are listed in table 5.3.

$$\begin{aligned} & \text{Pair} \\ V_s(r_{ij}) &= \sum_i \frac{A_s}{r_{ij}^{12}} \\ V_d(r_{ij}) &= \sum_i \frac{A_d}{r_{ij}^{12}} \end{aligned} \quad (5.62)$$

$$\rho_s(r_{ij}) = \begin{cases} C_s(d_s - r_{ij})^3 & r_{ij} < d_s \\ 0 & r_{ij} > d_s \end{cases} \quad \rho_d(r_{ij}) = \begin{cases} C_d(d_d - r_{ij})^3 & r_{ij} < d_d \\ 0 & r_{ij} > d_d \end{cases} \quad (5.63)$$

$$\text{Embedding} \quad (5.64)$$

Parameter	Value
C_s	$0.05617 eV^2 \text{angs}^{-3}$
C_d	$0.1681 eV^2 \text{angs}^{-3}$
d_s	9.5097angs
d_d	6.9189angs
A_s	$2.417 \times 10^7 \text{angs}^{12}$
A_d	$3.7668 \times 10^6 \text{angs}^{12}$

Table 5.3: Caesium 2BEAM parameters

The implementation of this two-band potential predicted the transformation of Caesium. The I phase, at ambient pressure, is BCC Caesium with a primitive cell volume of 115.9 cubic angstrom per atom. As more pressure is added, the optimum structure changes from BCC to FCC, and this results in a smaller volume per atom of 67.5 cubic angstrom. Finally Caesium undergoes an isostructural transformation, and the potential predicts a volume of 48.7 cubic angstrom per atom.

There is a transition pressure of 4.3 GPa between the phases II and III. The potentials developed by Ackland and Reed were "in good agreement with experiment" [97].

An alloy version of the two-band model was developed by Olsson et al. to investigate the α -prime phase formation in Fe-Cr[98]. Fe-Cr alloys form ferromagnetic alloys with concentrations of up to 10% chromium at 750°C . As the concentration of chromium in the alloy increases, the alloy begins to decompose into iron rich and chromium rich precipitates, and this decomposition is accelerated under irradiation.

The two-band method for Caesium was extended to an Fe-Cr alloy in order to describe the heat of mixing in the alloy.

$$\begin{aligned} U_{EAM} &= \frac{1}{2} \sum_{i=1}^N \sum_{j \neq i}^N V_{ij}(r_{ij}) + \sum_{i=1}^N F_D[\rho_{d,i}] + \sum_{i=1}^N F_S[\rho_{s,i}] \\ \text{where } \rho_{d,i} &= \sum_{j=i, j \neq i}^N \rho_{d,ij}(r_{ij}) \text{ and } \rho_{s,i} = \sum_{j=i, j \neq i}^N \rho_{s,ij}(r_{ij}) \end{aligned} \quad (5.65)$$

The embedding functional (eq. 5.69) was in the form of several functionals used in papers by Ackland and Mendelev, and an extension to the standard Finnis-Sinclair.

$$F_{band}(\rho_{band}) = A_1^{band} \sqrt{\rho_{band}} + A_2^{band} \rho_{band}^2 + A_3^{band} \rho_{band}^4 \quad (5.66)$$

The choice of function for both the pair and d-band density functions was a cubic spline, and a 4s Slater type function was used for the s-band alloy density. The Fe-Fe and Cr-Cr s-band density functions were zero functions. Where the alloy has high concentrations of Iron or Chromium, the respective d-band density functions will dominate, but as the elements mix, the s-band density will also contribute.

$$V(r) = \sum_i a_i (r - r_i)^3 H(r_i - r) \text{ where } H \text{ is the Heaviside Step function} \quad (5.67)$$

$$\phi_d^{CrCr}(r) = \sum_k b_k (r - r_k)^3 H(r_k - r) \text{ where } H \text{ is the Heaviside Step function} \quad (5.68)$$

$$\begin{aligned} \phi_s^{FeCr}(r) &= (H_s r^3 \exp(-\xi_s r))^2 \\ \phi_s^{FeFe}(r) &= 0.0 \\ \phi_s^{CrCr}(r) &= 0.0 \end{aligned} \quad (5.69)$$

The resulting potential, with the s-density fitted to the mixing enthalpy of Iron and Chromium, reproduces the formation of α -prime phase Cr with thermal ageing over a range of Cr concentrations.

The Fe-Cr potentials were further developed by Bonny et al. The mixing enthalpy changes sign in Fe-Cr, having a positive mixing enthalpy for Cr concentrations above 10% and a negative mixing enthalpy below. This results in the solubility of Cr in the alloy at lower concentrations, but as the concentration rises there's a tendency for Cr to form α -prime precipitates[99].

In the work by Bonny et al, the base Iron and Chromium potentials were those developed by Mendelev and Ackland, and from the Fe-Cr potentials of Olsson and Wallenius. In total, 11 functions make up the Bonny et al potential, higher than the usual 7 functions required for a two species EAM. In a standard EAM potential, the density contribution from an atom of one species is only dependent on that contributing atom, and not the embedded atom. This potential splits the density function into multiple bands and multiple permutations of atom species.

$$\begin{aligned} \rho_{AA}^d(r) &= \rho_{BA}^d = \rho_A^d \\ \rho_{BB}^d(r) &= \rho_{AB}^d = \rho_B^d \\ \rho_{AB}^s(r) &= \rho_{BA}^s \\ \rho_{AA}^s(r) &= \rho_{BB}^s = 0 \end{aligned} \quad (5.70)$$

The chosen s-band density function for the Bonny et al Fe-Cr alloy was selected as an exponential style function with a cut-off function.

$$\rho_{FeCr}^s(r) = kr^6 \exp(-2\xi r) g_{cut}(r) \quad (5.71)$$

$$g_{cut}(r) = \begin{cases} 1 & r \leq r_c^i \\ \frac{1}{2} (1 - \sin(\frac{\pi}{2} \frac{r-r_m}{d})) & r_c^i < r < r_c^f \\ 0 & r_c^r < r \end{cases} \quad (5.72)$$

$$F^s(\rho) = A_1\sqrt{\rho} + A_2\rho^2 \quad (5.73)$$

The magnetic properties of Chromium were considered during the development of these potentials. First-Principles calculations are equivalent to calculating properties at 0K, and at this temperature Chromium is antiferromagnetic. The Neel temperature for Cr, the point at which it transitions from an antiferromagnetic to a paramagnet, is 310K. The operating temperature of the alloys, within a reactor, will be above the Neel temperature. Chromium has a positive Cauchy pressure as a paramagnet, and a negative Cauchy pressure under the Neel temperature when an antiferromagnetic, and as a result of this and the operating temperature, a positive Cauchy pressure was used to fit the potential.

Modified embedded-atom method

All the potential types considered so far are spherically symmetrical about the atom. Typically covalent bonds tend to be directional but metallic bonds, in contrast, are isotropic about each atom. However, in alloys and magnetic materials where the system is not isotropic, the modified embedded atom method may be used.

$$U_{MEAM,i} = \frac{1}{2} \sum_{j \neq i}^N V_{ij}(r_{ij}) + F_i(\vec{\rho}_i) \quad (5.74)$$

The original form for the embedding functional is given below (eq. 5.75)[100].

$$F(\vec{\rho}) = AE_{coh} \frac{r \vec{h} \cdot \vec{o}}{\vec{\rho}^0} \ln \left(\frac{\vec{\rho}}{\vec{\rho}^0} \right) \quad (5.75)$$

The electron density is made from four contributing functions. The first, ρ^0 , is not dependent on direction, but ρ^1 , ρ^2 and ρ^3 are dependent on the angle.

$$\begin{aligned}
\rho^0 &= \sum_i^N \rho_i^0(r^i) \\
(\rho^1)^2 &= \sum_{\alpha} \left(\sum_i \rho_i^1(r^i) \frac{r_{\alpha}^i}{r^i} \right)^2 \\
(\rho^2)^2 &= \sum_{\alpha} \left(\sum_i \rho_i^2(r^i) \frac{r_{\alpha}^i}{r^i} \frac{r_{\beta}^i}{r^i} \right)^2 - \frac{1}{3} \left(\sum_i \rho_i^2(r^i) \right)^2 \\
(\rho^2)^2 &= \sum_{\alpha} \left(\sum_i \rho_i^2(r^i) \frac{r_{\alpha}^i}{r^i} \frac{r_{\beta}^i}{r^i} \frac{r_{\gamma}^i}{r^i} \right)^2 - \frac{3}{5} \left(\sum_{\alpha} \frac{r_{\alpha}^i}{r^i} \rho_i^3(r^i) \right)
\end{aligned} \tag{5.76}$$

The partial electron densities may be combined as is eq. 5.77[100].

$$\begin{aligned}
\vec{\rho}_i &= \rho_i^{(0)} \cdot 2 / (1 + \exp(-\Gamma_i)) \\
\Gamma_i &= \sum_{h=1}^3 t^{(h)} \left[\rho_i^{(h)} / \rho_i^{(0)} \right]^2 \\
\rho^{a(h)}(R) &= \exp(-\beta^{(h)}(R/r_e - 1))
\end{aligned} \tag{5.77}$$

This form of potential is implemented in LAMMPS and other molecular dynamics codes. It is more applicable to materials with bonds that depend on their angle, but it may have useful applications to metals where this is not as pronounced (if at all).

5.3.4 Function Types used by EAM and Two-Band EAM

In previous work, there are a wide range of functions used to represent the pair potential, density and embedding functional. These range from those that preserve and attempt to replicate the physics, to those that do away with knowledge of the physics underlying the potentials altogether. The pair potentials already covered here may also be used as the pair function component of the EAM or Two-Band EAM potentials.

A full list of the functions considered and included in the potential fitting code that resulted from this work is included in appendix ???. Only polynomial splines will be discussed further here.

Polynomial Splines

The form of polynomial spline used in the literature is a sum of N cubic polynomials that, by way of the Heaviside step function and their form, cutoff neatly at the desired radius. The cutoff radii are fixed and, during the fit of a potential, just the coefficients of each cubic spline are varied (eq. 5.82).

$$\begin{aligned}
V(r) &= \sum_i^N a_i (r - r_i)^3 H(r_i - r) \\
H(x) &= \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}
\end{aligned} \tag{5.78}$$

If a continuous second derivative is required the cubic spline may be replaced with a quintic spline (eq. 5.79).

$$V(r) = \sum_i^N a_i (r - r_i)^5 H(r_i - r) \quad (5.79)$$

Polynomial Knot to Knot Splines

So far, analytic potentials have been discussed. There are existing potentials that do not have an analytic form, but are tabulated data that have the properties that they are continuous and have a continuous first derivatives. In attempting to fit or re-fit tabulated data, it would be problematic to adjust each data point individually given the number of data points and the requirement to have a continuous well behaved function with continuous first derivatives.

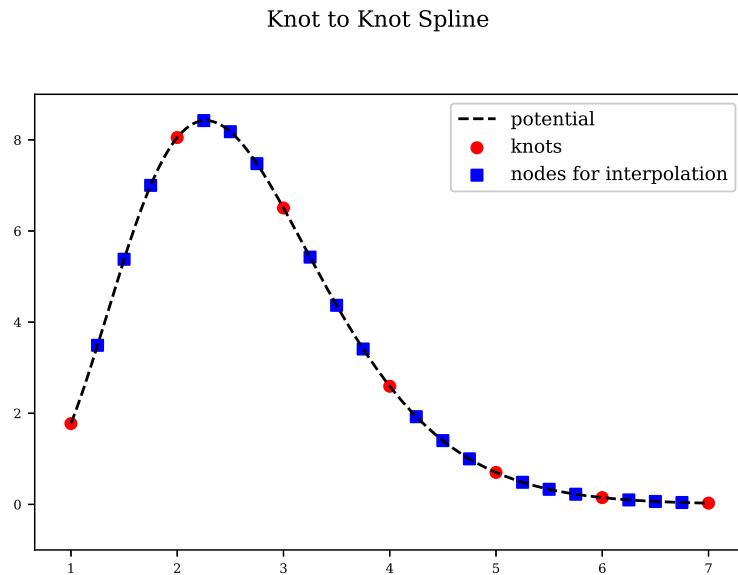


Figure 5.7: Polynomial knot to knot spline

The tabulated function is divided into sections, and the start and end of each section forms the knot of the spline. A polynomial may be splined between these knots, but the order of polynomial will depend on the data available. Immediately, the x and y positions are known: $x_a, x_b, f(x_a), f(x_b)$. A first order polynomial has two unknowns, and so these unknowns may be calculated using linear algebra.

$$\begin{aligned} c_0 + c_1 x_a &= y_a \\ c_0 + c_1 x_b &= y_b \end{aligned} \quad (5.80)$$

One of the requirements for the potential is to have a continuous first derivative, and this requires knowledge of the first derivative at each knot. In the figure, the knots are shown as circles. By taking several points near to each knot, shown as squares, the derivative at each knot may be calculated by interpolation. This allows a third order polynomial to be used (eq. 5.81).

$$\begin{aligned}
P_1 &= (x_A, f(x_A)) \\
P_2 &= (x_A, f'(x_A)) \\
P_3 &= (x_B, f(x_B)) \\
P_4 &= (x_B, f'(x_B))
\end{aligned} \tag{5.81}$$

$$\begin{aligned}
c_0 + c_1 x_a + c_2 x_a^2 + c_3 x_a^3 &= y_a \\
0 + c_1 + 2c_2 x_a + 3c_3 x_a^2 &= y'_a \\
c_0 + c_1 x_b + c_2 x_b^2 + c_3 x_b^3 &= y_b \\
0 + c_1 + 2c_2 x_b + 3c_3 x_b^2 &= y'_b
\end{aligned} \tag{5.82}$$

Rewriting as matrices:

$$\begin{bmatrix} 1 & x_a & x_a^2 & x_a^3 \\ 0 & 1 & 2x_a & 3x_a^2 \\ 1 & x_b & x_b^2 & x_b^3 \\ 0 & 1 & 2x_b & 3x_b^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} f(x_a) \\ f'(x_a) \\ f(x_b) \\ f'(x_b) \end{bmatrix} \tag{5.83}$$

It is important for the function and its derivative to be continuous. If it is also deemed necessary to have a continuous second derivative, a quintic spline may be used. By interpolating and calculating the second order derivative at each knot, a 5th order polynomial may be splined between pairs of knots.

This approach may be modified to spline functions of the form $\exp(p(r))$ between knots, and this is useful for splining between a ZBL or similar repulsive function at small r and a function of a different form at larger r.

where

$$V(r) = \begin{cases} ZBL(r, q_1, q_2) & r \leq r_a \\ \exp(a + br + cr^2 + dr^3) & r_a < r < r_b \\ v(r) & r_b < r < r_{cut} \\ 0 & r \geq r_{cut} \end{cases} \tag{5.84}$$

To spline the exponential the equations are set up in a similar way, but this time the system of equations are non-linear.

$$\begin{aligned}
\exp(c_0 + c_1 x_a + c_2 x_a^2 + c_3 x_a^3) &= y_a \\
(c_1 + 2c_2 x_a + 3c_3 x_a^2) \exp(c_0 + c_1 x_a + c_2 x_a^2 + c_3 x_a^3) &= y'_a \\
\exp(c_0 + c_1 x_b + c_2 x_b^2 + c_3 x_b^3) &= y_b \\
(c_1 + 2c_2 x_b + 3c_3 x_b^2) \exp(c_0 + c_1 x_b + c_2 x_b^2 + c_3 x_b^3) &= y'_b
\end{aligned} \tag{5.85}$$

Newton-Gauss is used to solve this problem where initial parameters are set for the constants in the equation and varying until the function values and derivatives at point A and point B match the values being fitted to.

Tending to Zero at the Cutoff Radius

It is desirable for the functions to be well behaved and continuous. The coulomb force between two charged particles reduces smoothly until theoretically at an infinite separation; it doesn't reach a set separation and abruptly drop to zero. It is impossible and unhelpful to consider a very large or infinite separation which is why the cutoff radius has been introduced. It represents the separation where the potential has reached zero; at the very least it's a trade off between accuracy and the computing time available for the problem.

The pair potential should therefore drop off to 0 at the cut off radius, and be equal to zero for larger separations. The electron density spherically around an atom should also smoothly drop off to zero, and similarly it should be equal to zero for distances larger than the cut off. The embedding energy is dependent on the density at the location of the atom embedded. The embedding function will not necessarily drop off to zero, and it depends on the density and not the separation.

In a molecular dynamics simulation, the neighbour list will usually be built considering atoms slightly more separated than the cut off radius of the functions. This allows the same neighbour list to be used for several time steps, before the need to update as atoms will get closer and further apart with each time step.

Collisions and Interatomic Potentials

In a simulation at room temperature the average energy of the atoms in the simulation are approximately 0.05eV, and at temperatures near the melting point of iron the energy is approximately 0.2eV. These are energy ranges that interatomic potentials are designed to operate in, but in typical damage cascades in nuclear core components the atoms in the cascade, including the PKAs, have energies up to approximately 100KeV.

If a potential is not designed for use in collision simulations, it may not contain the data points required for such close separations. If a potential, such as the Buckingham potential (section 5.3.2), is used, the energy peaks at a certain separation and then drops once more. This would cause colliding atoms to stick to one another once they reached a certain separation.

The ZBL function is designed specifically with collisions in mind and should be used in MD simulations that involve collisions. In simulations of BCC tungsten[101], the material had an initial temperature of 10K and 523K with PKAs energies at 10keV, 20keV and 50keV. The models were simulated with MOLDY using an EAM potential where the pair potential is splined to a ZBL for short range interactions.

5.3.5 Sheng EAM Potentials

A set of potentials have been derived by Sheng et al[102] and are available to download in tabulated form[29]. They were created using a modified version of the PotFit code by P. Brommer and F. Gähler[103]. The code had been modified extensively to include quintic spline interpolation, phonon calculation, elastic constant calculation and optimization techniques[29]. The potentials (typically) used 15 knot splines for pair and density functions and 6 knot splines for the embedding functional for each element, but the potentials were published as tabulated functions. Potentials for fourteen FCC metals were derived in the first instance, but more (including alloys) are available on the website. Unfortunately, the (Sheng et al.) modified PotFit code is not available through the PotFit website[103], and the latest version of their code has moved on since 2011.

5.3.6 Calculating Energy, Force and Stress

Molecular dynamics simulations do not need to include the weak or strong force, and the gravitational force between atoms in the simulated material are so weak in comparison to the electromagnetic force, they can also be ignored. There is a force between all the atoms within a real material, but the electromagnetic force is

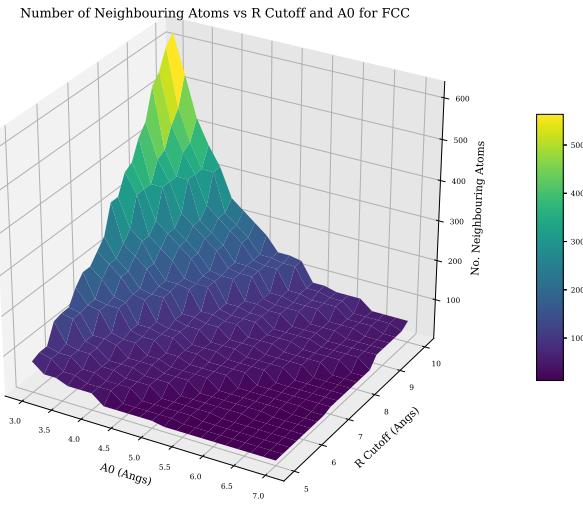


Figure 5.8: Neighbouring atom count dependant upon lattice parameter and cutoff radius

inversely proportionally to the separation of the atoms. Above a certain separation, the electromagnetic force can also be ignored, in order to simplify the computation.

Neighbour List

A cut off radius can be introduced to limit the number of neighbours (section 5.3.4). As the lattice parameter decreases, the atoms are brought closer together, and as the cut off radius is increased more atoms are considered to be within the sphere of influence of one another. Both increase the number of neighbours each atom has (fig. 5.8).

Building a neighbour list may take a long time, depending on the parameters. If a simplistic approach is taken, for N atoms in the supercell, there will be $27N^2$ checks between atoms to see whether they are within the cut off radius of one another. For larger numbers of atoms in a supercell, the whole may be decomposed into smaller domains.

For example, a $16 \times 16 \times 16$ FCC supercell, containing 16,384 atoms, would require looping $27 \times 16,384^2 = 7.25 \times 10^9$ times. Breaking the supercell into 64 smaller domains with 256 atoms in each, reducing the problem to $64 \times 27 \times 256^2 = 1.13 \times 10^8$ loops.

For this work, the supercells used to calculate bulk properties from the interatomic potentials, and the configurations generated by DFT, contain fewer than 1,000 atoms, removing the need to decompose the configuration into smaller sub cells.

To build the neighbour list a halo configuration is created such that it lies on top of the real atoms, but extends at least as far as the cut off radius on each side of the real simulation box (fig. 5.9). Periodic boundary conditions are used to construct the halo.

Once the halo list is computed, the neighbour list may also be computed by looping through the real atoms and halo atoms. A simple pseudo coded sub routine is given (listing 5.1).

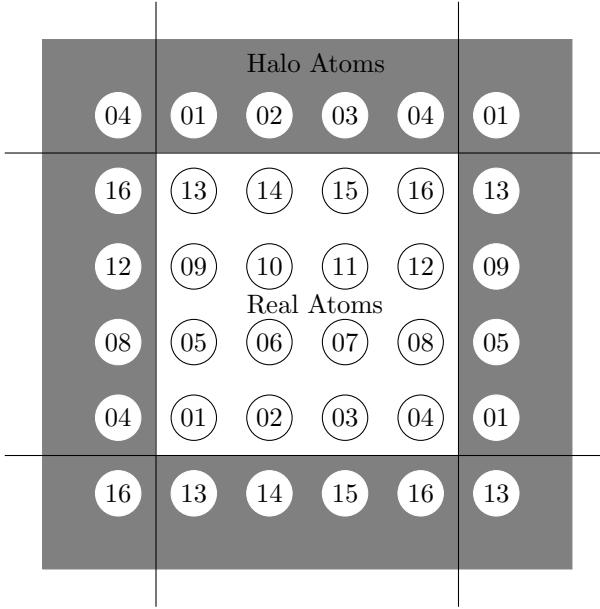


Figure 5.9: Atoms in simulation box with a halo of atoms at the boundary

Listing 5.1: Simple subroutine for generating a half size neighbour list

```

1
2 // real_atoms - array holding coordinates of all real atoms
3 // halo_atoms - array holding coordinates of all halo atoms
4 // real_ids - unique id for each atom
5 // halo_ids - unique atom ids for halo atoms
6 // nlist_ids - array to store ids
7 // nlist_r - array to store separation
8
9 // Define the cutoff and counter start value
10 r_cut = 5.0
11 nl_counter = 1
12
13 // Calculate square
14 r_cut_sq = r_cut ** 2
15
16 // Loop over all real atoms
17 DO n_real = 1, real_atom_count
18 // Loop over all halo atoms
19 DO n_halo = 1, halo_atom_count
20   IF (real_ids(n_real) .LT. halo_ids(n_halo)) THEN
21     r(1:3) = halo_atoms(n_halo, :) - real_atoms(n_real, :)
22     r_sq = SUM(r(1:3) * r(1:3))
23     IF (r_sq .LE. r_cut_sq) THEN
24       r_mag = sqrt(r_sq)
25       nlist_ids(nl_counter, 1) = real_ids(n_real)
26       nlist_ids(nl_counter, 2) = halo_ids(n_halo)
27       nlist_r(nl_counter, 1) = r_mag
28       nlist_r(nl_counter, 2:4) = r(1:3)/r_mag
29       nl_counter = nl_counter + 1
30   END IF
31 END IF
32 END DO
33 END DO

```

The resulting list will contain unique pair combinations; i.e. the pair atom 1 and atom 2 will only be recorded once, and not also as atom 2 and atom 1.

Computing Total Energy

The total energy of the system is the sum of the individual energies for the atoms in the simulation. The type of atom (or pairs of atoms) will determine which function is used.

First, to compute the pair potentials:

- set the total energy of the system equal to zero
- set the starting energy for each atom in the system to zero
- loop through the atom pairs in the entire neighbour list
- for each atom pair, A and B, use the known separation and the potential function to compute the potential energy on atom A due to B and vice versa
- add this potential energy to both A and B
- after looping through the neighbour list, add all the energies due to the pair potential to the total energy

For an EAM or 2BEAM potential, the densities and embedding energies must also be computed:

- set the electron density at the position of (for each) atom to zero
- loop through the atom pairs in the entire neighbour list
- for each atom pair, using the density function for the atom A, the density at atom B due to atom A will be calculated and added to the density at atom B
- if the atoms are both of the same type the same density will be added to atom A due to atom B
- if the atom types are different, using the density function for the atom B, the density at atom A due to atom B will be calculated and added to the density at atom A
- following looping through the neighbour list, to calculate the densities at each atom, the list of atoms will be looped through
- for each atom, the density value at the position of that atom, will be input into the appropriate embedding function to calculate the embedding portion of the energy
- add all the embedding energies to the total energy of the system

For a 2BEAM potential, repeat the above procedure for the second group of density functions and embedding functions (do not repeat calculation of the pair potentials).

Computing Forces on Atoms

In order to calculate the forces on the atoms with an EAM or 2BEAM potential, the neighbour list and atom list will need to be looped through several times. First, the pair potential and force due to the pair potential must be calculated by a complete loop through the neighbour list. At the same time, the density at each atom location is also calculated. The embedding energy may then be computed by looping through all the atoms and using the electron density at each atom to give the energy of the atom embedded in that density. The third and final loop will run through the neighbour pairs in the neighbour list once more computing the force on each atom due to its embedding.

The force on each atom, due to surrounding atoms, may be split into two; force due to the pair potential, and the force due to the embedding of the atom[104].

$$\begin{aligned} F_{pair}^k &= - \sum_{j=1, j \neq k}^N \frac{\partial V_{kj}(r_{kj})}{\partial r_{kj}} \frac{\vec{r}_{kj}}{|\vec{r}_{kj}|} \\ F_{embed}^k &= - \sum_{j=1, j \neq k}^N \left(\frac{\partial F}{\partial \rho_k} + \frac{\partial F}{\partial \rho_j} \right) \frac{\partial \rho(r_{kj})}{\partial r_{kj}} \frac{\vec{r}_{kj}}{r_{kj}} \end{aligned} \quad (5.86)$$

Computing Stress

The stress is to be calculated assuming the system is at 0K[105] so the virial stress equation is eq. 5.87.

$$\tau_{ij} = \frac{1}{2V} \sum_{k,l \in V} \left(x_i^{(l)} - x_i^{(k)} \right) f_j^{(kl)} \quad (5.87)$$

Within the computer code, the individual force components between pairs of atoms are stored as well as an equal sized array of vectors representing their spatial separation. Those pairs with one atom in the halo around the simulation box are used to calculate the stress on the simulation box.

Pseudo Code for the Energy, Force and Stress Subroutine

The calculation requires several arrays to temporarily store data. The electron densities and gradients of the density function are stored for multiple groups of densities and embedding functionals. The overall forces on each atoms also need to be stored as well as the force between pairs of atoms, as this is required to compute the virial stress. The Fortran code to compute this is included in the appendix (section ??) and a pseudocode in listing 5.2 explains the process.

Listing 5.2: Pseudo Code for Energy and Stress Force Calculation

```

1
2 electron_density[1:n_atoms, 1:bands] = 0.0 // electron density at each atom
3 density_grad_ab[:, :] = 0.0                // grad density function
4 density_grad_ba[:, :] = 0.0                // grad density function
5 energy = 0.0                                // total energy
6 forces[:, :] = 0.0                          // forces each atom, 3D
7 force_between_pairs[:, :] = 0.0             // used to store force between pairs, for stress computation
8 stress[:, :] = 0.0DO
9
10 // LOOP 1 - Pair energy, force and calculate densities
11 DO n = 1, neighbour_count
12   E = get_PairEnergy(atom_a, atom_b, r[n,:])
13   F[:] = get_PairForce(atom_a, atom_b, r[n,:])
14
15 // Save energy
16 energy = energy + E
17
18 // Save Force
19 f[atom_a, :] = f[atom_a, :] - F[:]
20 f[atom_b, :] = f[atom_b, :] + F[:]
21 force_between_pairs[:, :] = force_between_pairs[:, :] - F[:] // Used to calculate stress
22
23 // Loop through density bands
24 DO band = 1, bands

```

```

25 // Electron density at A due to atom B
26 electron_density[atom_a, b] = get_Density(atom_b, r)
27 density_grad_ab[n, band] = get_DensityGradient(atom_b, r)
28
29 // Electron density at B due to atom A
30 electron_density[atom_a, band] = get_Density(atom_a, r)
31 density_grad_ba[n, band] = get_DensityGradient(atom_a, r)
32 END DO
33 END DO
34
35 // LOOP 2 - Embedding energy
36 DO n = 1, atom_count
37 // Loop through density bands
38 DO band = 1, bands
39   energy = energy + get_EMBEDDINGEnergy(n, band)
40 END DO
41 END DO
42
43 // LOOP 3 - Embedding force
44 DO n = 1, neighbour_count
45 // Loop through density bands
46 DO band = 1, bands
47   epA = get_EMBEDDINGGradient(atom_a, electron_density(atom_a, b))
48   epB = get_EMBEDDINGGradient(atom_b, electron_density(atom_b, b))
49
50 F[:] = (epA * density_grad_ba(n, band) + epB * density_grad_ab(n, band)) * r[n, :]
51
52 f[atom_a, :] = f(atom_a, :) - F[:]
53 f(atom_a, :) = f(atom_a, :) + F[:]
54
55 force_between_pairs[n,:] = force_between_pairs[n,:] - F[:] // Used to calculate stress
56 END DO
57 END DO
58
59 // LOOP 4 STRESS
60 DO n = 1, neighbour_count
61 // Only compute if the second atom is in the halo
62 IF(nlisthalo(n))THEN
63   DO i = 1,3
64     DO j = 1,3
65       stress[i,j] = stress[i,j] + (r[n, i] * force_between_pairs[n,j])
66     END DO
67   END DO
68 END IF
69 END DO
70 stress[1:3, 1:3] = stress[1:3, 1:3] / (2.0 * volume)

```

5.3.7 Choice of Functions and Functionals

The EAM potential, for n elements, is made up of n pair functions, $n(n+1)/2$ density functions and $n(n+1)/2$ embedding functionals. The choice of function varies greatly. Irrespective of the choice of function, there are a number of desireable properties:

- the function should be well behaved
- this should be true for the derivative and second derivative - ideally, these should be continuous too
- the pair and density functions should have a cutoff, and the value at this cutoff should be zero

ZBL to Pair Function Spline

Certain functions may behave oddly at small values of atom separation, and this can be seen for the Buckingham potential, where the potential may drop sharply to negative values, pulling the modelled atoms together. The ZBL function is a good choice to model higher energy atoms that do come close together. It may be desirable to keep the characteristics for the

A One Time Neighbour List

As there are only static calculations in this work, the neighbour list will only need to be computed once. The potential functions will change during the fitting process, but the configuration, and neighbourlist, is frozen in place from the start to the end of the fitting procedure.

5.3.8 Sacrificing Physical Elegance

We have no knowledge of a formula that can be applied to any material and describe the energy and forces between atoms exactly. Even first principles calculations rely of approximations, and the knowledge of an exact exchange functional eludes us. While it would be satisfying to derive a potential that reflects the fundamental physics, it may be something that is forever out of our reach.

It is more useful to develop a potential that replicates the behavior of DFT calculations (matches the forces, energies, stresses etc) and reproduces material properties such as its elastic constants. If it can do this, and give insights into the material in a way that is not yet possible experimentally, does it matter if the physical elegance is lost?

5.4 Force Matching

To derive a potential, one may approach the problem from first principles in an attempt to replicate reality. It has been more useful, however, to lose any physical elegance [97] to give potentials that work for specific elements under certain conditions. Force data, gathered experimentally or by first-principles calculations, has been used to develop potentials since the 1990s. The force matching method was developed in 1994 by Ercolessi and Adams [106] to link the more accurate, more processor and memory intensive, world of first-principles calculations to Molecular Dynamics.

The force-matching method uses the difference between the actual force (either measured experimentally or calculated by first-principles calculations)

Given a set of M different atomic configurations, and a potential with a set of L parameters (\vec{p}), the function Z_F is a measure of the difference between the the forces calculated using the potential for all configurations and the actual (or DFT generated) forces.

$$Z_F(\vec{\alpha}) = \sum_{k=1}^M \sum_{i=1}^k \sum_{j=1}^3 |F_{i,j}^k(\vec{\alpha}) - F_{i,j}^0|^2 \quad (5.88)$$

This may be extended to include the calculation of other properties:

- Lattice Parameter a_0
- Cohesive Energy E_{coh}

- Bulk Modulus B_0
- Equation of State (E_0, V_0, B_0, B_0^p)
- Stress Tensor
- Elastic Constants
- Shear Modulus, Young's Modulus, Poisson Ratio
- Melting Temperature
- Surface Energy

Each may be weighted depending on how important the property is to the simulation the potential is required for.

$$Z(\vec{p}) = w_F Z_F(\vec{p}) + w_{b0} Z_{b0}(\vec{p}) + w_{e0} Z_{e0}(\vec{p}) + w_{a0} Z_{a0}(\vec{p}) + w_{ec} Z_{ec}(\vec{p}) + w_{ecoh} Z_{ecoh}(\vec{p}) \quad (5.89)$$

The result is a residual squared sum that measures how well or poorly a potential fits the required data.

5.5 Magnetism

5.5.1 Brief History of Magnetism

Magnetism as a phenomenon has been known to our civilisation for thousands of years. The magnetic mineral Lodestone contains the iron oxide Magnetite Fe_3O_4 and it was used by the ancient Greeks over two thousand years ago.

The magnetic compass had been used since the 12th century, but it wasn't until the 17th Century that the Earth was discovered as a magnet itself by William Gilbert.

In the early 1800s, Hans Christian Oersted discovered that an electric current could move a compass needle, and this lead to much research into the connection between electricity and magnetism. Experimental work by Michael Faraday lead to the discovery that a changing magnetic field produces an electric field. By the mid-1800s the mathematical physicist, James Clerk Maxwell, derived his set of equations describing the connection between magnetism and electricity.

$$\partial \cdot \vec{E} = \frac{\rho}{\epsilon_0}$$

$$\nabla \cdot \vec{B} = 0$$

$$\nabla \times \vec{E} = -\frac{1}{c} \frac{\partial \vec{B}}{\partial t}$$

$$\nabla \times \vec{B} = \frac{1}{c} \left(4\pi \vec{J} + \frac{\partial \vec{E}}{\partial t} \right)$$

5.5.2 Magnetism: Moving Charges and Spin

General Relativity and Quantum Mechanics were invented as theories to solve anomalies in science during the first part of the 20th Century, and they describe the very large and very small realms we do not notice in every day life. In describing how magnetism comes to be, it may be useful to use classical analogies, but that's all they are. Quantum Mechanics is required and this replaces intuition with a mathematical description.

$$\vec{B} = \frac{\mu_0 q}{4\pi r_{ij}^3} \vec{v} \times \vec{r}_{ij} \quad (5.90)$$

When a charged particle moves, it creates a magnetic field. The field at some point j created by a charge moving at \vec{v} at point i is calculated (eq. 5.90). Electrons do not orbit around a nucleus in the classical sense, but they do have an orbital angular momentum. The overall magnetic field created by an atom is made up of:

- orbital angular momentum of the electron
- spin of electrons
- spin of nucleons

The contribution of the nucleus to the magnetic field is small, and the orbital component and spin component of electrons are mostly responsible.

Spin is a property of quantum mechanics and it doesn't have an equivalent in classical mechanics, although it may be thought of similar to a spinning top or a rotating planet. Electrons are point like particles, and do not actually rotate, but they do have an intrinsic angular moment (spin). Where a spinning top might have a range of angular velocities, electrons have a quantised value and, as fermions, they have half integer spin.

5.5.3 Ferromagnetism, Antiferromagnetism and Hund's Rule

Electrons fill the shells of a ground state atom such that the energy is minimised. Four quantum numbers are used to describe electrons bound to an atom: n , l , m_l and m_s . The Pauli exclusion principle states that the electrons, which are fermions, cannot have the same quantum numbers as another electron bound to that atom.

Electrons may be spin up or spin down thus two electrons can have the same n , l , m_l with two values for m_s available. The energy is minimised by not pairing up and down electrons until all the free slots due to the coulomb interaction between the electrons. Once each slot contains one electron, they begin to pair.

The electronic configuration of Iron highlights this:

$$\begin{array}{ll}
 1s^2 & \underline{\uparrow\downarrow} \\
 2s^2 & \underline{\uparrow\downarrow} \\
 2p^6 & \underline{\uparrow\downarrow} \ \underline{\uparrow\downarrow} \ \underline{\uparrow\downarrow} \\
 3s^2 & \underline{\uparrow\downarrow} \\
 3p^6 & \underline{\uparrow\downarrow} \ \underline{\uparrow\downarrow} \ \underline{\uparrow\downarrow} \\
 4s^2 & \underline{\uparrow\downarrow} \\
 3d^6 & \underline{\uparrow\downarrow} \ \underline{\uparrow} \ \underline{\uparrow} \ \underline{\uparrow} \ \underline{\uparrow}
 \end{array} \quad (5.91)$$

Rather than fill the 3d shell from left to right, the first five slots take electrons with spin in the same direction, and the remaining electron fills the first slot with spin opposite to the first five electrons.

Being fermions the wavefunction for the two electrons must be antisymmetric, with spins opposite to one another. Electrons with the same spin will repel one another and this causes an increase in the screening between the electrons and the nucleus. The screening lowers the attraction between the electrons and the nucleus that then results in the total energy of the atom decreasing[107].

Chromium

$$\begin{array}{c} 4s^1 \quad \uparrow \\ 3d^5 \quad \underline{\uparrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \end{array} \quad (5.92)$$

Iron

$$\begin{array}{c} 4s^2 \quad \underline{\uparrow\downarrow} \\ 3d^6 \quad \underline{\uparrow\downarrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \end{array} \quad (5.93)$$

Cobalt

$$\begin{array}{c} 4s^2 \quad \underline{\uparrow\downarrow} \\ 3d^7 \quad \underline{\uparrow\downarrow} \quad \underline{\uparrow\downarrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \end{array} \quad (5.94)$$

Nickel

$$\begin{array}{c} 4s^2 \quad \underline{\uparrow\downarrow} \\ 3d^8 \quad \underline{\uparrow\downarrow} \quad \underline{\uparrow\downarrow} \quad \underline{\uparrow\downarrow} \quad \underline{\uparrow} \quad \underline{\uparrow} \end{array} \quad (5.95)$$

In pure Iron (eq. 5.93), under normal conditions, it is energetically favourable for the atoms to align in the same direction, and as such iron is ferromagnetic. Cobalt (eq. 5.94) has a face centered tetragonal structure at room temperature. It has a similar electronic structure to iron, but has three unpaired electrons in the 3d shell rather than four. It is a ferromagnetic whilst it remains in its CPH form. Nickel (eq. 5.95) is also ferromagnetic, but it is ferromagnetic in its FCC phase.

In its gamma phase, as it is in austenitic stainless steel, iron atoms align opposite to one another in an antiferromagnetic arrangement. This is also the same for BCC chromium (eq. 5.92) which is antiferromagnetic in its pure form under normal conditions.

5.6 Density Functional Theory

Acrlfulldft is a branch of quantum chemistry that approximately solves the Schrödinger equation using electron density, rather than the coordinates of each electron in the system. A number of simplifications are also applied in order for DFT to be practical to use, but despite this calculations are limited to just hundreds or thousands of atoms. A calculation of a hundred or so atoms may take thousands of CPU hours at the time of writing, depending on the type of calculation and complexity of the electron structures of the atoms involved.

It is through acrshortdft that the first principles energy, stress and force calculations will be made, and it's these results that the EAM potentials will be trained and fit to using the force matching method. This will allow much larger scale modelling using the extrapolated behaviour of accurate DFT calculations.

5.6.1 Brief Overview of DFT

Several important theories and approximations are used by DFT with the aim of calculating and minimising energies and forces. The Born-Oppenheimer approximation separates the electron-nucleus wave function. It treats the nuclei as fixed points, and the system of electrons in a fixed potential created by the nuclei.

The DFT of Kohn, Sham and Hohenberg proved that the potential of a system is uniquely determined by its ground state density. This makes solving the Schrödinger equation significantly easier.

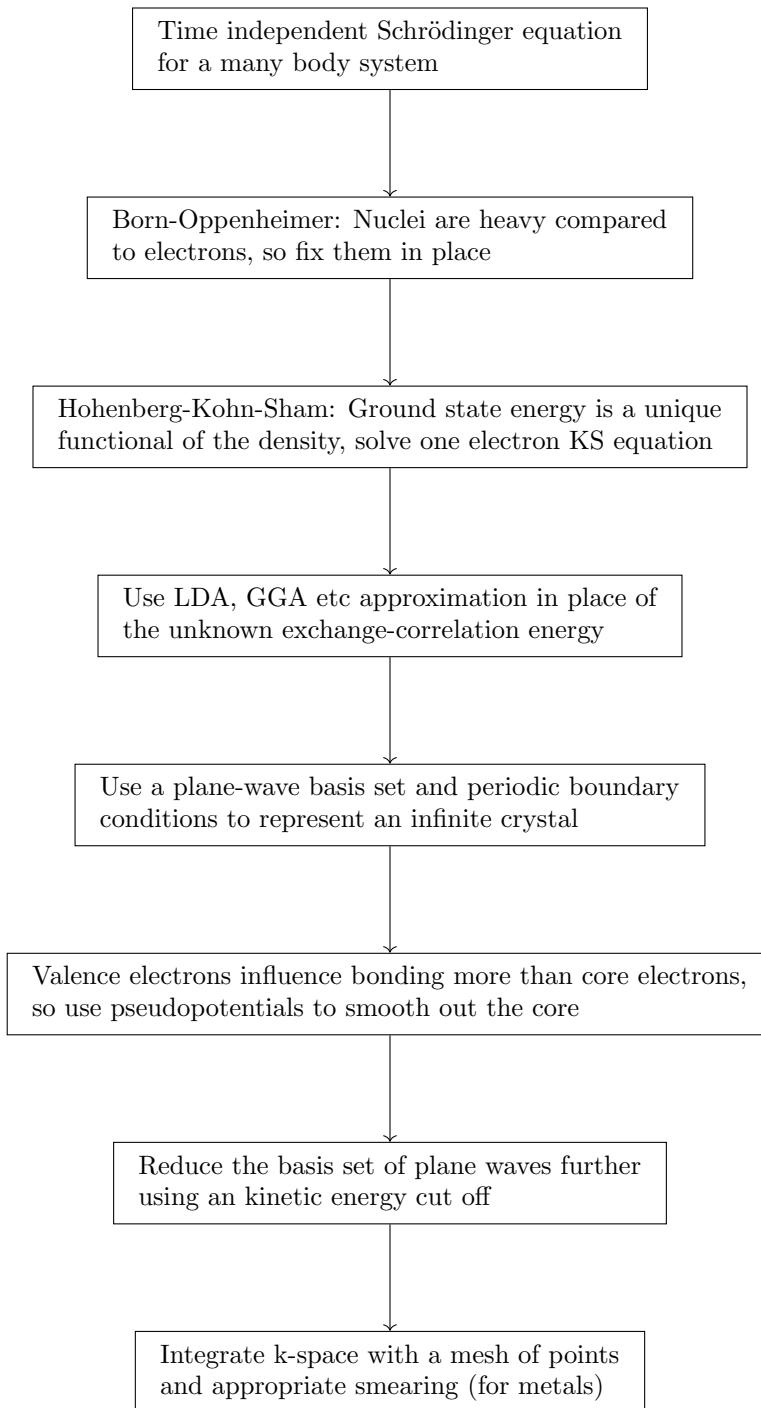


Figure 5.10: Common approximations used to enable DFT calculations

5.6.2 Time Independent Schrödinger Equation

The Schrödinger equation is a linear partial differential wave equation and it was proposed by Erwin Schrödinger in 1925. There is a time-dependent and time-independent form of the equation. As the DFT calculations in this work are static only the time-independent version will be discussed.

$$\hat{H}|\Psi\rangle = E|\Psi\rangle \quad (5.96)$$

The Hamiltonian \hat{H} is an operator and it is the total energy in the system. Ψ is the wave function (for example, the wave function of an electron) and this contains all the measurable information possible about whatever it represents. E is the energy eigenvalue of this system and this will depend on the eigenstate of the system. The wave function is also connected to the probability of a particle being found at a certain point in space, and the integral over all space is equal to 1 i.e. the probability of it being found somewhere in space is equal to 1 (eq. 5.97).

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Psi(x, y, z) dx dy dz = 1 \quad (5.97)$$

The Schrödinger equation is set up depending on the system being studied. Starting with the simplest, a free particle, the only non zero energy of the Hamiltonian is kinetic.

$$\begin{aligned} \hat{H} &= \frac{\hbar^2}{2m} \nabla^2 \\ -\frac{\hbar^2}{2m} \nabla^2 \Psi(\vec{r}) &= E\Psi(\vec{r}) \end{aligned} \quad (5.98)$$

If the particle is in a potential, it will have both kinetic energy \hat{T} and potential energy \hat{V} (eq. 5.99).

$$\begin{aligned} \hat{H} &= \hat{T} + \hat{V} = \frac{\hbar^2}{2m} \nabla^2 + V(\vec{r}) \\ \left[-\frac{\hbar^2}{2m} \nabla^2 \right] \Psi(\vec{r}) &= E\Psi(\vec{r}) \end{aligned} \quad (5.99)$$

5.6.3 Many Body TISE

Electronic structure calculations are important and allow the calculation of material properties that may be difficult or impossible to measure with current technology. The next step towards this is to set up the Schrödinger equation (time-independent) for nuclei and electrons in a crystal.

Relative to the strong force, the electromagnetic force is 1/137th as strong, but the strong force acts over a range of approximately 1.0×10^{-5} angstrom. In the calculations carried out in this section, the atoms will never be arranged close enough for the strong force to be considered at all. The gravitational force, as with the electromagnetic force, acts over an infinite range. The electromagnetic force is more than 1.0×10^{36} times greater than the gravitational force, so it too can be neglected. Finally, the weak interacting force has a range of approximately 1.0×10^{-8} angstrom which, as with the strong force, is a range small enough that the weak force may be neglected.

The energy operators required are kinetic and electromagnetic potential (eq. eq:TimeIndependentSchrodinger3).

$$\hat{H} = \hat{T}_e + \hat{T}_n + \hat{V}_{e-e} + \hat{V}_{e-n} + \hat{V}_{n-n} \quad (5.100)$$

The first two terms are the kinetic energy of the electrons and nuclei respectively (eq. eq:TimeIndependentSchrodinger4).

$$\begin{aligned}\hat{T}_e &= -\sum_i \frac{\hbar^2}{2m} \nabla^2 \text{Sum of kinetic energy of electrons} \\ \hat{T}_n &= -\sum_k \frac{\hbar^2}{2M} \nabla^2 \text{Sum of kinetic energy of nuclei}\end{aligned}\quad (5.101)$$

The last three terms are potential energy terms due to the electromagnetic force (eq. eq: eqTimeIndependentSchrodinger5).

$$\begin{aligned}V_{e-e} &= \frac{1}{2} \sum_{i,j,i \neq j} \frac{1}{|\vec{r}_i - \vec{r}_j|} \text{sum of potential energy between electrons} \\ V_{e-n} &= \sum_{i,k} \frac{z_i}{|\vec{r}_i - \vec{r}_l|} \text{sum of potential energy between electrons and nuclei} \\ V_{n-n} &= \frac{1}{2} \sum_{k,l,k \neq l} \frac{z_k z_l}{|\vec{r}_l - \vec{r}_k|} \text{sum of potential energy between nuclei}\end{aligned}\quad (5.102)$$

These operators are now input into the TISE (eq. 5.103).

$$\left[\left(-\sum_i \frac{\hbar^2}{2m} - \sum_k \frac{\hbar^2}{2M} \right) \nabla^2 + \sum_{i,k} \frac{z_i}{|\vec{r}_i - \vec{r}_l|} + \frac{1}{2} \left(\sum_{i,j,i \neq j} \frac{1}{|\vec{r}_i - \vec{r}_j|} + \sum_{k,l,k \neq l} \frac{z_k z_l}{|\vec{r}_l - \vec{r}_k|} \right) \right] |\Psi\rangle = E|\Psi\rangle \quad (5.103)$$

5.6.4 Born-Oppenheimer

The TISE arrived at is far too complicated to solve, even for the simple ensembles of atoms. It represents the many body system of electrons and nuclei, and it takes the positions of all nuclei and electrons as input variables (eq. 5.104).

$$\begin{aligned}\hat{H}|\Psi(\vec{r}_e, \vec{r}_n)\rangle &= E|\Psi(\vec{r}_e, \vec{r}_n)\rangle \\ \text{where } \vec{r}_e &= r_{e,1}, r_{e,2}, \dots, r_{e,i} \text{ (electron positions)} \\ \text{where } \vec{r}_n &= r_{n,1}, r_{n,2}, \dots, r_{n,i} \text{ (electron positions)}\end{aligned}\quad (5.104)$$

In 1927 the Born-Oppenheimer approximation was proposed to separate the electron components from the nuclei in the Hamiltonian.

Protons and Neutrons are almost 2,000 times the mass of electrons. With respect to the electrons, they move much slower and may be considered to be fixed or frozen in place. Simplifying for a moment to a single electron and proton, due to Newton's second law, we can see that the acceleration of the electron would be similarly 2,000 times that of the proton: $f_e = f_p$ and $m_e a_e = m_p a_p$ which leads to $a_e = \frac{m_p}{m_e} a_p$. As the nuclei move, the electrons are assumed to respond instantly, remaining in the ground state and not being promoted into higher energy levels.

The Hamiltonian for the electrons may be written with the electron co-ordinates as a variable, and the nuclei co-ordinates as a parameter (eq. 5.105).

$$\hat{H}_e(\vec{r}_e; \vec{r}_n) = \hat{T}_e(\vec{r}_e) + \hat{V}_{e-e}(\vec{r}_e) + \hat{V}_e - n(\vec{r}_e; \vec{r}_n) \quad (5.105)$$

The wavefunction and energy for the electrons may be calculated, although if the nuclear coordinates are changed, i.e. by changing the parameter r_n , the wavefunction and energy will need to be recalculated.

$$\hat{H}_e(\vec{r}_e; \vec{r}_n) \psi_e(\vec{r}_e; \vec{r}_n) = E_e(\vec{r}_n) \psi_e(\vec{r}_e; \vec{r}_n) \quad (5.106)$$

$$\Psi(\vec{r}_e, \vec{r}_n) = \chi_{ne}(\vec{r}_e) \psi_e(\vec{r}_e; \vec{r}_n) \quad (5.107)$$

The electronic Hamiltonian (eq. 5.106) is still dependent on the electronic co-ordinates. As there are three co-ordinates per electron, there are 3 dimensions when solving for a hydrogen atom. For an Iron atom, with 26 electrons, there are 78 dimensions, and for a 4x4x4 FCC supercell of Iron there would be 256 atoms, 26 electrons per atom and 3 dimensions per electron, giving a total of 19,968 dimensions. Even for small numbers of electrons, solving this equation is impractical.

5.6.5 Crystals, Reciprocal Space and Bloch Theorem

A Bravais lattice is a construct used to describe a periodic crystal lattice. It has the following properties given in eq. 5.108.

$$\begin{aligned} \vec{R} &= n_1 \vec{a}_1 + n_2 \vec{a}_2 + n_3 \vec{a}_3 \\ n_1, n_2, n_3 &\in \mathbb{Z} \\ \vec{a}_1, \vec{a}_2, \vec{a}_3 &\text{ are independent} \end{aligned} \quad (5.108)$$

There are 14 Bravais lattices and 7 families of these lattices, and these lattices may be grouped by vector length and angle relationships (table. 5.4). This work is primarily concerned with cubic and tetragonal crystals, although distortions are applied to these crystals throughout.

Class	Lengths	Angles
Cubic	$a = b = c$	$\alpha = \beta = \gamma = 90$
Hexagonal	$a = b, c$	$\alpha = \beta, \gamma = 120$
Rhombohedral	$a = b = c$	$\alpha = \beta = \gamma \neq 90$
Tetragonal	$a = b, c$	$\alpha = \beta = \gamma = 90$
Orthorhombic	a, b, c	$\alpha = \beta = \gamma = 90$
Monoclinic	a, b, c	$\alpha = \beta = 90, \gamma \neq 90$
Triclinic	a, b, c	$\alpha, \beta, \gamma,$

Table 5.4: Bravais lattice vector length and angle relationships

Bloch Theorem

Metals are made up from grains which in turn are crystal lattices of atoms. Despite the majority of metals being composed of a large collection of microscopic crystals, rather than being a single perfect crystal, most of their properties may be calculated as if the metal were a single crystal.

The grain size of water quenched SS304 is approximately 30 micrometers across[108]. If the crystal were a cube, it would contain 2.0×10^{15} atoms and it would have sides over 100,000 atoms in length.

In solid state physics, in order to solve the TISE for such a crystal, it is useful to consider an infinite sized crystal. It may be helpful to visualise this as a "ring" of atoms in one dimension (fig. 5.11) or as a super-cell with periodic boundary conditions in three dimensions.

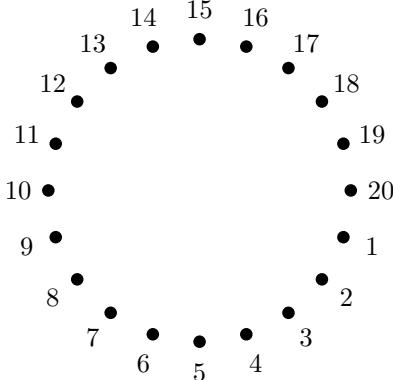


Figure 5.11: A useful, albeit incorrect, way of visualising an "infinite" chain of atoms in 1D

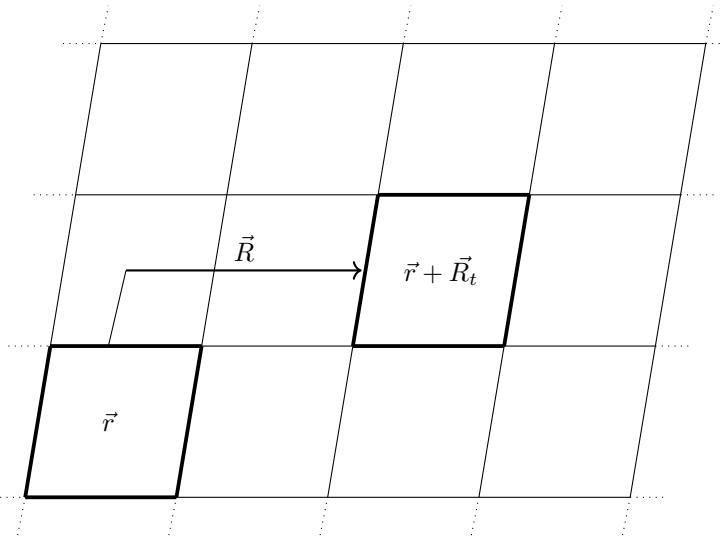


Figure 5.12: A 2D example of a translation by an integer multiple of the unit cell from the origin unit cell

As the structure is a repeating lattice, any point within a unit cell is equivalent to any other point translated by an integer multiple of the lattice vector (fig. 5.12).

$$\begin{aligned} \psi(\vec{r}) &= \psi(\vec{r} + \vec{R}_t) \\ \text{where } \vec{R}_t &= n_1 \vec{R}_1 + n_2 \vec{R}_2 + n_3 \vec{R}_3 \\ \text{where } \vec{R} &\text{ is the real lattice vector} \end{aligned} \tag{5.109}$$

Reciprocal space, also known as k-space or momentum space, is a mathematical construct. It is an imaginary space where the lengths, and volumes, are the inverse of real space and planes of atoms are represented as points.

Starting with a lattice of points in real space \vec{R} , points in reciprocal space \vec{G} only are valid points if $\exp(i\vec{G} \cdot \vec{R}) = 1$ [109]. A real space vector \vec{R} is transformed to it's reciprocal in the following way, where Ω is the volume of the primitive cell in real space.

$$\begin{aligned} \vec{g}_1 &= \frac{2\pi}{\Omega} \vec{r}_2 \times \vec{r}_3 \\ \vec{g}_2 &= \frac{2\pi}{\Omega} \vec{r}_3 \times \vec{r}_1 \\ \vec{g}_3 &= \frac{2\pi}{\Omega} \vec{r}_1 \times \vec{r}_2 \\ \Omega &= \vec{r}_1 \cdot (\vec{r}_2 \times \vec{r}_3) \end{aligned} \tag{5.110}$$

If the crystal structure is modelled as an infinitely repeating lattice, the potential also has the same periodicity (eq. 5.110).

$$V(\vec{r}) = V(\vec{r} + \vec{R}_t) \quad (5.111)$$

As the potential is a periodic function, it may be written as a fourier series in reciprocal space (eq. 5.112).

$$V(\vec{r}) = \sum_{\vec{G}} V_{\vec{G}} \exp(i\vec{G} \cdot \vec{r}) \quad (5.112)$$

The wave function for an electron in the lattice is also periodic and may be written as the product of a plane wave.

$$\psi_{n\vec{k}}(\vec{r}) = u_{n\vec{k}}(\vec{r}) \exp(i\vec{k} \cdot \vec{r}) \quad (5.113)$$

$$\begin{aligned} u_{n\vec{k}}(\vec{r}) &= u_{n\vec{k}}(\vec{r} + \vec{R}_t) \\ \psi_{n\vec{k}}(\vec{r}) &= \psi_{n\vec{k}}(\vec{r}) \exp(i\vec{k} \cdot \vec{R}_t) \end{aligned} \quad (5.114)$$

The Born-von Karman boundary conditions apply to the wave function such that:

$$\psi_{n\vec{k}}(\vec{r} + \vec{R}_t) = \psi_{n\vec{k}}(\vec{r}) \text{ where } \vec{R}_t = \sum_i N_i \vec{a}_i \text{ and } N_i \in \mathbb{Z} \quad (5.115)$$

With these boundary conditions and use of plane waves, Bloch theorem replaces an enormous number of electrons with an infinite number electrons in a periodic lattice where only those in the unit cell need to be considered.

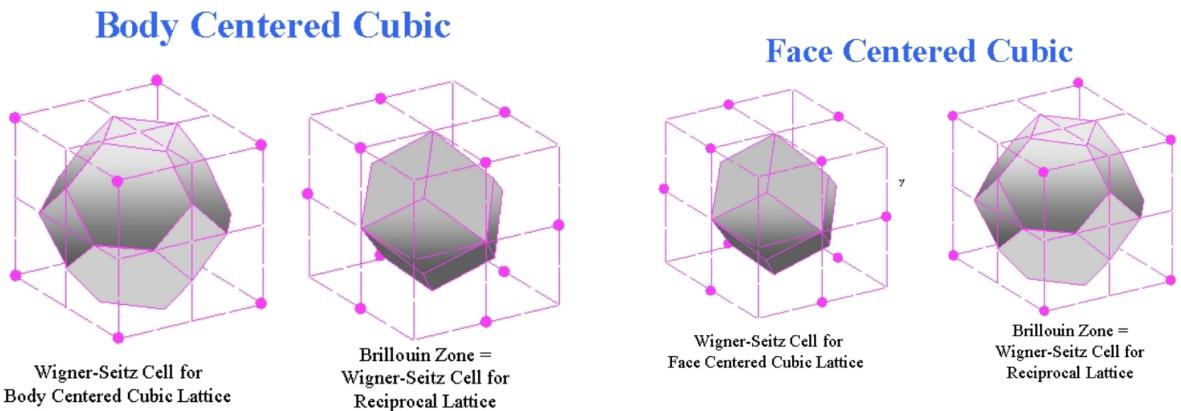


Figure 5.13: BCC Wigner Seitz Cell and BZ [110]

Figure 5.14: FCC Wigner Seitz Cell and BZ [110]

The (first) Brillouin Zone is a volume in (3D) reciprocal space and it is analogous to the Wigner-Seitz cell in real space. It is the primitive lattice cell in reciprocal space. The reciprocal of a BCC lattice is a FCC lattice (fig. 5.13) and vice versa (fig. 5.14). In real space a point anywhere in the lattice is a integer multiple of the lattice vector away from the same point in the origin Wigner-Seitz cell. There are an infinite number of wave vectors \vec{k} in reciprocal space, but by virtue of the periodicity of the lattice, they are also all found within the Brillouin Zone. If the crystal has symmetries, the BZ may be reduced further to the irreducible Brillouin zone.

Within the BZ there are energy surfaces. Picking a straight line through reciprocal space allows one to plot the energy bands along that one dimensional space. A particularly important energy surface is the Fermi surface and, in the one electron model, this marks the boundary between the occupied and unoccupied states at 0K [111]. The Fermi energy is the energy of the highest occupied state at 0K.

5.6.6 Hohenberg-Kohn Theorem

In the 1960s, Hohenberg and Kohn[112] simplified the problem of solving the many electron TISE in an exact way by proving:

- in an external potential $v(\vec{r})$, the potential is uniquely determined by the density of the ground state $n_0(\vec{r})$ assuming that the particles are non-degenerate
- a uniquely defined functional $E[\rho(\vec{r})]$ exists and the ground state energy, $\min(E[\rho(\vec{r})])$, is found by varying the density

The proof starts by picturing a box of electrons that interact with each other through coulomb repulsion within an external potential $v(r)$, for example the potential of “fixed” nuclei following the Born Oppenheimer approximation.

$$\begin{aligned} \hat{H} &= \hat{T} + \hat{V} = \frac{1}{2}\nabla^2 + V(\vec{r}) \\ &\left[-\frac{1}{2}\nabla^2 \right] \Psi(\vec{r})_0 = E_0 \Psi(\vec{r}) \end{aligned} \quad (5.116)$$

$$\hat{H} = \hat{T} + \hat{V} + \hat{U} \quad (5.117)$$

where

$$T = \frac{1}{2} \int \nabla \psi^*(\vec{r}) \nabla \psi(\vec{r}) d\vec{r} \quad (5.118)$$

$$V = \int v(\vec{r}) \psi^*(\vec{r}) \psi(\vec{r}) d\vec{r} \quad (5.119)$$

$$U = \frac{1}{|\vec{r} - \vec{r}'|} \psi^*(\vec{r}) \psi^*(\vec{r}') \psi(\vec{r}) \psi(\vec{r}') d\vec{r} d\vec{r}' \quad (5.120)$$

In the eqs. 5.117 5.118 5.117 5.120 the hamiltonian operator \hat{H} is a sum of the kinetic energy \hat{T} , the mutual coulomb repulsion \hat{U} and the operator due to an external potential (due to the nuclei) \hat{V} . The non degenerate ground state electron density is $n(\vec{r}) = \langle \Phi | \phi^*(\vec{r}) \phi(\vec{r}) | \Phi \rangle$.

The HK theorem shows that the external potential $v(\vec{r})$ is a unique functional of the electron density $n(\vec{r})$. The proof is that by contradiction where by two different states are assumed to have the same ground state charge density.

State A

$$\begin{aligned}\Psi_A \text{ with potential } V_A(\vec{r}) \\ \text{Hamiltonian } \hat{H} = \hat{T} + \hat{V}_A + \hat{U} \\ \text{TISE } \hat{H}_A \Psi_A = E_A \Psi_A\end{aligned}$$

Assumption - this state has the charge density $n(\vec{r})$

The minimal property of the ground state gives the following[112][113]:

$$E_A = \langle \Phi_A | \hat{H}_A | \Phi_A \rangle$$

$$\begin{aligned}E_A &= \langle \Phi_A | \hat{H}_B + \hat{V}_A - \hat{V}_B | \Phi_A \rangle \\ E_A &= \langle \Phi_A | \hat{H}_B | \Phi_A \rangle + \int d^3 \vec{r} n(\vec{r}) (v_A(\vec{r}) - v_B(\vec{r}))\end{aligned}\tag{5.121}$$

$$E_A > E_B + \int d^3 \vec{r} n(\vec{r}) (v_A(\vec{r}) - v_B(\vec{r}))\tag{5.122}$$

The indices A and B are interchanged to give a second equation:

$$\begin{aligned}E_B &> E_A + \int d^3 \vec{r} n(\vec{r}) (v_B(\vec{r}) - v_A(\vec{r})) \\ E_B &> E_A - \int d^3 \vec{r} n(\vec{r}) (v_A(\vec{r}) + v_B(\vec{r}))\end{aligned}\tag{5.123}$$

By adding eq. 5.122 and eq. 5.123 together the integrals will cancel out.

$$E_A + E_B > E_A + E_B\tag{5.124}$$

This (eq. 5.124) is a contradiction and proves the first part of the HK theorem: $v(\vec{r})$ is uniquely determined by $n(\vec{r})$. If the charge density (fig. 5.15 and 5.15) is known, a single external potential exists for it.

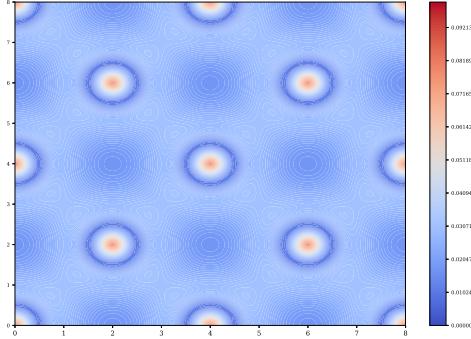


Figure 5.15: Charge density FCC aluminium xy plane at $z = 0.0a_0$

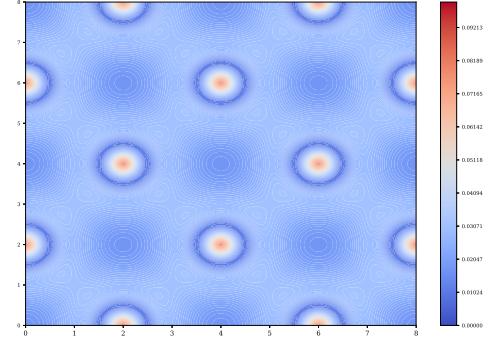


Figure 5.16: Charge density FCC aluminium xy plane at $z = 0.5a_0$

The next part of the theorem is to show that the energy functional $E[n(\vec{r})]$ exists and the minimum can be found by varying the charge density function. The kinetic and interaction functional \hat{T} and \hat{U} define a functional $F[n(\vec{r})]$.

$$F[n(\vec{r})] \equiv \langle \Phi | \hat{T} + \hat{U} | \Phi \rangle \quad (5.125)$$

This is a universal functional[112] and it makes up a part of the energy functional. The energy functional is comprised of the HK functional $F[n(\vec{r})]$ and the external potential as shown in eq. 5.126.

$$E[n(\vec{r})] \equiv \int v(\vec{r})n(\vec{r})d\vec{r} + F[n] \quad (5.126)$$

For the ground state electron density $n(\vec{r})$ the energy will be equal to the ground state energy $E_0 = E_v[n]$. Where N is the number of particles in the system, the integral of the density over all space will sum to N (eq. 5.127).

$$N[n] = \int n(\vec{r})d\vec{r} \quad (5.127)$$

If the system has N particles the energy level for a none ground state energy, k, is given in eq. 5.128.

$$\epsilon_{v,k}[\Psi_k] \equiv \langle \Phi_k | \hat{V} | \Phi_k \rangle - \langle \Phi_k | \hat{T} - \hat{U} | \Phi_k \rangle \quad (5.128)$$

This has a minimum where $\Psi_k = \Psi_0$, the ground state.

$$\begin{aligned} \epsilon_{v,k}[\Psi_k] &= \int v(\vec{r})n_k(\vec{r})d\vec{r} + F[n_k] \\ \epsilon_{v,0}[\Psi_k] &= \int v(\vec{r})n_0(\vec{r})d\vec{r} + F[n_0] \\ \epsilon_{v,k} &> \epsilon_{v,0} \end{aligned} \quad (5.129)$$

5.6.7 Kohn-Sham Equations

The year following the Hohenberg-Kohn theorem, a set of self-consistent equations were derived by Kohn and Sham. The ground state energy of interacting jellium in the potential of fixed nuclei, an external potential, can be written as follows[114]:

$$E = T_s + U + V_{n-e} + E_{xc} \quad (5.130)$$

$$E = T_s[\rho(\vec{r})] + \int d\vec{r}v(\vec{r})\rho(\vec{r}) + \frac{1}{2} \int \int d\vec{r}d\vec{r}' \frac{\rho(\vec{r})\rho(\vec{r}')}{|\vec{r} - \vec{r}'|} + E_{xc}[\rho(\vec{r})] \quad (5.131)$$

The kinetic energy, T_s , is that of a system of non interacting and E_{xc} is the exchange and correlation of an interacting system. The exchange and correlation energy functional exists, but is unknown.

The Kohn-Sham equations are used to calculate the energy of a system. The Schrödinger equation is not for all the atoms in the system; it's a one electron equation

$$\hat{H}_{KS}\psi_i = E_i\psi_i \quad (5.132)$$

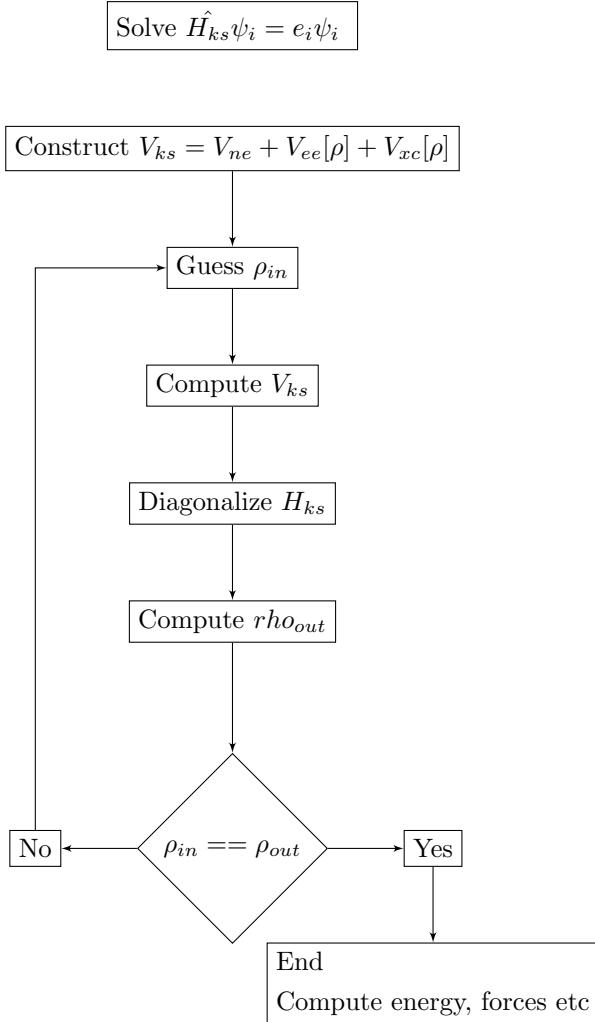
$$(-\frac{1}{2}\nabla^2 + \hat{v}_{KS})\psi_i = E_i\psi_i \quad (5.133)$$

$$\hat{v}_{KS}(\vec{r}_e, \vec{r}_n) = v_{n-e}(\vec{r}_e, \vec{r}_n) + \int d^3\vec{r}' \frac{\rho(\vec{r}'_e)}{|\vec{r}_e - \vec{r}'_e|} + v_{xc}[\rho](\vec{r}_e) \quad (5.134)$$

It is important to note that even though this is a one electron equation it is an exact solution.

5.6.8 Self-Consistent Solution

The Kohn-Sham equations cannot be solved in the usual way. They require a value for the density, but the density is obtained by solving the equations, so there is a dilemma. It can however be solved self consistently: an initial density is guessed, and this is repeatedly updated until the density in and out values are the same (or within a set convergence threshold). The basic algorithm used by PWscf from the Quantum Espresso suite is shown below[115].



5.6.9 Exchange-Correlation Energy

The Kohn-Sham equations have an exchange-correlation energy and this functional is used to collect together the electron energy not captured in the non-interacting energy functionals.

The Pauli exclusion principle states that two fermions, half integer spin particles, cannot occupy the same quantum state. This is why electrons occupy a unique orbital within an atom defined by the quantum numbers n, l, m_l and m_s . Consider two particles at points \vec{r}_a and \vec{r}_b ; the probability amplitude of the wavefunction of these particles equals 1.

$$|\Psi(\vec{r}_a, \vec{r}_b)|^2 = 1 \quad (5.135)$$

Exchanging the particles must also give the same result; they still must exist somewhere with probability 1.

$$|\Psi(\vec{r}_b, \vec{r}_a)|^2 = 1 \quad (5.136)$$

Bosons, integer spin particles, are symmetric when exchanged:

$$\Psi(\vec{r}_a, \vec{r}_b) = \Psi(\vec{r}_b, \vec{r}_a) \quad (5.137)$$

Fermions, on the other hand, are antisymmetric:

$$\Psi(\vec{r}_a, \vec{r}_b) = -\Psi(\vec{r}_b, \vec{r}_a) \quad (5.138)$$

By setting up a wavefunction for two Fermions, and exchanging them, it can be seen why they cannot exist in the same state.

$$\Psi_{ab} = \psi_1(\vec{r}_a, \vec{r}_b) - \psi_2(\vec{r}_b, \vec{r}_a) \quad (5.139)$$

$$\Psi_{ab} = \psi_1(\vec{r}_a, \vec{r}_b) - \psi_2(\vec{r}_a, \vec{r}_b) = 0 \quad (5.140)$$

The XC term combines the difference between the real system and the fictitious non-interacting system set out in the Kohn-Sham equations. It also includes the difference between the quantum mechanical electron-electron repulsion and classical electron-electron repulsion. Unfortunately, whilst we know a functional exists, we do not know the exact form of it[116].

5.6.10 Pseudopotentials

Plane wave basis sets are used to help solve the TISE. The DFT code used in this work is Quantum Espresso, and the binary that carries out the calculations is named PWscf: plane wave self consistent field.

Where \vec{G} is the reciprocal lattice vector, a summation of plane waves may be used to construct each electronic wave function (eq. 5.141)[117].

$$\Psi_{\vec{k},n} = \sum_{\vec{G}}^{|G| < G_{max}} c_{\vec{k} + \vec{G},n} \exp(i(\vec{k} + \vec{G}) \cdot \vec{r}) \quad (5.141)$$

Unfortunately, the plane-wave basis sets required are too large to be used in practice, as they would need to represent the tight, rapidly oscillating inner orbitals as well as the valence electrons.

Bonding and material properties are primarily determined by valence electrons, not core electrons. Iron, for example, has two valence electrons in the 4s shell; however, it is a transition metal and the partially empty 3d shell is also important to consider. The core electrons do not contribute as much to the bond and the model may be simplified using pseudopotentials.

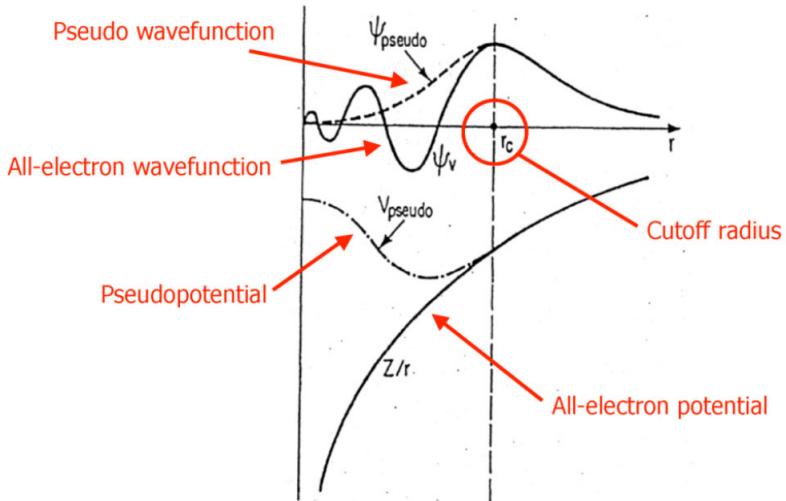


Figure 5.17: Replacing the complex potentials with a pseudopotential[118]

The eigen-energies of core electrons are also much larger than properties we see in materials, such as the cohesive energies and this also raises a concern that including the core electrons may introduce errors that are on a similar scale to the energies we are calculating.

The rapidly oscillating function in the core is replaced by a smoother pseudopotential. The same material properties are calculated as the valence electron functions are preferred, but a much smaller basis set is required to do so. There are a wide range of pseudopotentials available to use in DFT calculations, and one major distinguishing feature is how they treat the XC potential $v_{xc}([n]; \vec{r})$.

$$\left(\frac{1}{2} + V_{ext}(\vec{r}) + V_H(\vec{r}) + V_{xc}(\vec{r}) \right) \psi_{k,\sigma}(\vec{r}) = \epsilon_{k,\sigma} \psi_{k,\sigma}(\vec{r}) \quad (5.142)$$

The KS equation consists of several potentials including the exchange-correlation energy. This term represents the many-electron effects.

LDA

The local density approximation replaces the electron density with jellium, a homogeneous electron gas. Kohn and Sham, in their original 1965 work, introduced jellium as the density model and, despite its simplicity in

comparison to the real electron density distribution, it has worked well for simple metals.

$$E_{xc}^{LDA}[\rho] = \int d^3\vec{r}\rho(\vec{r})[e_x(\rho(\vec{r})) + e_c(\rho(\vec{r}))] \quad (5.143)$$

Several different LDA functionals have been developed over the years, including that of Perdew and Zunger. This functional takes the form given in eq. 5.144[119].

$$\epsilon_C[n(\vec{r})] = \begin{cases} \frac{-0.14231}{1+1.9529r_s(0.5)+0.334r_s} & r_s \geq 1 \\ -0.0480 + 0.0311lr(r_s) - 0.0116r_s + 0.0020r_sln(r_s) & r_s < 1 \end{cases} \quad (5.144)$$

LSDA

Electrons are fermions and have half integer spin. When trapped in a potential, such as an atom or a crystal lattice, electrons must take difference quantum states and one of the parameters is the spin of the electron. The LDA does not treat spin at all, but the LSDA splits the density into up and down spin (eq. 5.145[116]).

$$\begin{aligned} \rho_{\uparrow}(\vec{r}) &= \sum_k^{occ} |\phi_{k,\uparrow}(\vec{r})|^2 \\ \rho_{\downarrow}(\vec{r}) &= \sum_k^{occ} |\phi_{k,\downarrow}(\vec{r})|^2 \\ \rho(\vec{r}) &= \rho_{\uparrow}(\vec{r}) + \rho_{\downarrow}(\vec{r}) \end{aligned} \quad (5.145)$$

The XC energy is now dependent on the spin up and spin down density as well as the position (eq. 5.146[116]).

$$\begin{aligned} E_{xc}^{LSDA}[\rho_{\uparrow}, \rho_{\downarrow}] &= \int d^3\vec{r}\rho(\vec{r})[e_x(\rho(\vec{r}))f(\zeta(\vec{r})) + e_c(\rho(\vec{r}), \zeta(\vec{r}))] \\ \zeta &= \frac{\rho_{\uparrow} - \rho_{\downarrow}}{\rho_{\uparrow} + \rho_{\downarrow}} \\ f(\zeta) &= \frac{1}{2} \left((1 + \zeta)^{4/3} + (1 - \zeta)^{4/3} \right) \end{aligned} \quad (5.146)$$

The energy of the system now being solved is also dependent upon the charge density and spin polarization (eq. 5.147).

$$E = T[\rho, \zeta] + E_{ext}[\rho] + \frac{1}{2}E_{ee}[\rho] + E_{xc}[\rho, \zeta] \quad (5.147)$$

GGA

In the LDA and LSDA approximations, the electron density is that of an homogeneous electron gas that does not represent the real electron density. The density of this gas may change in space, but locally it is a constant density with no gradient.

$$E_{xc}^{GGA}[\rho_\uparrow, \rho_\downarrow] = \int d^3r f(\rho_\uparrow, \rho_\downarrow, \nabla \rho_\uparrow, \nabla \rho_\downarrow) \quad (5.148)$$

The GGA functional (eq. 5.148 [116]) takes the spin up spin down electron densities as well as their gradients.

The PBE functional was developed to address some of the short falls of the Perdew-Wang PW91 LSDA functional. These include an over complication in the formulation of PW91 and that it was designed to satisfy many exact conditions. The PBE development focused more on energetically significant conditions[120].

$$E_c^{PBE}[\rho_\uparrow, \rho_\downarrow] = \int d^3r \rho [e_c(r_s, \zeta) + H(r_s, \zeta, t)] \quad (5.149)$$

$$E_x^{PBE}[\rho] = \int d^3r \rho e_x(\rho) F_x(s) \quad (5.150)$$

The functional is split into a correlation energy (eq. 5.149) and exchange energy (eq. 5.150). ζ is the relative spin polarization $\zeta = (\rho_\uparrow - \rho_\downarrow)/(\rho_\uparrow + \rho_\downarrow)$ and r_s is the local Seitz radius where $\rho = 3/4\pi r_s^3$. The functions $e_x(\rho)$ and $e_c(\rho)$ are the exchange and correlation energy per electron of unpolarised uniform electron gas[116]. A full description of the functional is in the appendix ??.

LSDA vs GGA Potentials

Typically, GGA is an improvement over LSDA, with improvements in the total energies and structural energy differences[120]. Another short coming of LSDA is that it predicts FCC to be the optimal structure for pure iron at 0K, which is incorrect[121]. A list of success stories and failures were discussed shortly after the publication of the PBE functional and these are summarised in table ??[116].

Parameter	Experimental	LDA	GGA
a/angs	8.27	8.08	8.21
b/angs	4.80	4.74	4.81
c/angs	8.55	8.53	8.64
B_0 (Reuss)/GPa	146.8	156.9	141.9
B_0 (Voigt)/GPa	150.9	159.1	145.0
C_{11} /GPa	317.5	377.2	326.0
C_{22} /GPa	320.4	341.1	298.4
C_{33} /GPa	413.2	425.3	371.9
C_{44} /GPa	112.5	136.5	123.5
C_{55} /GPa	75.8	93.7	85.3
C_{66} /GPa	117.5	154.6	135.9
C_{12} /GPa	29.3	27.8	22.4
C_{13} /GPa	38.4	21.3	26.5
C_{23} /GPa	86.0	95.1	105.5

Table 5.5: Experimental vs LSDA vs GGA for $TiSi_2$ [90]

In the case of the compound Titanium Disilicide, $TiSi_2$, GGA is in better agreement, overall, with experiment than LDA, although several values are better predicted by the latter. As shown in table 5.5, the data shows

Successes of GGA

- atomization energy of molecules better
- binding energy curves more realistic
- Fe is bcc ferromagnet with GGA (fcc non-magnet with LSDA)
- Gives the anti invar effect in gamma-Fe and fc Fe-Mn
- Improvement on 4% accuracy for LSDA of alkali metal lattice constants
- Better lattice constants and bulk moduli of transition metals
- Isostructural transformation from open to close packed improved
- Successful calculation of a monovacancy in silicon
- Oxidation of the Si(001) surface using spin polarized GGA

Failures

- exchange-correlation holes can be unrealistic under some circumstances
- works better for exchange correlation together than either alone
- interaction of electrons in different shells poorly described

Figure 5.18: Successes and failures of the GGA functional[116]

that LDA is within 1.26% of the experimental lattice parameters, 6.16% of the experimental bulk modulus and 18.3% the value of the experimental elastic constants; several of these were particularly poor, with an almost 45% disagreement. On the other hand, GGA is within 0.66% of the experimental lattice parameters, 3.62% of the experimental bulk modulus and 14.97% the value of the experimental elastic constants[90].

Lanthanum aluminate ($LaAlO_3$) has also been studied using DFT, but this material is better modelled (to reproduce lattice parameters and bulk modulus) with either LDA or a specific form of GGA, the Perdew-Burke-Ernzerhof functional revised for solids (PBESOL) potential. The PBESOL is a revised PBE functional designed to better reproduce lattice parameters for solids, as shown in table 5.6.

Parameter	Experimental	LDA	GGA	GGA (PBESOL)
a/angs	3.78	3.74	3.82	3.77
B_0/GPa	215	224.00	190.35	206.41

Table 5.6: Experimental vs LSDA vs GGA vs GGA PBESOL for $LaAlO_3$ [122]

A complete library of pseudo-potentials is available through the Quantum Espresso website. Several of these have been tested with the PWscf code to calculate the lattice parameter and bulk modulus of simple metals including Aluminium and Sodium.

As can be seen from table 5.7, the PBE and PBESOL GGA type potentials are in general better at reproducing the lattice parameters. However, for Aluminium, the PZ LSDA type potential is in better agreement when calculating the bulk modulus. Overall, the GGA type potentials with the total self-consistent potential pseudized (these are the pseudopotential files that contain -n- in the name, rather than -nl-) perform best for these simple metals.

Whilst DFT is an exact method for solving the TISE, there are a number of approximations that still need to be made in addition to there being gaps in our knowledge (i.e. a lack of an exact functional for the XC).

Pseudo-potential	Element	a/angs	B_0/GPa
Experimental	Na	4.29[123]	6.3[123]
Na.pz-spn-kjpaw_psl.1.0.0	Na	4.06	8.72
Na.pbe-spn-kjpaw_psl.1.0.0	Na	4.20	7.67
Na.pbesol-spn-kjpaw_psl.1.0.0	Na	4.17	7.50
Experimental	Al	4.05[124]	76[124]
Al.pz-nl-kjpaw_psl.1.0.0	Al	3.98	78.6
Al.pz-n-kjpaw_psl.1.0.0	Al	3.98	78.6
Al.pbe-nl-kjpaw_psl.1.0.0	Al	4.04	91.3
Al.pbe-n-kjpaw_psl.1.0.0	Al	4.04	75.0
Al.pbesol-nl-kjpaw_psl.1.0.0	Al	4.01	87.7
Al.pbesol-n-kjpaw_psl.1.0.0	Al	4.01	79.0

Table 5.7: Experimental vs LSDA vs GGA - the DFT values were computed with a PWscf[125] using a 2x2x2 cell, 7x7x7 kpoints and ecutwfc=50

5.6.11 Ecut, K-Point Integration and Smearing

As discussed in section 5.6.5 the wavefunction can be expressed as a periodic function with the same period as the crystal lattice (eq. 5.151). The periodic function may then be written as a sum of plane waves (eq. 5.152).

$$\psi_{n\vec{k}}(\vec{r}) = u_{n\vec{k}}(\vec{r}) \exp(i\vec{k} \cdot \vec{r}) \quad (5.151)$$

$$u_{n\vec{k}}(\vec{r}) = \sum_{\vec{G}} V_{\vec{G}} \exp(i\vec{G} \cdot \vec{r}) \quad (5.152)$$

Solving the Kohn-Sham equations requires the matrix diagonalization of a size NxN where N is the number of planewaves for each k-point in the Brillouin Zone[126]. The number of planewaves can be reduced to satisfy eq. 5.153. In practice this will mean reducing E_{cut} whilst ensuring the desired accuracy is still met.

$$\frac{\hbar^2 G^2}{2m} <= E_{cut} \quad (5.153)$$

In order to compute the energy using HK, the charge density in the BZ is calculated. It is impossible to diagonalise the matrix at an infinite number of k-points. The integral is replaced with a summation over selected points which in turn are weighted (eq. 5.154)[127].

$$\rho(\vec{r}) = \frac{1}{\Omega_{bz}} \sum_i \int f(\vec{k}, i) \psi_{i,\vec{k}}^*(\vec{r}) \psi_{i,\vec{k}}(\vec{r}) d\vec{k} \frac{1}{\Omega_{bz}} \int_{BZ} d\vec{k} \rightarrow \sum_{\vec{k}} \omega_{\vec{k}} \quad (5.154)$$

Due to Bloch theorem and the Born-Von-Karman boundary conditions, only the first BZ needs to be sampled. As discussed earlier, if there are further symmetries in the unit cell, only the IBZ need to be sampled.

The integral over k-space is replaced by a set of points in k-space that are sampled. The higher the number of points, the higher resolution the space is sampled in, but the longer the calculation will take. The Gamma point

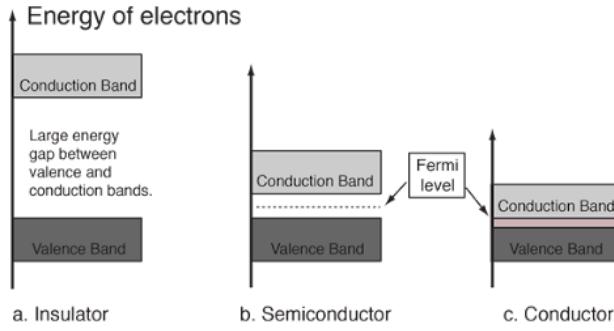


Figure 5.19: Energy gap between valence and conduction bands[129]

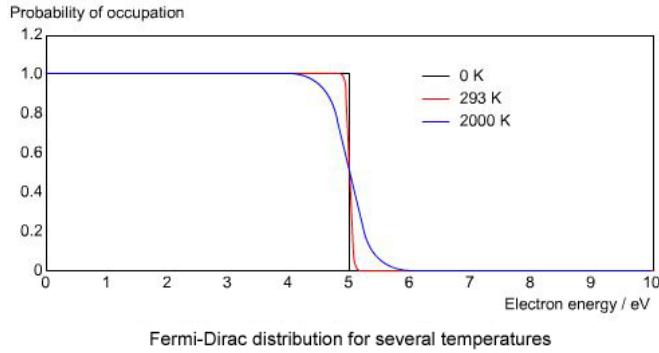


Figure 5.20: Probability $F(E)$ of finding a fermion with energy E at several temperatures[130]

Γ is a high symmetry point in reciprocal space and is located at $(0,0,0)$. It may be used as the only sampling point in some DFT calculations.

The Monkhorst-Pack grid is a special set of k -points. They are distributed evenly in reciprocal space and may be aligned such that either one point coincides with Γ or such that 8 points closest points surround Γ at an equal distance. By offsetting the grid from Γ and the coordinate axis of reciprocal space, the sampling better avoids points of high symmetry.

Where the system is spin degenerate, each state is occupied by 2 electrons for the lower $\frac{N_e}{2}$ states ($f(\vec{k}, i) = 2$), and zero otherwise ($f(\vec{k}, i) = 0$)[128]. This drop, from states occupied by two electrons to empty states occurs at the Fermi surface and it results in a discontinuity in the functions being integrated over in eq. 5.154. This is the case for an electron gas at absolute zero, but if the gas is heated then there is sufficient energy for electrons to occupy states above the Fermi energy.

Fermi-Dirac statistics determine the probability of a Fermion having an energy E (eq. 5.155). As the temperature increases the probability of finding a fermion above the Fermi energy increases (fig. 5.20). This also smooths the discontinuity at the Fermi surface between the occupied and unoccupied states (at 0K).

$$F(E) = \frac{1}{\exp(\frac{E-E_F}{kT}) + 1} \quad (5.155)$$

To avoid having to integrate the BZ with a very fine mesh, a smearing function is introduced to remove the discontinuity at the Fermi surface, making the integral over the BZ differentiable at every point. The relationship between the delta and Heaviside step is used to replace the step function when integrated (eq. 5.156).

$$\int_{-\infty}^{\infty} \delta(k) dk = \Theta(x) \quad (5.156)$$

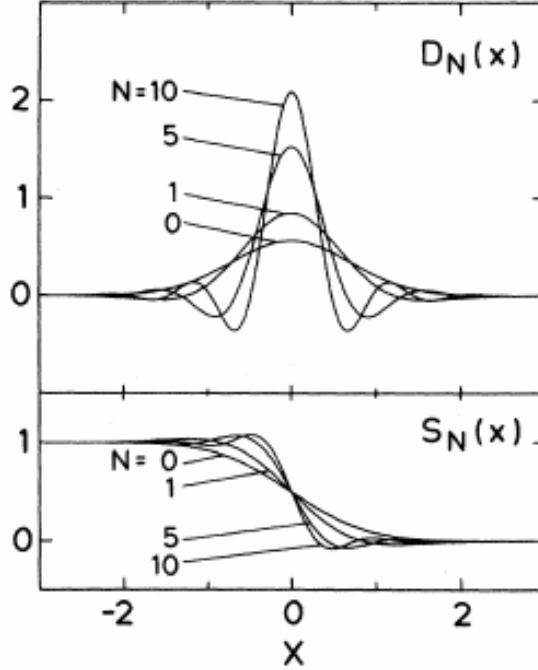


Figure 5.21: Delta function replaced by successive Hermite polynomials[131]

$$I = \int_{-\infty}^{\infty} S(\epsilon - E_F) \int_{BZ} f(\vec{k}) \delta(\epsilon - E(\vec{k})) d\vec{k} d\epsilon \quad (5.157)$$

The step function at the Fermi surface (eq. 5.157), that causes the discontinuity, may be replaced with a smearing function[131]. Four of these functions are available to use in the PWscf DFT code.

The Fermi-Dirac function, as mentioned above, is used to heat the system. It removes the discontinuity but the SCF will converge to the wrong energy (eq. 5.158).

$$S_{FD}(\epsilon) = \frac{1}{\exp(\frac{\epsilon - E_F}{kT}) + 1} \quad (5.158)$$

A Gaussian smear is also used to replace the step function although, unlike the Fermi-Dirac function, it has no physical meaning (eq. 5.159).

$$S_G(\epsilon) = \frac{1}{2} \left[1 - \text{erf} \left(\frac{\epsilon - \mu}{\sigma} \right) \right] \quad (5.159)$$

The Methfessel-Paxton method attempts to correct the errors introduced by the previous smearing functions where the delta function is replaced with Hermite polynomials (fig. 5.21 and eq. 5.161). A drawback of this method is that it allows for negative occupancy of electron states[128].

$$x = \epsilon - E_F \quad S_{MP,N}(x) = \frac{1}{2} [1 - \text{erf}(x)] + \sum_{n=1}^N A_n H_{2n+1}(x) \exp(-x^2) \quad A_n = \frac{(-1)^n}{n! 4^n \sqrt{\pi}} \quad (5.160)$$

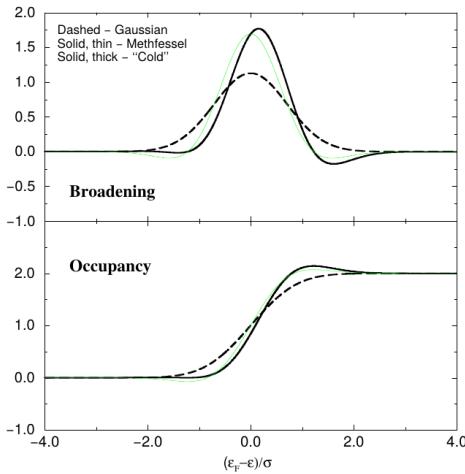


Figure 5.22: The occupancy for the Marzari-Vanderbilt smearing function is always positive, unlike the Methfessel-Paxton function[132]

The Mazari-Vanderbilt function was developed to address the short falling of the Methfessel-Paxton function. The occupancy where this function is used is always positive (fig. 5.22).

$$x = \frac{\mu - \epsilon}{\sigma} \delta(x) = \frac{1}{\sqrt{\pi}} \exp\left(-\left(x - \left(\frac{1}{\sqrt{2}}\right)\right)^2\right) \left(2 - \sqrt{2}x\right) \quad (5.161)$$

5.7 Classical Molecular Dynamics

5.7.1 Introduction

Ab initio (DFT) calculations approximately solve the Schrödinger equation to calculate the energy, forces and stress of a given simulation. They take a relatively long time to calculate, have relatively small numbers of atoms (hundreds to thousands) and are fixed at one moment in time, although there are now programs that will run DFT MD. In comparison, Molecular Dynamics model a collection of atoms over a specified period of time. Much larger collections of atoms are possible (thousands to millions) and the interaction between atoms are predefined by interatomic potentials.

Molecular dynamics has been used to investigate the effects of radiation on materials. The damage event on the atomic scale is rapid in time, in the femtosecond to picosecond range, and affects a small volume of the material whereas the effects of the damage to the material on a mesoscopic and macroscopic scale may range from days to decades. In reactor pressure vessels, for instance, the damage and defect production will cover a large range of time and size scales[25]:

- $10^{-10} m$ to $10^{-3} m$ (10^{18} to 10^{25} atoms)
- $10^{-17} s$ to $10^9 s$ (tens of attoseconds to 30 years)

As a major element of steel, iron has often been the subject of MD simulations. Damage cascades have been studied in iron at a range of temperatures and PKA energies.

5.7.2 Molecular Dynamics Codes

Many MD codes are available for researchers to download freely and use from the Internet (although, terms may apply to the use). These include DL POLY (Fortran), Large-scale Atomic/Molecular Massively Parallel Simulator (C++) and MOLDY (C). Most are able to run using message passing interface to run the simulation in parallel across many compute nodes, with others being modified to take advantage of graphical processing units.

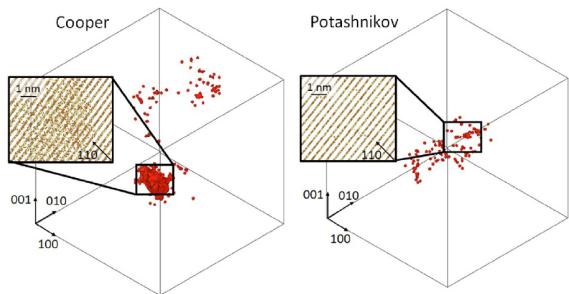


Figure 5.23: Damage in MOX (LAMMPS)[133]

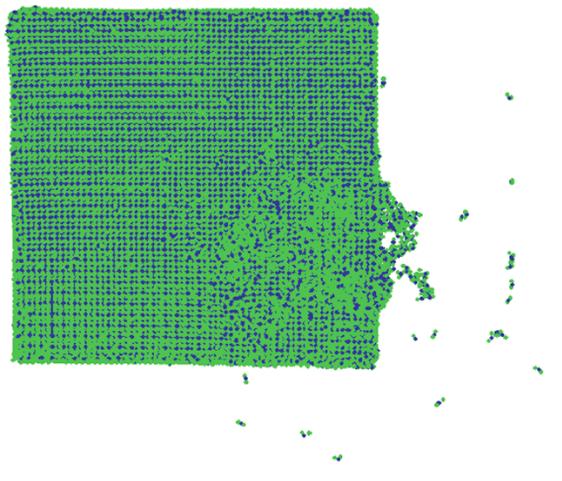


Figure 5.24: Sputtering of plutonium oxide (LAMMPS)[134]

Damage cascades in mixed oxides have been modelled recently using the LAMMPS code. A $70 \times 70 \times 70$ box with sides approximately 40nm in length and containing 4,116,000 atoms was used with PKAs having energies ranging from 5 to 75keV[133]. The sputtering of material from the surface of plutonium oxide (IV) has also been modelled with LAMMPS using a box of 393,216 atoms with a PKA of 87.7keV. The timescale of the sputtering simulation was 7.8 picoseconds (7.8×10^{-12}) and the entire event was split into 1.0×10^5 time steps of 7.8×10^{-17} s.

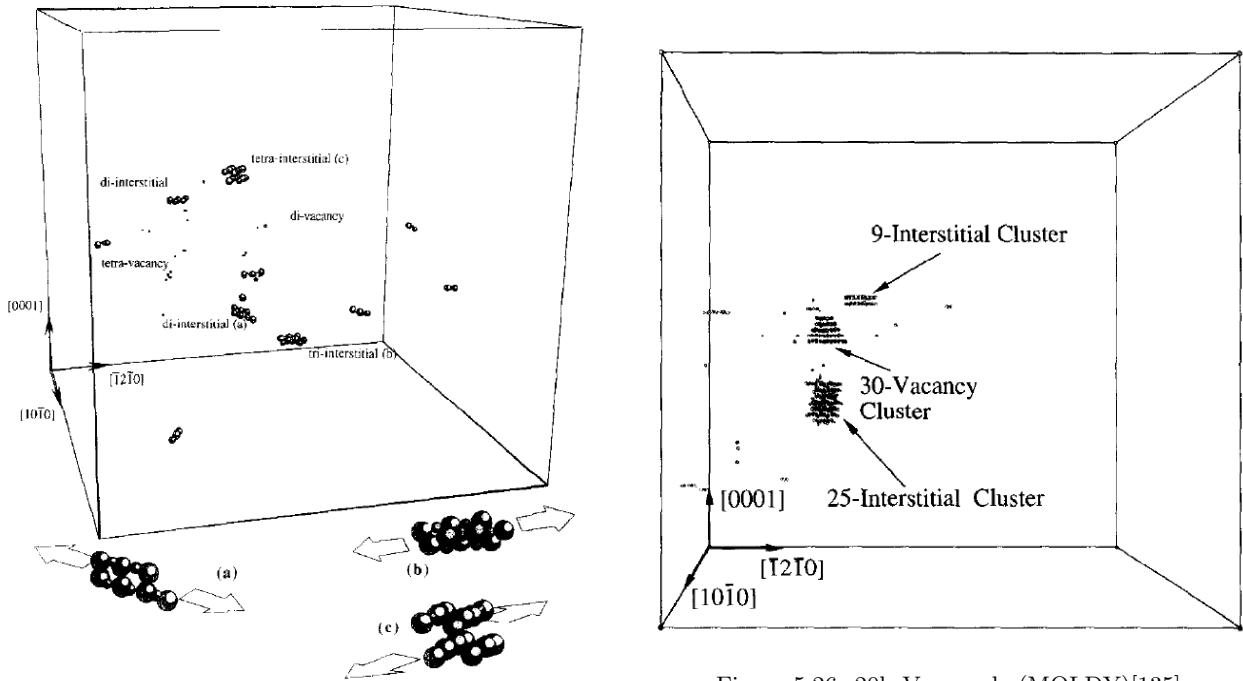


Figure 5.26: 20keV cascade (MOLDY)[135]

Figure 5.25: 5keV cascade (MOLDY)[135]

MOLDY has previously been used to model primary knock on atoms in metals, at energies of 5kev, 10keV and 20keV which would be slightly lower than the typical iron PKAs close to U235 fuel in the reactor (fig. 1.10). This simulation however was in alpha-zirconium and the higher mass of zirconium, relative to iron, would lead to lower energy PKAs. The damage cascades were modelled in a 104,832 atom block for the 5keV PKA (figure 5.25) and a 445,536 atom block for the 20keV PKA (figure 5.26).



Figure 5.27: 100keV cascade after 50ps in TiO₂ (DL_POLY)[136]

DL_POLY has also been used to model damage cascades in various oxides (titanium, aluminium, silicon)[136]. The PKA energies are higher than those performed in MOLDY 15 years earlier, but still under 1MeV. The authors expect 1MeV simulation boxes to require at least 1 billion atoms, but the 100-200keV PKAs they modelled were within 10 million atom simulation boxes. Slightly larger boxes and damage cascades required longer simulation times, and with a box size of 43nm per side, a 50ps simulation time was used (fig. 5.27).

5.7.3 Neighbour List

The simple neighbour list creation has already been discussed earlier in this chapter (section 5.3.6), but many modern MD codes are designed to run on computers with many thousands of processor cores.

In the case of DL_POLY a link-cell domain decomposition is used. The supercell is divided up into subcells, and these must be at least the size of the cut-off radius. Each processor will work on a geometric domain and these will only interact with their immediate neighbours. The halo data for each subcell is only passed between processors before and after the calculation of the equations of motion of the atoms within each subcell.

5.7.4 Velocity Verlet Algorithm

Verlet integration is used by MD codes such as DL_POLY, LAMMPS, Democritus as well as MD DFT codes, as in the case of CASTEP.

First, the forces at time step n are calculated \vec{f}_n . Now, using these forces and the velocity at time step n , the half step velocity is computed (eq. 5.162).

$$v_{n+\frac{1}{2}} = \vec{v}_n + \frac{\vec{f}_n \delta t}{2m} \quad (5.162)$$

The position of the next step, $n + 1$, is calculated using the position at n , the time step dt and the half step velocity (eq. 5.163).

$$\vec{r}_{n+1} = \vec{r}_n + v_{n+\frac{1}{2}} \times \delta t \quad (5.163)$$

The forces are recalculated at the new position \vec{r}_{n+1} to give \vec{f}_{n+1} and this, along with the half step velocity, is used to compute the velocity at step $n+1$, v_{n+1} (eq. 5.164).

$$\vec{v}_{n+1} = v_{n+\frac{1}{2}} + \frac{\vec{f}_{n+1} \delta t}{2m} \quad (5.164)$$

This process is repeated for each time step δt from the start of the simulation to the end.

5.7.5 Selecting a Time Scale

Take, for example, a 50x50x50 BCC Iron supercell with a lattice parameter of 2.87 angstrom. The supercell measures almost 15nm on each side. Simulating a thermal neutron, travelling at 2,200 m/s, would require an overall time period of at least 7 picosecond to capture the neutron passing through the supercell. Higher energy particles would require smaller overall time periods, divided up into as many time steps as the user requires.

If PKA damage is being modelled, there may be a point where the depth of the damage cascade becomes larger than the supercell itself. Damage cascades of 500KeV iron atoms into an iron target travel to a depth of up to 300nm, which would require a simulation supercell at least 1,000 BCC cells deep. Kinetic Monte Carlo would be better suited at simulating higher energy damage cascades.

To capture the required details the time steps would also need to be very fine, on the order of attoseconds or tens of attoseconds. This has been shown in the literature with time steps of 7.8×10^{-17} s[133].

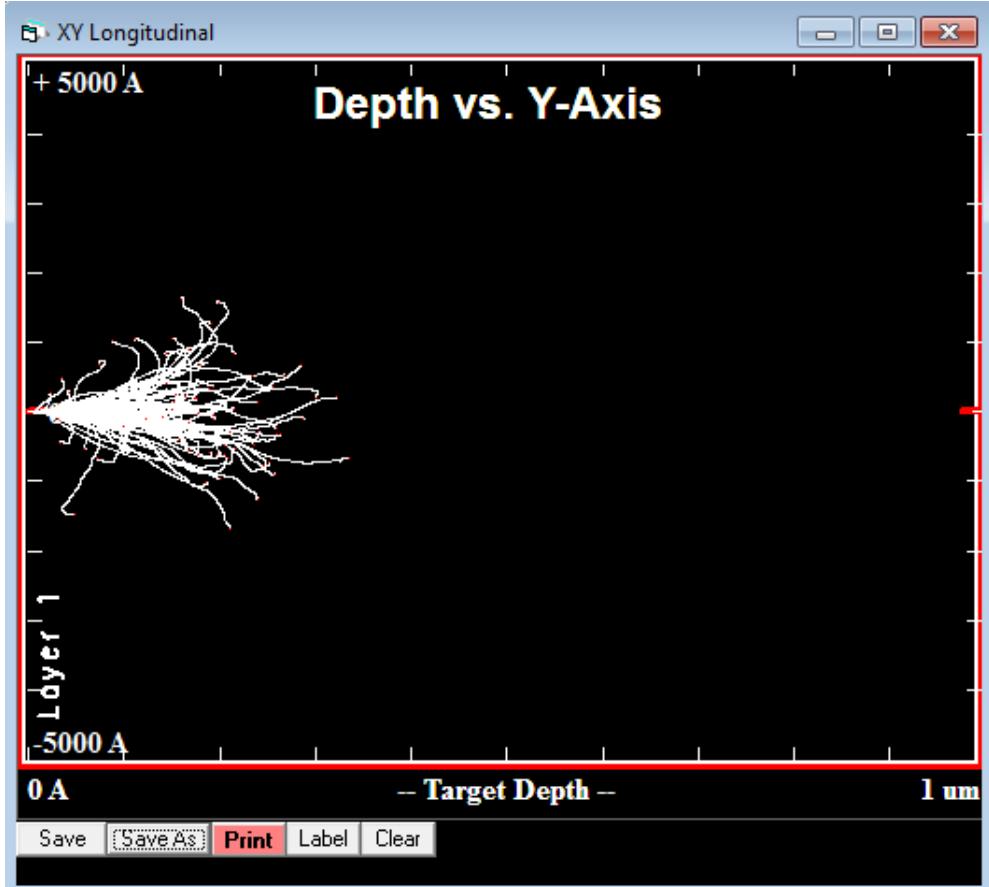


Figure 5.28: Damage cascade: 500kev iron projectiles into an iron target calculated with SRIM

5.8 Optimization

5.8.1 Introduction

Optimization is the process of finding the best solution for a problem that may also need to satisfy a number of imposed constraints. In terms of this work, there are several tasks where optimization plays a key role. The DFT code used, PWscf, employs the Broyden Fletcher Goldfarb Shanno algorithm to relax structures to give the optimum volume and optimal structure of an arrangement of atoms. During the fitting of the variety options of equation of state to the energy-volume points, the Levenberg-Marquardt Algorithm is used. Finally, a genetic algorithm and simulated annealing algorithm are coupled with LMA to locate the global optimal parameters for interatomic potential functions.

5.8.2 Continuous and Discrete Optimization

Problems that may be solved using discrete optimisation have, as the name suggests, discrete solutions. An example would be the travelling salesman problem, where the aim is to minimise the distance travelled by the salesman between cities. An example of the solution may be city E to A to B to C to D. There's a set of possible solutions to the problem. If the problem was changed such that any point within a 1 mile radius of the centre of each city is a valid solution, and the aim is to find the optimal points to travel between within the area of each city, it would become a continuous problem.

Fitting interatomic potentials to data is an example of continuous optimisation. The Morse potential takes three parameters plus the separation r between the two atoms. If the parameters are being optimised such that

Morse potential matches the experimental forces (assuming they could be measured experimentally) between two atoms for a set of separation values, the choice of parameters isn't discrete; they are taken from a continuous range of real numbers.

5.8.3 Constrained and Unconstrained Optimization

There are constraints that may need to be considered, and constraints that apply to the fitting of interatomic potentials are as follows:

- continuous well behaved function
- continuous first-derivative, smooth change in force wrt r
- positive electron density
- smooth cutoff at some $r = r_{cut}$
- repulsion (ZBL/exponential) as r approaches 0
- continuous second-derivative
- enough parameters coupled with a good choice of functions so the potential is flexible enough to replicate all the experimental data well
- a small enough range of parameters to keep the parameter space small enough to search in a reasonable amount of time
- maintain physical elegance
- ignore physical elegance

Certain constraints listed will battle one-another, while others may be dropped, such as restricting to positive electron-densities if the choice to ignore physical elegance is selected.

5.8.4 Global and Local Optimization

An example to discuss the difference between local and global optimization will be given in a very literal sense. I am located near Birmingham, and I wish to find the highest point local to me. Using a modern smartphone and map I could quickly find the nearest hill to my current position; this would be the local maxima. I would, however, need to go up and down many peaks and troughs, scouring the entire surface of the Earth, until I found the global maxima, somewhere amongst the many peaks and troughs in the Himalayas.

When optimising the parameters of a function, it is relatively easy to find a local extreme that would give me the optimum parameters locally. It is a much harder task to find the global optimum, especially if very little, or nothing at all, is known about the function for which the parameters are being optimised.

5.8.5 Global Optimization Algorithms

Simulated Annealing

A large amount of computing power is required today to solve many problems in Physics. The optimum arrangement of Iron atoms, at various concentrations and temperatures is one such example, but nature knows

the correct structure it should take because it has access to the rule book. It makes perfect sense to look to nature when designing optimisation algorithms.

Simulated annealing mimics the way a solid cools, with its atoms settling into the optimum relaxed positions as time passes. Initially, the solid is hot and the atoms are able to take non-optimal positions, but as the solid cools the atoms take their optimal places. The heating allows the atoms to move from non-optimal positions in the initial configuration.

Listing 5.3: Pseudocode for the simulated annealing algorithm

```
1 # Calculate the starting rss
2 rss = get_rss(p)
3
4 # Loop (n times, until a threshold is met, time runs out etc)
5 while(loop)
6 {
7     p_new = vary(p)          # Make a new set of parameters
8     rss_new = get_rss(p_new) # Get rss for these parameters
9
10    # Always accept BETTER
11    # Sometimes accept WORSE (depends on temperature and how bad it is)
12    if(rss_new < rss or rand(0.0,1.0) < exp((rss-rss_new) / temp))
13    {
14        p = p_new
15        rss = rss_new
16    }
17 }
```

The simulated annealing algorithm takes in a starting set of parameters. These are varied and, if the new parameters give a better solution to the problem, the parameters are updated. However, there is a chance that a worse set of parameters are used, and this depends on how bad they are and the current value of the temperature in the algorithm (listing 5.3). By allowing a worse set of parameters to be used, it gives the solution the chance to jump out of a local minimum and explore other solutions that it would otherwise be oblivious to.

The algorithm used in this work is available in the appendix (section ??).

Genetic Algorithm

It is unclear whether life started on this planet, or was brought to this planet, and this idea only removes the problem of how life as we are able to understand it was conceived to a different time and place. However it started, there is evidence of life on this planet approximately 3.5 billion years ago. Prokaryotes were the simple single cell organisms that inhabited our planet for billions of these. Their cell does not have a nucleus and the genetic information is contained in RNA and DNA within the cell.

Approximately 1.5 billion years ago, life made a leap forward in complexity and evolved into the first Eukaryotic cells. These are larger and more complex than Prokaryotes, and the genetic material is stored within a nucleus. Through processes of inheritance, variation, natural selection and the vast expanse of time, more complex organisms developed.

A set of solutions are bred with one another, and where there is an improvement the new variation is reinserted into the gene pool. There is a danger that the population can become a set of clones, so a large pool of sets of parameters is required, fresh parameters proposed every so often as well as a mechanism to prevent clones appearing.

As simulated annealing takes inspiration from cooling solids, genetic algorithms take inspiration from evolution. The algorithm used in this work is available in the appendix (section ??).

5.8.6 Local Optimization Algorithms

Gradient Descent

The gradient descent method is a reliable way of searching for a local minimum. It requires only the first derivative coupled with a line search to move closer to the minimum point. Close to the minimum, it is slower than the Newton-Gauss and LMA, but these require the computation of the Jacobian to estimate the Hessian matrix, so these are more computationally intensive. A pseudocode for the algorithm is available in the appendix (section ??).

Newton-Gauss

The Newton-Gauss method is an algorithm that finds the local minimum of a function by approximating the Hessian matrix with the Jacobian and its transpose (eq. 5.165). By estimating the Hessian, the algorithm only requires the computation of the first order derivatives with respect to each parameter, at each point pair x , y . It is possible to use the function and the analytical derivative, but as this work requires the calculation of functions that have no simple analytical form, the first derivative will be approximated within the algorithm.

$$\begin{aligned} \vec{J}^T \vec{J} \vec{p} &= \vec{J}^T \vec{r} \\ \vec{r} &= \vec{y} - f(\vec{x}, \vec{p}) \end{aligned} \tag{5.165}$$

Close to a minima, the NG method quickly moves to the optimum point. Depending on the starting point, it may converge only to a local minimum, and not the global, and it may also be unstable, not converging at all.

Levenberg Marquardt

The LMA is a NG type algorithm that includes the addition of a dampening term that helps to increase the robustness of the algorithm. The particular algorithm here uses a number of other schemes to improve the overall effectiveness of the algorithm. The equation at the center of the algorithm is similar to that of NG (eq. 5.166).

$$\begin{aligned} (\vec{J}^T \vec{W} \vec{J} + \lambda \text{diag}(\vec{J}^T \vec{W} \vec{J})) \vec{p} &= (\vec{J}^T \vec{W} \vec{r}) \\ \vec{r} &= \vec{y} - f(\vec{x}, \vec{p}) \end{aligned} \tag{5.166}$$

The starting value for lambda is calculated as a function of the estimate of the Hessian and a cutoff for lambda is introduced to remove dampening altogether. A delayed gratification scheme has also been introduced to increase lambda by 50 percent if the trial solution is worse, and to decrease lambda by a factor of 5 if the trial solution is better. Finally, a diagonal weighting matrix has been included to allow certain parameters to be preferential during their optimisation.

Chapter 6

Methodology: Proton Activation and Radioactive Decay

A computer program was developed to calculate the radioactivity of an ion irradiated target. The choice of projectile was limited to protons, but there is the option to increase this to other light ions. The decay equations were derived to be able to calculate the activity of isotopes within the target at time t. The simulation is split into two sections: the first with the beam on, where there will be source terms for isotopes, and the second with the beam off and no source terms.

SRIM, an ion transport code, was used to calculate ion trajectory data. Iron was irradiated experimentally by 36MeV protons with the University of Birmingham cyclotron, and its activity was measured and calculated using the activity code. Two versions of the code were created.

The equations and code were then used to predict the activity of an iron sample irradiated to 100DPA for a range of proton energies. This is a reasonable damage dose that is expected in Gen IV reactors.

6.1 Introduction

High flux neutron reactors are expensive to use, whereas proton accelerators capable of producing beams of adequate fluence and energy are more readily available, cheaper to buy and cheaper to run. Ion beams, due to the Coulomb interaction, are more controllable in terms of energy, direction and fluence. They may be concentrated on a desired target at a set fluence and energy.

Depending on the energy of the ion beam, the target material will become radioactive. The stable nuclei are transmuted and when the resulting isotope is unstable it will decay. An equation was derived to predict the activity of isotopes for any decay chain with source terms and branching factors included. Two versions of a computer code were developed to compute the reaction rates and activity of the irradiated targets.

6.2 Activation by Ion Irradiation

The Bateman equation was derived using Laplace transforms, and this same method has been used to develop a modified equation that incorporates branching factors and production rates for each isotope in the decay chain, as illustrated by fig. 6.1.

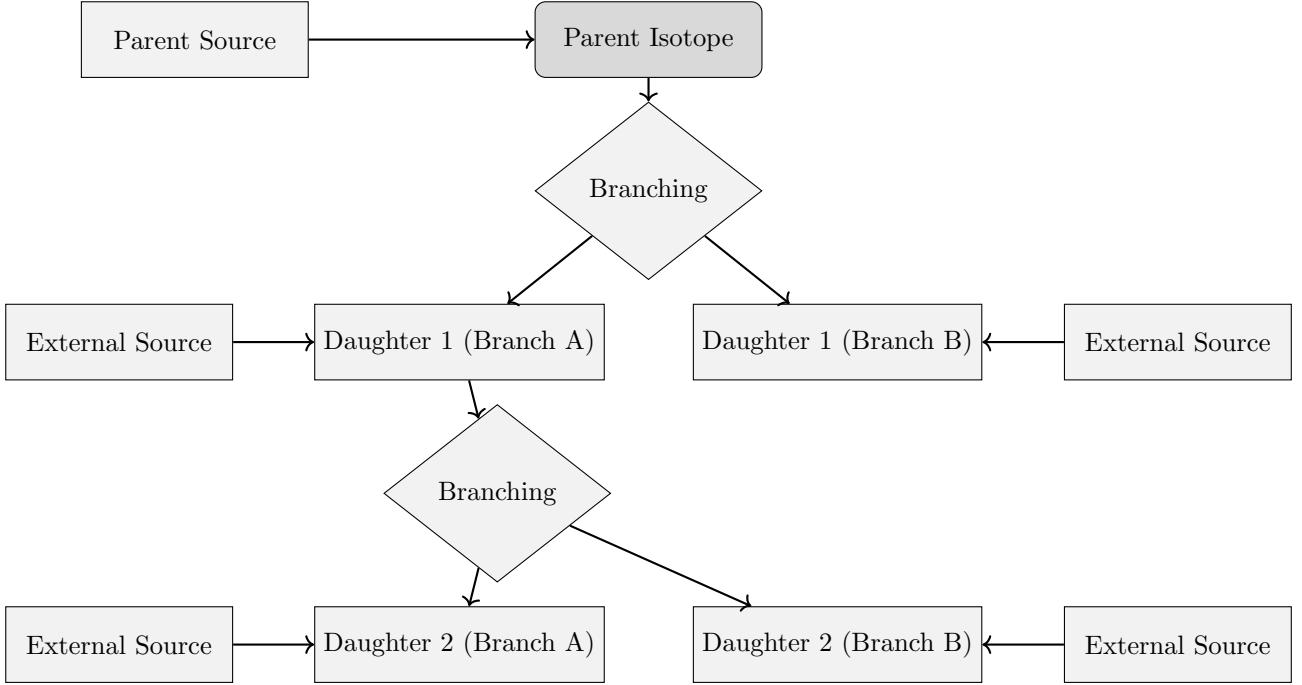


Figure 6.1: An example of several decay chains including branching factors and possible external source terms for each isotope on each chain.

6.2.1 Laplace Transform

Laplace Transforms (eq. 6.1) are a useful mathematical tool, and allow ordinary differential equations to be solved by simple algebraic manipulation in the s domain. Bateman took advantage of Laplace Transforms in deriving his equation, and this is the method that has been taken here as well.

$$F(s) = \int_0^{\infty} f(t) \exp(-st) dt \quad (6.1)$$

6.2.2 Constructing the Differential Equations

The first step is to set up differential equations for the parent isotope, unstable daughter isotopes and stable daughter isotope. The parent isotope has a source term, due to production, and a loss term, due to decay. The unstable daughter isotopes have two source terms, from the production by irradiation induced transmutation and the decay of preceding isotopes in the decay chain, and a loss term, due to decay. Finally, the stable daughter that finalizes the decay chain has two source terms (the same as the unstable daughters) but no loss term.

The variables (and vectors) used in these equations are defined as follows:

- $\vec{\lambda}$ vector containing isotope decay constants λ_i
- \vec{b} vector containing isotope to isotope branching factors b_i
- \vec{w} vector containing isotope production rates w_i
- t time at which activity/amount of isotope is measured
- $N_i(0)$ starting amount of the i^{th} isotope
- $N_i(t)$ amount of the i^{th} isotope at time t

- $N'_i(t)$ change in amount of the i^{th} isotope, with respect to time, at time t

The differential equations for the parent isotope (first isotope), unstable daughter isotopes (i^{th} isotopes) and stable, final, daughter isotope (z^{th} isotope) in the time domain are as follows:

$$N'_1(t) = \omega_1 - \lambda_1 N_1(t) \quad (6.2)$$

$$N'_i(t) = \omega_i + b_{i-1} \lambda_{i-1} N_{i-1}(t) - \lambda_i N_i(t) \quad (6.3)$$

$$N'_z(t) = \omega_z + b_{z-1} \lambda_{z-1} N_{z-1}(t) \quad (6.4)$$

Applying the Laplace Transform to these three differential equations allows them to be manipulated and solved algebraically in the s-domain.

$$N_1(s) = \frac{1}{s + \lambda_1} N_1(0) + \frac{1}{s(s + \lambda_1)} \omega_1 \quad (6.5)$$

$$N_i(s) = \frac{1}{s(s + \lambda_i)} (\omega_i) + \frac{1}{s + \lambda_i} (b_{i-1} \lambda_{i-1} N_{i-1}(s)) + \frac{1}{s + \lambda_i} N_i(0) \quad (6.6)$$

$$N_z(s) = \frac{1}{s^2} \omega_z + \frac{1}{s} b_{z-1} \lambda_{z-1} N_{z-1}(s) + \frac{1}{s} N_z(0) \quad (6.7)$$

6.2.3 Numerical Inversion of the Laplace Transform

The Gaver-Stehfest[137] algorithm was developed in the 1960s and 1970s and is a method of calculating the inverse of a Laplace Transform in the real number domain. It is an easy to implement and reasonably accurate method, although it is an approximation to the real value. A comparison between an analytic and numeric inversion for the unstable isotope Po-218 is discussed at the end of this section (figure 6.2).

$$f(t) \approx f_n(t) = \frac{\ln(2)}{t} \sum_{k=1}^{2n} a_k(n) F(s) \text{ where } n \geq 1, t > 0 \quad (6.8)$$

$$s = \frac{k \ln(2)}{t} \quad (6.9)$$

$$a_k(n) = \frac{(-1)^{(n+k)}}{n!} \sum_{j=\text{Floor}(\frac{k+1}{2})} j^{n+1} \binom{n}{j} \binom{2j}{j} \binom{j}{k-j} \quad (6.10)$$

The equation for the i^{th} isotope may be calculated by recursively calculating the equations by numeric inversion, starting from the first (parent isotope) and inserting the result into each subsequent recursion until the i^{th} isotope is reached (changing the equations appropriately for the parent, unstable daughter and stable daughter isotopes).

6.2.4 Analytic Solution by Partial Fraction Expansion

The equation for the i^{th} isotope in the s domain can be written in full by substituting the preceding equation until the parent isotope is reached, and this full equation may be rearranged with the production amount of

each isotope and starting amount of each isotope in individual terms. Each of these terms is multiplied by a fraction that can be expanded, using partial fractions, and inverted analytically.

This is illustrated with an example unstable isotope, fourth in the decay chain (including the parent isotope) (eq. 6.11).

$$\begin{aligned}
N_4(s) = & \frac{1}{(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\
& + \frac{1}{(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\
& + \frac{1}{(s + \lambda_3)(s + \lambda_4)} b_4 \lambda_3 N_3(0) \\
& + \frac{1}{(s + \lambda_4)} N_4(0) \\
& + \frac{1}{s(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\
& + \frac{1}{s(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\
& + \frac{1}{s(s + \lambda_3)(s + \lambda_4)} b_4 \lambda_3 \omega_3 \\
& + \frac{1}{s(s + \lambda_4)} \omega_4
\end{aligned} \tag{6.11}$$

An example stable isotope, fourth (last) in the decay chain (including the parent isotope) (eq. 6.12).

$$\begin{aligned}
N_4(s) = & \frac{1}{s(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\
& + \frac{1}{s(s + \lambda_2)(s + \lambda_3)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\
& + \frac{1}{s(s + \lambda_3)} b_4 \lambda_3 N_3(0) \\
& + N_4(0) \\
& + \frac{1}{s^2(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\
& + \frac{1}{s^2(s + \lambda_2)(s + \lambda_3)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\
& + \frac{1}{s^2(s + \lambda_3)} b_4 \lambda_3 \omega_3 \\
& + \frac{1}{s^2} \omega_4
\end{aligned} \tag{6.12}$$

By using partial fraction expansion and standard Laplace Transforms, the set of equations (eqs. 6.13, 6.14, 6.15, 6.16) is used to calculate the amount of the m^{th} isotope in the decay chain, providing the m^{th} isotope is unstable.

$$N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = \sum_{k=1,m} r(k; \vec{\lambda}, \vec{b}) [f(t; k, m, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k] \tag{6.13}$$

$$r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k, m-1} (b_{i+1} \lambda_i), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \quad (6.14)$$

$$f(t; k, m, \vec{\lambda}) = (-1)^{m-k} \sum_{i=k, m} \left[\exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (6.15)$$

$$g(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k, m} \lambda_i} + (-1)^{m-k+1} \sum_{i=k, m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (6.16)$$

The set of equations (eqs. 6.17, 6.18, 6.19, 6.20) is used to calculate the amount of the m^{th} isotope in the decay chain, where the m^{th} isotope is stable.

$$N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = N_m + w_m t + \sum_{k=1, m-1} r(k; \vec{\lambda}, \vec{b}) \left[f(t; k, m-1, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k \right] \quad (6.17)$$

$$r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k, m-1} (b_{i+1} \lambda_i), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \quad (6.18)$$

$$f(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k, m} \lambda_i} + (-1)^{m-k+1} \sum_{i=k, m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (6.19)$$

$$g(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k, m} \lambda_i} t - \frac{\sum_{i=k, m} \left[\prod_{j=k, m; j \neq i} \lambda_j \right]}{\prod_{i=k, m} \lambda_i^2} + (-1)^{m-k} \sum_{i=k, m} \left[\frac{\exp(-\lambda_i t)}{\prod_{j=k, m; j \neq i} (\lambda_i - \lambda_j)} \right] \quad (6.20)$$

6.2.5 Preference: Analytic over Numeric

The numeric solution only requires the equation to be solved in the s-domain; the Gaver-Stehfest algorithm performs the inversion. It is worth the extra effort to derive and implement an analytic solution, as the numeric is only an approximation. Examples of the pitfalls of the numeric solution are that it can give negative amounts of an isotope and the difference between the numeric and analytic calculated amounts can become quite large when the isotope decays away to a very small value. Fig. 6.2 shows the predicted decay of a sample of Po-218 irradiated for 1,000s, and sampled until 10,000s. In the region between 4,000s and 9,000s the amount from the numeric calculation drops below zero, whereas the analytic calculation remains above zero, as would be expected.

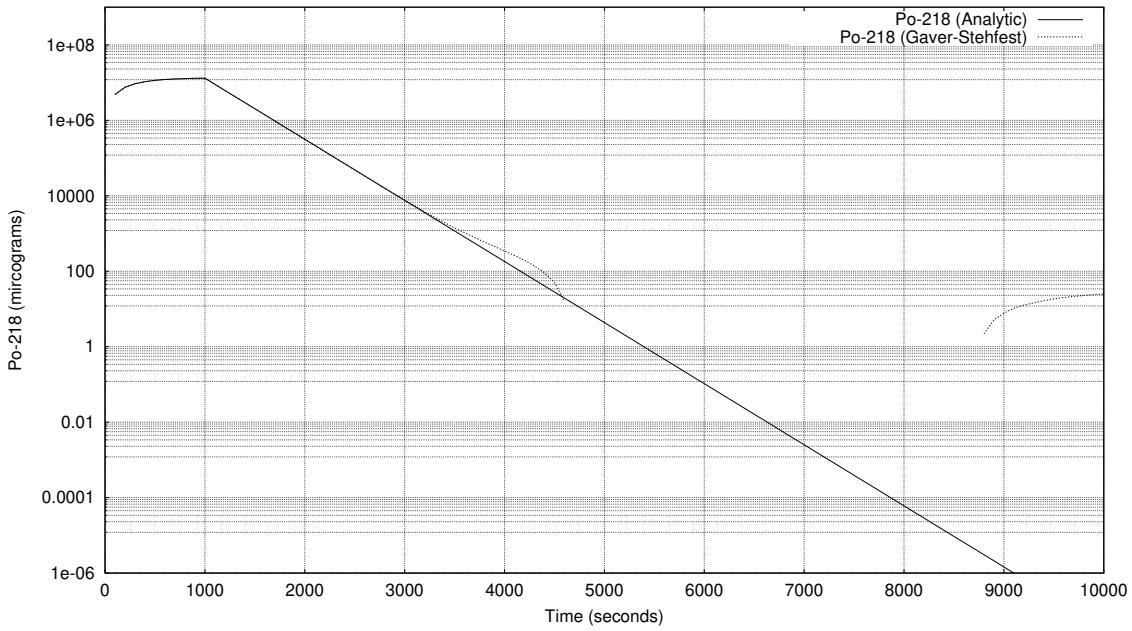


Figure 6.2: Decay of Po-218: Analytic and Gaver-Stehfest Calculations [22]

6.3 Computational Methods

6.3.1 Activity V1

The Activity V1 program has been developed in Fortran and takes advantage of MPI (Message Parsing Interface) to speed up calculation times by allowing the use of multiple processes in parallel. It has a self contained maths library, although this could be improved in the future by using optimised maths libraries for certain functions (e.g. linear algebra).

The code was developed on a Debian based distribution of Linux, but it should be supported on other variants of Linux and Unix, and does not require any specialist hardware.

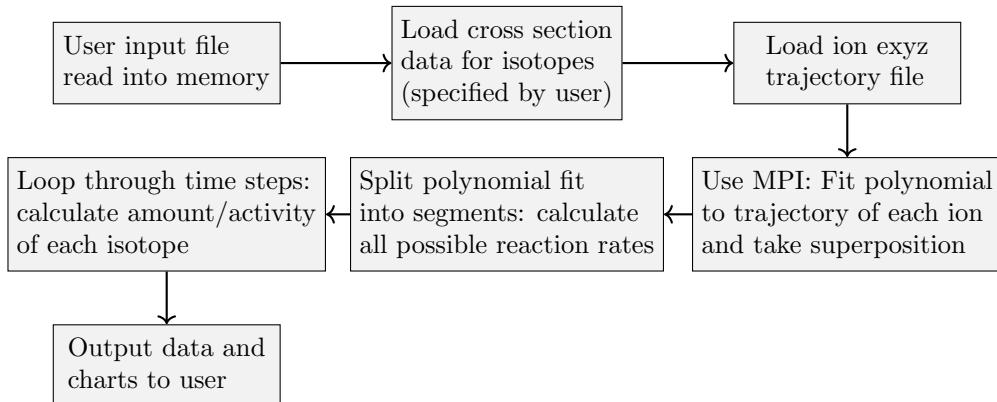


Figure 6.3: Flow chart of major processes in the Activity code

The user is required to prepare an input file that contains the instructions required to perform a calculation. In addition to the input file, the user must provide an EXYZ ion trajectory file output by SRIM. Activity will read in the user input file, and the SRIM and data files listed within, before performing the calculation. Fig. 6.3 shows a flowchart of the major steps the code performs.

There are various settings in the user input file, but the main ones relating to the simulated experiment are:

- Element composition of target (percentage by mass).
- Beam flux (current), energy, duration and area on target.
- Activity measurement time (end of the “experiment”).
- Material density.
- Target thickness.

A brief manual that accompanies the program is in the appendix section ?? and the published paper that accompanies the code is in section ??.

6.4 Ion Trajectory Data

The SRIM code is used to generate ion trajectory data. This uses a quantum mechanical treatment of ion-atom collisions and statistical algorithms to speed up the process[138].

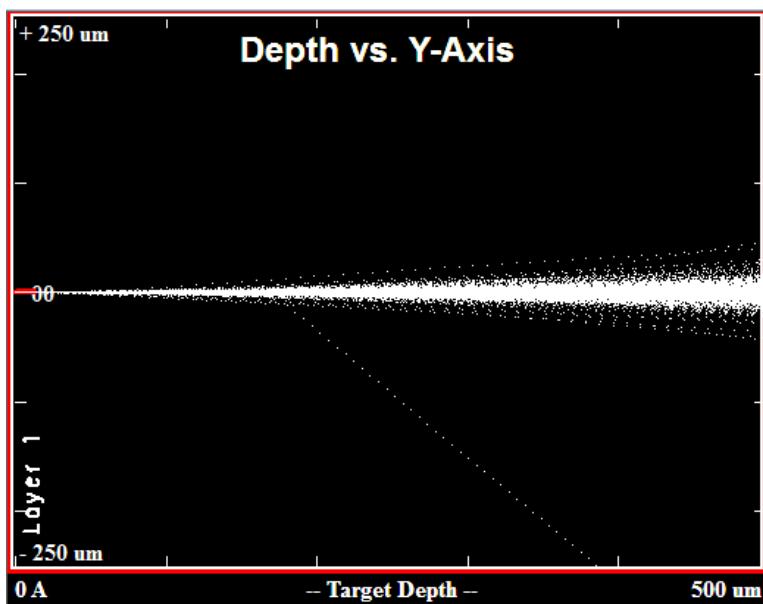


Figure 6.4: 36MeV Ion Track

The trajectory data files used by both versions of the Activity code will be created using SRIM for the required beam energy and target combinations.

The target is set as pure Iron, and due to the way SRIM functions, the structure type is inconsequential. At just 0.5mm in thickness, the 36MeV ions pass through the targets and exit with an energy above 30MeV, and the exyz.txt data file covers this range of ion energies.

6.4.1 Activity V2

A second program was developed to use Python rather than Fortran. One problem that motivated this was occasionally a particular isotope would cause the code to crash, and a second issue was the complication of compiling the code for new users. The Python version runs with relatively few interventions by the user and it handles errors in the data (or lack of) more gracefully. The procedure it follows is similar to the Activity V1 code and the manual is included in appendix section ??.

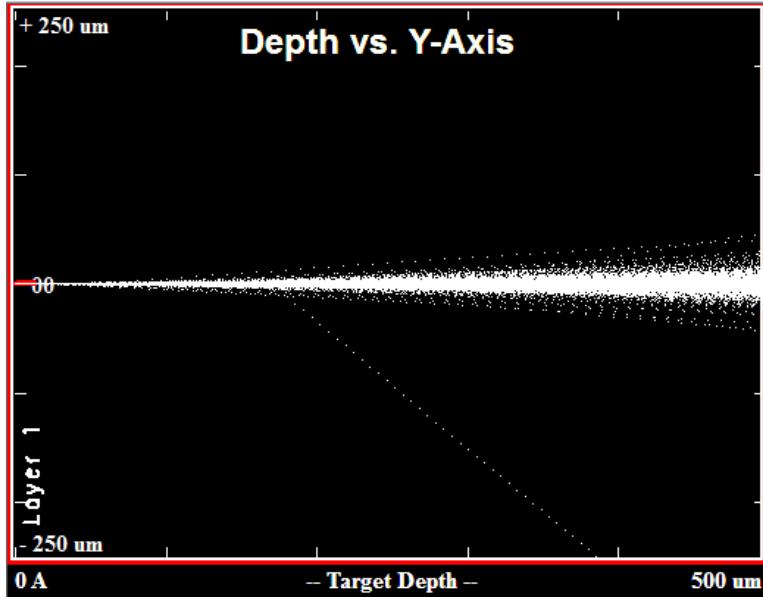


Figure 6.5: 36MeV Ion Track

6.5 Activation by Neutron Irradiation

A short program was created to estimate neutron activity based on the existing code developed to calculate ion activation. It is a simple non-transport neutron activation code, used to estimate the activity and subsequent cooling of materials irradiated by neutrons. A single energy neutron flux or Maxwell-Boltzmann distribution may be selected by the user and the TENDL-2019 data file is used to provide cross sections for calculating reaction rates. Finally, the previously derived extended Bateman equations are used to calculate isotopes at time t after irradiation begins.

6.6 Cyclotron Irradiated Iron

6.6.1 Cyclotron Beam Line

The Scanditronix MC40 cyclotron at the University of Birmingham has several beamlines and is capable of accelerating protons, deuterons, Helium 3 and Helium 4 with fluxes and energy ranges detailed in table 6.1. After running a simulation with SRIM, the 36MeV protons were expected to create an average of 15.8 vacancies each.

Particle	Energy (MeV)	Max Current (micro A)	Flux (ions per second)
p	8-40	60	3.75×10^{14}
d	8-40	30	1.87×10^{14}
$^4He^{2+}$	8-53	30	9.36×10^{13}
$^3He^{2+}$	4-20	60	1.87×10^{14}

Table 6.1: Beam Characteristics of the Scanditronix MC-40

The cyclotron is located in one room, and several beam lines run from the cyclotron to other rooms in the cyclotron building. The operation room is separate from the cyclotron and beam line rooms (fig. 6.6).

The iron sample was held perpendicular to the beam line, with the ions passing through the 0.5mm thickness of the sheet. The beam area was approximately $6.4 \times 10^{-5} m^2$, irradiating a volume of approximately $3.2 \times 10^{-8} m^3$.

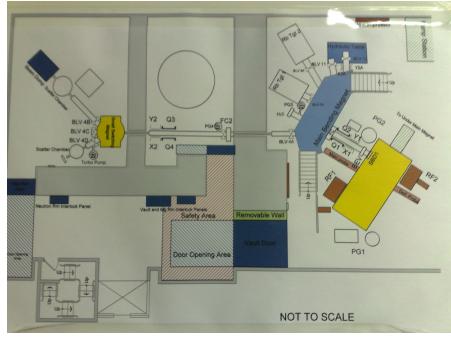


Figure 6.6: Cyclotron layout - main hall and beam lines

The target was irradiated for 300 seconds at 0.5 micro amps and it was expected to cause over 1.4×10^{16} displacements within the volume of iron targetted by the beam. With a number density of approximately 8×10^{28} atoms per cubic meter, giving a relatively low damage dose, when compared to that expected over the lifetime of a component within the reactor, of 5^{-6} DPA.



Figure 6.7: Gamma warning alarm during 0.5 microamp 36MeV proton irradiation of iron

During irradiation, the amount of Gamma radiation released was enough to trip the alarm for the room (fig. 6.7). The proton fluence was less than 1% of the maximum fluence capable of being produced by the cyclotron, so this was definitely a concern. To increase the damage dose, and keep to a shorter period of time, the current would most probably be increased to a much higher percentage, also increasing the rate of Gammas produced during irradiation.

6.6.2 Measurement of Sample Activity

The sample was too radioactive to safely handle immediately after irradiation, so it was left to cool for several days before taking measurements. After it had cooled, a high purity germanium detector (fig. 6.8) was used to measure the activity of the irradiated sample.

The detector and preamplifier are both cooled by liquid nitrogen to 77K, as the band-gap of germanium is 0.7eV and the thermal motion of electrons and nuclei at higher temperatures would induce currents causing noise and interference in the detector. As a gamma ray enters the detector it creates an electron-hole pair in the medium of the detector. The holes and electrons are attracted to the center and to the outer cylinder of the detector, or vice versa.

6.6.3 Prediction of Activity

Following the measurement of the foil target, in the first instance the predicted activity will be estimated using the average ion energy, the cross section of this energy and the two reactions that result in Co-55 ($^{54}_{26}Fe(p,\gamma)^{55}_{27}Co$)



Figure 6.8: High Purity Germanium Detector

and $^{56}_{26}Fe(p,2n)^{55}_{27}Co$). The activity code (version 1 and version 2) are then used to predict the activity of the sample as well as the expected gamma spectra.

Chapter 7

Results: Activity Code

The aim of the first part of this work was to derive an equation capable of calculating the radioactivity of an isotope and all the subsequent isotopes in the decay chain, taking into account branching factors and source terms. This was achieved and a computer code was developed in Fortran to predict the radioactivity of a target irradiated by an ion beam.

The equation has been written into several Python 3 functions and a second code was developed to estimate the radioactivity of a thin foil irradiated by neutrons passing through perpendicular to the plane of the foil.

7.1 Extended Bateman Equations

The Bateman equations were derived from scratch to include a source term for each isotope and branching factors from parent isotope to daughter isotope(s). This process along with the numerical and analytic solutions are outlined in section 6.2.4.

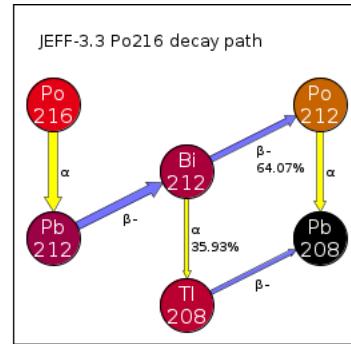


Figure 7.1: Decay path of Polonium-216

The decay equations were programmed as class in python (section `section:decayclass`) that used decay data accessible through a second class, `isotopes`, that loaded data from the JEFF 3.1.1 data file. The static function for computing activity in this class was tested against numerically computed values.

Due to the large range in magnitude of the half life for various isotopes, a decay chain was selected where the time step for the numeric calculation would be applicable to the half life of the isotopes in the decay chain, whilst keeping the number of steps low enough to compute the activities in a reasonable amount of time with a simple algorithm.

Isotope	Half Life (s)
Po-216	1.45×10^{-1}
Pb-212	3.83×10^4
Bi-212	3.63×10^3
Po-212	2.99×10^{-7}
Tl-212	3.1×10^1
Pb-208	Stable

Table 7.1: Half life of isotopes in the Po-216 decay chain

The range of half life in the Polonium decay chain was from 3.83×10^4 s for Pb-212 down to 2.99×10^{-7} s for Po-212 and the maximum number of steps used in the numeric calculation is 10^{10} .

Isotope	Source Rate	N_0
Po-216	2.0×10^{-1}	1.0×10^2
Pb-212	0.0×10^0	5.0×10^0
Bi-212	7.0×10^{-2}	1.5×10^1
Po-212	5.0×10^{-3}	0.0×10^0
Tl-212	0.0×10^0	0.0×10^0
Pb-208	1.0×10^{-2}	3.0×10^2

Table 7.2: Parameters used for the decay equation vs numeric calculation comparison

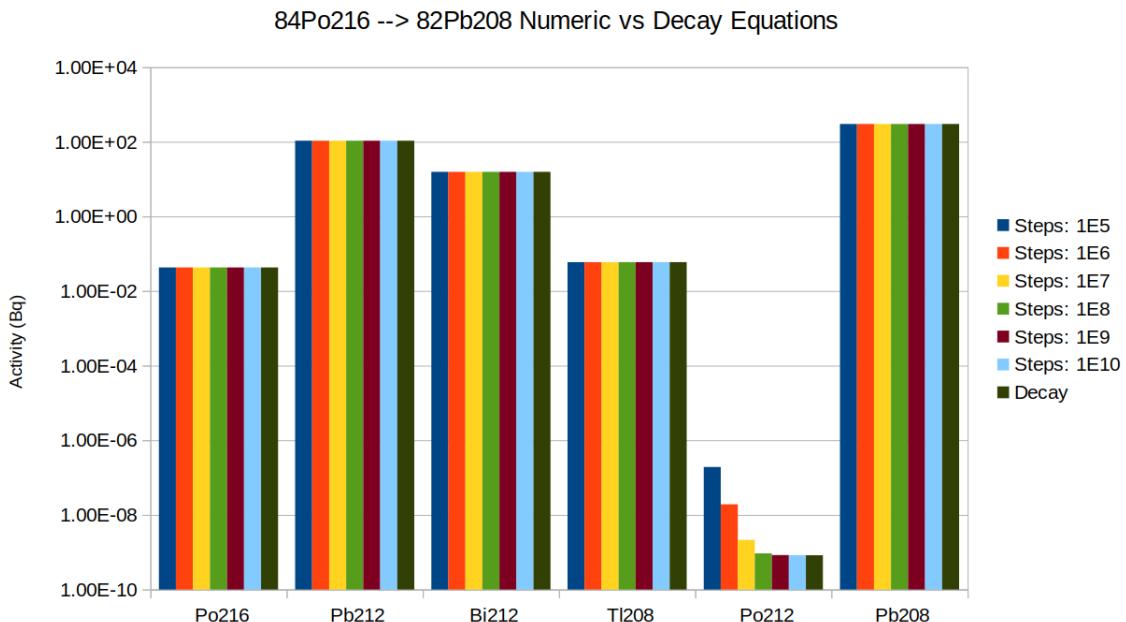


Figure 7.2: Decay path of Polonium-216

The numeric code is detailed in the appendix (section ??). For all but the short half life isotope Po-212, the numeric calculations are in good agreement with the decay equations. As the number of steps increases, and hence the time step decreases, the numerically computed activity for Po-212 also converges to that calculated with the decay equations.

7.2 Activity V1 Computer Package

A large proportion of the results for this part of the work was the development of a computer program to implement the decay equations and calculate reaction rates of isotopes, protons and production rates of resultant isotopes.

The source code and instructions on how to use the program are available to download from GitHub.

Program	Repository
Activity	https://github.com/BenPalmer1983/activity

Table 7.3: Activity V1 Code on GitHub

Activity V1 uses the TENDL-2009 cross section database, as well as several other data files that contain element, isotope and decay data. The user provides an ion trajectory file and an input file that specifies beam parameters and target composition.

7.3 Activity V2 Computer Package

There were a number of issues with version 1 of the computer code. It was written in Fortran, but over time this has been a stumbling block for new users, especially those new to Linux and unfamiliar with Fortran.

In the first version of the code, the trajectory data was replaced with a series of polynomials, but this method was not a reliable solution and it was preferred to use the data directly to calculate the reaction rates. Finally, a language such as Python3 is more widely used and contains very useful features such as dictionaries and a plotting module (matplotlib).

The source code and instructions on how to use the program are available to download from GitHub.

Program	Repository
Activity	https://github.com/BenPalmer1983/activity_v2

Table 7.4: Activity V2 Code on GitHub

Activity V2 uses the TENDL-2019 cross section database for cross section reaction data and the JEFF3.1.1 datafile for decay data. As with version 1, the user provides an ion trajectory file and an input file that specifies beam parameters and target composition.

7.4 Neutron Activation

A small side-project was the development of a simple neutron activation code that uses the TENDL-2019 neutron cross section data file to estimate the activity of a neutron irradiated target. The reason for this was to be able to estimate the activity of components irradiated in a high flux neutron reactor and compare these to the radioactivity of proton irradiated components.

The source code and instructions on how to use the program are available to download from GitHub.

Program	Repository
Neutron Activation	https://github.com/BenPalmer1983/neutron_activation

Table 7.5: Neutron Activation Code on GitHub

7.5 Proton Irradiated Iron

In this work, Iron has been irradiated with a proton beam. Using the TENDL cross section libraries, the Co-55 activity has been estimated and this, along with other expected gammas, have been predicted using the Activity code.

7.5.1 Cobalt-55

Co-55 is a relatively short lived isotope, with a half life of 17.5 hours, that is a concern to health during and shortly after irradiation. Co-55 decays into another radioactive isotope, Fe-55, which then decays into Mn-55 (fig. 7.3). The second step in the decay chain is 2.73 years, so the first step with a much shorter half life will be responsible for the majority of the radiation shortly after being irradiated by protons.

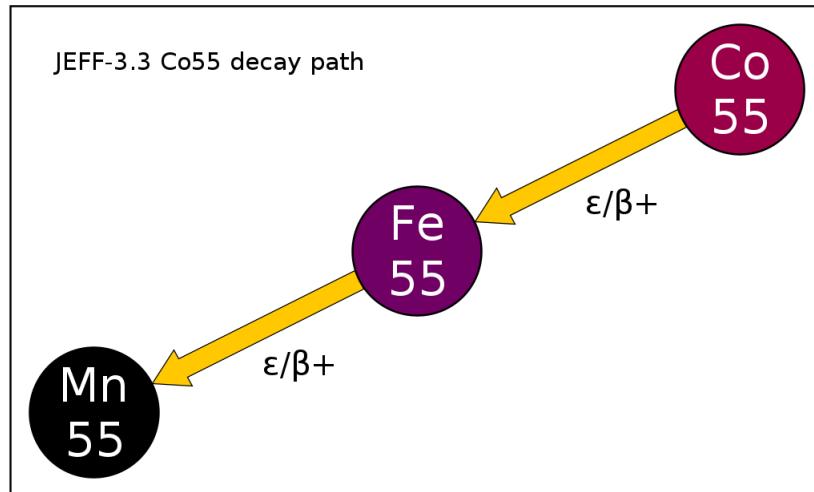


Figure 7.3: Decay path of Cobalt-55

The decay of Co-55 will produce a range of gammas (fig. 7.4), but one in particular to focus on when measuring the radioactivity is the 931KeV gamma with an intensity of 75%.

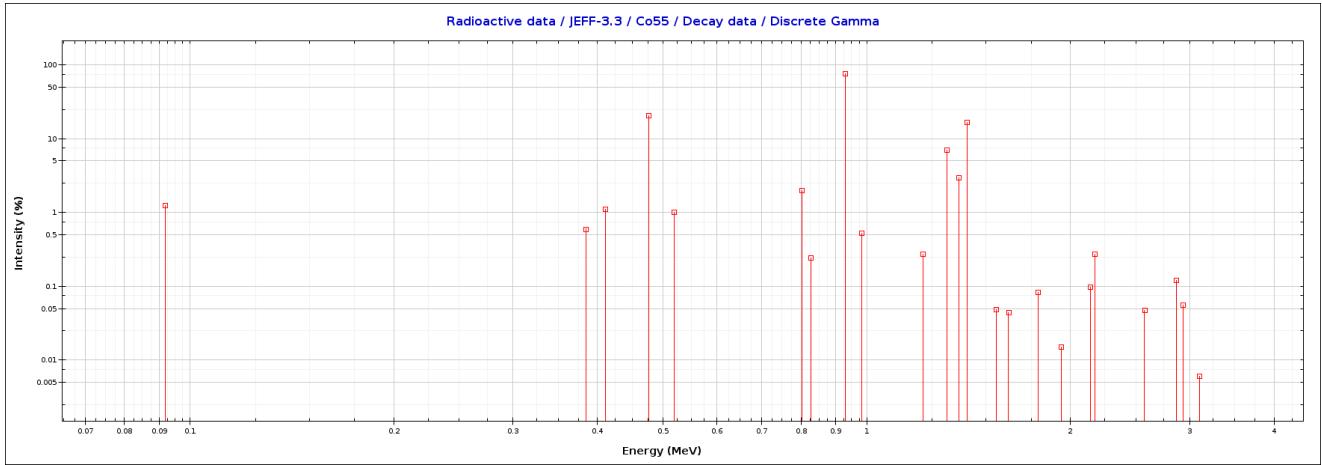


Figure 7.4: Cobalt-55 Gamma Intensity Plot

The stable isotope Iron-54 makes up almost 6% of natural iron, and one reaction possibility with a proton results in Co-55 and a gamma, $^{54}_{26}Fe(p, \gamma)^{55}_{27}Co$. The most abundant isotope in natural Iron also has a route to transmute to Co-55 through the capture of a proton and the loss of two neutrons, $^{56}_{26}Fe(p, 2n)^{55}_{27}Co$.

Due to the short half-life and intensity of the 931KeV gamma, this peak was measured in particular after irradiating the iron target. It was also estimated manually using the TENDL-2009 and TENDL-2019 cross section databases.

7.6 Cyclotron Beam Line - Proton Irradiated Iron

A 0.5mm thick pure iron target was irradiated for five minutes at a current of 0.5 microamps with 36MeV protons. The proton beam formed a $64mm^2$ square that was taken from the cyclotron through one of the beam lines, with the target perpendicular to the beam.

The target was left to cool for several days before being safe enough to handle. The radioactivity was measured using a high purity germanium detector which was calibrated to detect approximately 4 out of every 100 gammas emitted.

-
- 1 RANGE: 617 = 911.51keV to 649 = 958.79keV
 - 2 AREA : Gross = 2127168 Net = 1342565 +/- 2207
 - 3 CENTROID: 631.74 = 933.29keV
 - 4 SHAPE: FWHM = 6.82 FW(1/5)M = 10.06
 - 5 ID: Bi-214 at 934.05keV
 - 6 Corrected Rate = 35349.26 +/- 58.11 cA
-

Listing 7.1: Maestro 931KeV Peak Measurement

The gamma counts were recorded over a range of 0MeV to 3MeV (fig. 7.5). Correcting the measured count for the 931KeV peak, to account for the geometry and detector, the count rate for 931KeV gamma rays from Co-55 was measured at $4.43 \times 10^4 + / - 1.05 \times 10^3$ counts per second.

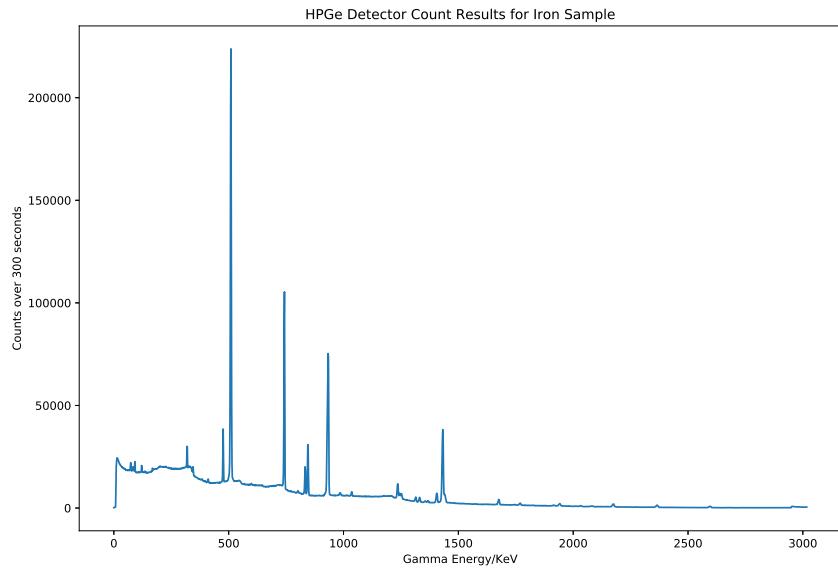


Figure 7.5: Gamma counts from the irradiated iron target over 300 seconds

7.7 SRIM Proton Irradiated Iron

The trajectory data used by the activity code(s), and also to estimate the Co-55 activity, was provided by SRIM. An input configuration was set with Iron as the target material and protons as the projectile ion. The sample depth was set to at least the target thickness, 0.5mm. The activity code truncates the data to fit the target thickness, so a much larger depth, say 1m, could be set in SRIM to ensure all the ions stop within the simulated target, and this data file may be reused for targets of varying thicknesses.

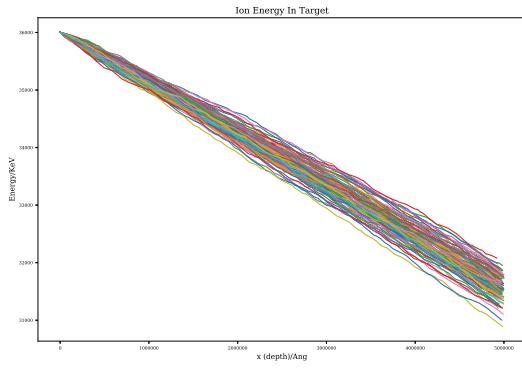


Figure 7.6: Energy vs target depth - protons through a 0.5mm thick iron target

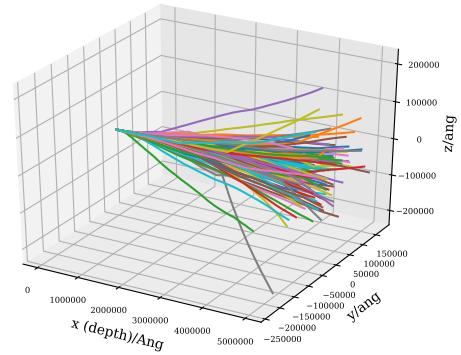


Figure 7.7: Ion trajectory - 0.5mm thick iron target

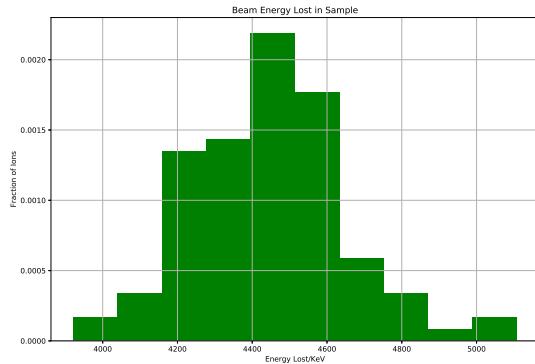


Figure 7.8: Ion energy lost in 0.5mm thick target

The ions travelling through the thin 0.5mm target lose between 4MeV and 5MeV (fig. 7.8) as they pass, and they do so quite smoothly (fig. 7.8). The majority of the energy is lost smoothly through electronic stopping. As the thickness of the iron is increased, all the energy is lost to the iron target, and occurs within a shorter range in the remaining quarter of a millimetre depth of the iron target (fig. 7.10).

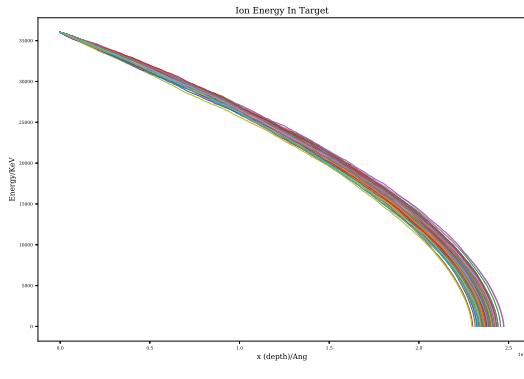


Figure 7.9: Energy vs target depth - protons in iron target (1m thick)

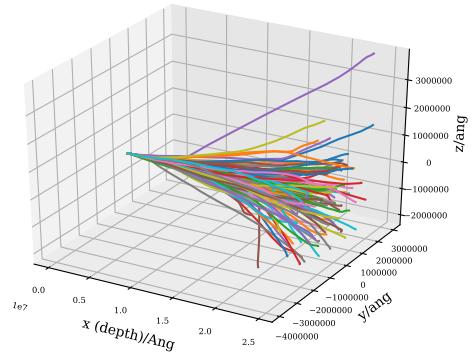


Figure 7.10: Ion trajectory - iron target (1m thick)

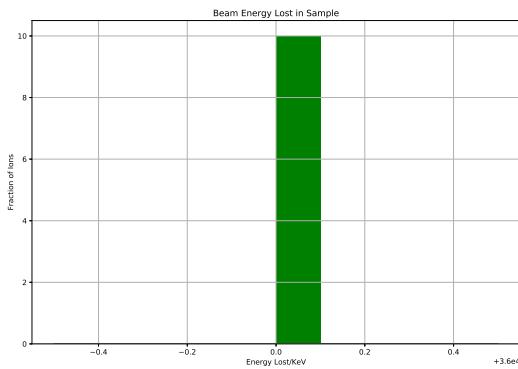


Figure 7.11: Ion energy lost in target (1m thick)

7.8 Estimate of Activity Using TENDL Data

There are over ten TENDL libraries available, and the available data and the range of data points is inconsistent across the different versions. The earlier 2009 library, that was used in the first version of the Activity code, covers a range up to 200MeV for Iron cross sections but the more recent 2019 version, that was used in the second version of the activity code, only covers up to 30MeV for Iron.

The 2019 data file ranges up to 30MeV for the $^{54}_{26}Fe(p, \gamma)^{55}_{27}Co$ and $^{56}_{26}Fe(p, 2n)^{55}_{27}Co$ reactions. As the data cuts off at 30MeV, the cross section data for a 28-30MeV proton is used in the estimation.

Parameter	$^{54}_{26}Fe(p, \gamma)^{55}_{27}Co$	$^{56}_{26}Fe(p, 2n)^{55}_{27}Co$
ND/atoms	4.6×10^{27}	7.4×10^{28}
J/amps	0.5×10^{-7}	0.5×10^{-7}
Q/C	1.60×10^{-19}	1.60×10^{-19}
d/m	5.0×10^{-4}	5.0×10^{-4}
sigma/barns	2.0×10^{-4}	2.0×10^{-2}
irradiation time/s	3.0×10^2	3.0×10^2
Co55 t-half/s	6.3×10^4	6.3×10^4
Co55 λ	1.1×10^{-5}	1.1×10^{-5}
931KeV Gamma Intensity	0.75	0.75
Reaction Rate/atoms per second	1.4×10^5	2.3×10^8
N(Co55 at 300s)	4.3×10^7	6.9×10^{10}
Activity(Co55 at 300s)	4.8×10^2	7.5×10^5
N(Co55 at 3 days)	2.5×10^6	4.0×10^9
Activity(Co55 at 3 days)	2.8×10^1	4.4×10^4
931KeV Gamma	2.1×10^1	3.3×10^4

Table 7.6: Estimation using 28-30MeV protons and the TENDL-2019 database

The 2009 data file ranges from a fraction of an eV to 200MeV for the $^{54}_{26}Fe(p, \gamma)^{55}_{27}Co$ and $^{56}_{26}Fe(p, 2n)^{55}_{27}Co$ reactions, and the average cross section is taken for each reaction based on a 36MeV proton that loses up to 5MeV travelling through the 0.5mm thick Iron target.

Parameter	$^{54}_{26}Fe(p, \gamma)^{55}_{27}Co$	$^{56}_{26}Fe(p, 2n)^{55}_{27}Co$
ND/atoms	4.6×10^{27}	7.4×10^{28}
J/amps	0.5×10^{-7}	0.5×10^{-7}
Q/C	1.60×10^{-19}	1.60×10^{-19}
d/m	5.0×10^{-4}	5.0×10^{-4}
sigma/barns	1.5×10^{-4}	3.0×10^{-2}
irradiation time/s	3.0×10^2	3.0×10^2
Co55 t-half/s	6.3×10^4	6.3×10^4
Co55 λ	1.1×10^{-5}	1.1×10^{-5}
931KeV Gamma Intensity	0.75	0.75
Reaction Rate/atoms per second	1.1×10^5	3.4×10^8
N(Co55 at 300s)	3.3×10^7	1.0×10^{11}
Activity(Co55 at 300s)	3.6×10^2	1.1×10^6
N(Co55 at 3 days)	1.9×10^6	6.0×10^9
Activity(Co55 at 3 days)	2.1×10^1	6.6×10^4
931KeV Gamma	1.6×10^1	4.9×10^4

Table 7.7: Estimation using 36MeV protons and the TENDL-2009 database

The largest contribution to the 931KeV gamma is due to the $^{56}_{26}Fe(p, 2n)^{55}_{27}Co$ reaction, and the overall activity is predicted to be 33,000Bq using 28-30MeV protons with the TENDL-2019 data file (table 7.6), and 49,000Bq using 36MeV protons with the the TENDL-2009 data file (table 7.7).

7.9 Activity V1 - Simulated Proton Irradiated Iron

The activity code with the derived activity equations was used to calculate the predicted radioactivity (fig. 7.12) and predicted gamma lines (fig. 7.13).

Listing 7.2: Activity V1 Results 36MeV Protons and Iron

1 Fe36MeV.in

2 User Input Sim Details:

```

3     Beam Duration/s:          0.3000000000D+03
4     Beam Current/uA:          0.5000000000D+00
5     Beam Energy/MeV:          0.3600000000D+02
6     Beam Area/mm2:          0.1000000000D+03
7     Target Thickness/Angstrom: 0.5000000000D+07
8     Target Density/kgm-3:      0.8000000000D+04
9     Activity Measurement Time/s: 0.2600000000D+06
10
11    Most Active Isotopes      1.0907
12    Symbol   Z     A     M     Activity/Bq
13    CO       27    55    0     0.5942036078E+05
14    MN       25    52    0     0.4510767081E+05
15    CR       24    51    0     0.1444370965E+05
16    FE       26    55    0     0.1280863433E+05
17    MN       25    54    0     0.9738993802E+04
18    Total Activity/Bq:      0.1514219906E+06      1.1291
19    Run time: 1.1292
20
21    Total Activity/Bq:      0.1514219906E+06
22    Total Gamma Power/eV/s: 0.2559334931E+12
23    Total Gamma Power/Watts: 0.4100506435E-07
24    Absorbed Dose*/Grays/s: 0.4078849177E-10
25    Absorbed Dose*/Grays/hr: 0.1468385704E-06
26    Fraction of annual dosage if exposed for 1 hr: 0.1468385634E-03
27
28    Absorbed dose, assumes all energy absorbed, 80kg human, 1m from point-target, 1m surface area exposed to
         irradiation.
29
30    Dose Limits:
31    employees 18+ 20 millisieverts/year
32    trainees 18+ 6 millisieverts/year
33    public and under 18s 1 millisieverts/year
34    public and under 18s millisieverts averaged per hour: 0.1140771128E-03
35    Dose averaged over area of skin not exceeding 1cm2
36    Source: http://www.hse.gov.uk/radiation/ionising/doses/

```

The summary of the calculation is given in listing 7.2 and the full output file and details of all isotopes and their activities are in Appendix ??.

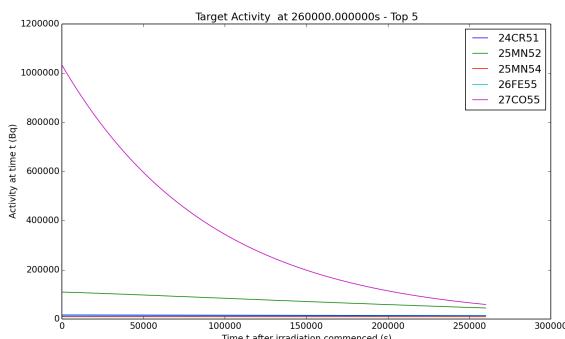


Figure 7.12: Isotopes with the top 5 radioactivity plotted over approximately 3 days

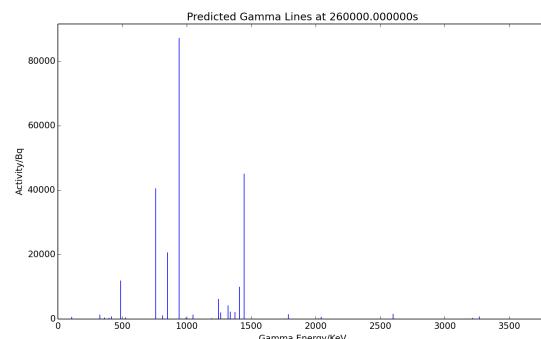


Figure 7.13: Predicted gamma lines approximately 3 days after irradiation

The code predicts a final 931KeV gamma activity of over 59,000 Bq and highlights several other isotopes of interest, including Mn-52, Cr-51, Fe-55 and Mn-54, all of which have an activity of approximately 10,000 Bq or above. After 3 days of cooling, the target is predicted to dose an 80KG human, 1 meter away from the target, with 4.0×10^{-11} Gy/s (absorbed dose) which is less than the limit for a member of the public (6.34×10^{-10} Si/s, effective dose).

7.10 Activity V2 - Simulated Proton Irradiated Iron

Whilst the TENDL-2009 database does extend to several hundred MeV, the latest database does not (at least, for the isotopes that were of interest). This issue will be discussed in more detail in the final two chapters but, briefly, in future developments it would be desirable to give the user the option of using multiple data files as well as generating cross section data with Talys[139] to cover proton energies up to at least 100MeV.

If a calculation was set with a 36MeV beam and trajectory data for 36MeV protons, with the current TENDL-2019 database used by Activity V2, it would return zero activity as the database does not extend high enough. As a result, a lower energy (28MeV) beam was used for the calculation. The lower energy protons do have a higher reaction cross section, and the activities computed are higher than measured and calculated by Activity V1.

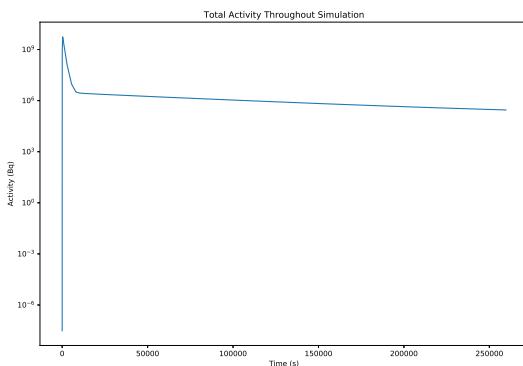


Figure 7.14: Total activity over time of the proton irradiated iron target

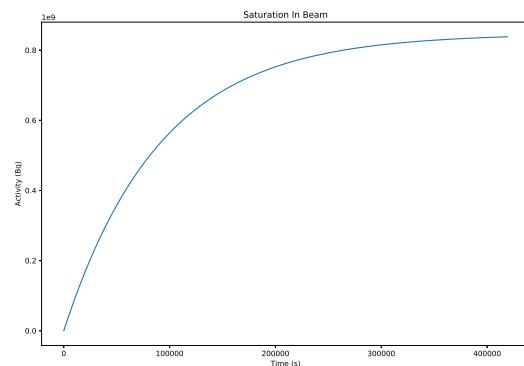


Figure 7.15: Saturation curve for Co55 with the current beam settings

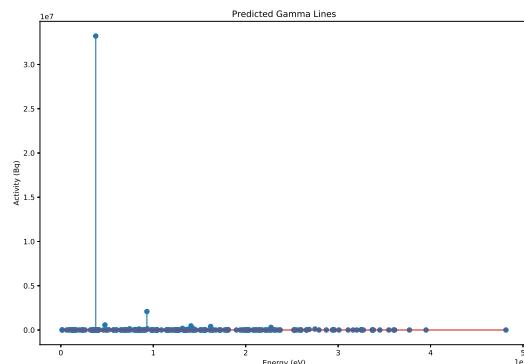


Figure 7.16: Predicted gamma lines at the end of beam time

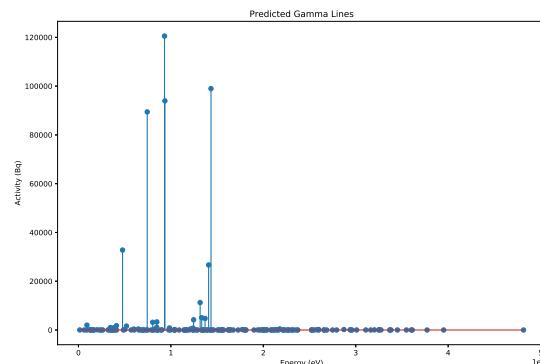


Figure 7.17: Predicted gamma lines at the end of simulation time

Listing 7.3: Activity V2 Results 28MeV Protons and Iron

```

1 3.209E-03 ### Start
2 4.769E-01 ### Loading isotope tendl xs
3 5.289E+00 ### Load complete
4 5.292E+00 ### Load EXYZ.txt
5 5.500E+00 ### Run Simulation
6 5.502E+00 ### Prep sim1
7 1.424E+01 #####
8 1.424E+01 #### Number Density
9 1.424E+01 #####
10 1.425E+01 #### Fe54      5.095579876290546e+27

```

```

11 1.425E+01 ### Fe56      7.713195369931147e+28
12 1.425E+01 ### Fe57      1.750086860505843e+27
13 1.425E+01 ### Fe58      2.2920662292036265e+26
14 2.136E+01 ### ##### End of Beam Tally
15 2.137E+01 ### End of Beam Tally
16 2.137E+01 ### ##### End of Sim Tally
17 2.141E+01 ### 27055    Co55      253567495453.0      2785060.44487
18 3.152E+01 ### ##### End of Sim Tally
19 3.152E+01 ### End of Beam Tally
20 3.153E+01 ### ##### End of Beam Tally
21 3.158E+01 ### 27055    Co55      14632033320.7      160711.045225
22 4.344E+01 ###
23 4.344E+01 ### Gamma Dose - Beam End
24 4.344E+01 ### Activity/Bq      82822873.7128
25 4.345E+01 ### Power eV/s      1.86841008076e+16
26 4.345E+01 ### Power J/s      0.00299352926319
27 4.345E+01 ### Dose Gy/s      2.97685885361e-06
28 4.345E+01 ### Dose Gy/hr     0.010716691873
29 4.345E+01 ### Percentage of annual dose/hr 9394.25237015
30 4.346E+01 ### Gamma Dose - Sim End
31 4.346E+01 ### Activity/Bq      283043.751191
32 4.346E+01 ### Power eV/s      5.1310279645e+14
33 4.346E+01 ### Power J/s      8.22083038417e-05
34 4.347E+01 ### Dose Gy/s      1.96294899335e-07
35 4.347E+01 ### Dose Gy/hr     0.000294301803729
36 4.347E+01 ### Percentage of annual dose/hr 257.984968681

```

Listing 7.4: Activity V1 Results 36MeV Protons and Iron

1	25052	Mn52	1246250000.0	0.0423	98986.0848283	4187.11138824
2	27055	Co55	1369800000.0	0.029499975	160711.045225	4740.97181637
3	25052	Mn52	1333620000.0	0.0507	98986.0848283	5018.5945008
4	27055	Co55	1316500000.0	0.070100025	160711.045225	11265.8482881
5	27055	Co55	1408500000.0	0.16599975	160711.045225	26677.9933297
6	27055	Co55	477200000.0	0.204	160711.045225	32785.053226
7	25052	Mn52	744214000.0	0.9034	98986.0848283	89424.0290339
8	25052	Mn52	935520000.0	0.949	98986.0848283	93937.7945021
9	25052	Mn52	1434050000.0	0.99987	98986.0848283	98973.2166373
10	27055	Co55	931300000.0	0.75	160711.045225	120533.283919

The code predicted a Co55 931KeV gamma activity of 160,000 Bq. The predicted dose is much higher than that predicted by the first activity code, and this may be in part due to a much more comprehensive data file, JEFF 3.1.1, being used in the second version of the code. It is also due to the increased cross section of 28MeV protons with iron, as already mentioned.

The saturation activity is also plotted by the code for Cobalt-55. The experiment duration was just five minutes within the beam and, with the current beam settings, the target would not be fully saturated until almost a week in the beam. This would result in a dangerously radioactive sample.

7.11 Comparing Co-55 Peaks

Whilst there are a number of peaks predicted by the activity codes, in the experimental work the primary (and only) peak calibrated for and measured was the 931KeV peak for Co-55.

The first version of the activity code is in better agreement with the experimental value, but this may be due to the data file used by the second version of the code not extending past 30MeV (table 7.8).

Value source	Co-55 Gammas/second at approximately 3 days
Experimental (36MeV)	$4.43 \times 10^4 + / - 1.05 \times 10^3$
Estimate (28MeV TENDL-2019)	3.28×10^4
Estimate (36MeV TENDL-2009)	4.92×10^4
Activity V1 (36MeV)	5.94×10^4
Activity V2 (28MeV)	1.21×10^5

Table 7.8: Comparison of proton irradiated iron Co55 931KeV gamma rates

7.12 Predicted Activity for 100DPA Irradiated Iron

Generation IV reactors are required to withstand 100-150DPA throughout the lifetime of the plant, which may span 30 years or more. The Scanditronix cyclotron is capable of creating a 60 microamp beam and, due to the small area (and volume of material) this may be concentrated on, it is able to create high damage rates over a shorter period of time.

The Activity V2 code was used to calculate how radioactive a 100DPA iron sample would be after irradiation at the maximum current setting of 60 microamps. The simulated target is 0.5mm thick, pure iron and the beam is protons and has an area of 64mm^2 . The projectile fluence is 3.75×10^{14} protons per second, and the number of atoms within the volume of iron are 2.56×10^{21} . Five different energy settings were used (5MeV, 10MeV, 15MeV, 20MeV, 25MeV) and the vacancies per ion as well as the ion trajectories were generated by SRIM.

	Background	5MeV	10MeV	15MeV	20MeV	25MeV
VPI	-	38.5	58.8	51.0	23.7	20.2
DPA/year	-	178	272	236	109	93
Days for 100 DPA	-	205	134	155	333	391
Damage Depth (mm)	-	Under 0.1	0.25	0.5+	0.5+	0.5+
Dose (Gy/hr)	1.14×10^{-7}	2.64×10^{-3}	2.67×10^{-2}	5.67×10^{-1}	3.26×10^1	1.16×10^2
Dose (Gy/hr) (1 week of cooling)	1.14×10^{-7}	2.04×10^{-3}	2.19×10^{-2}	5.17×10^{-1}	8.15×10^0	2.38×10^1

Table 7.9: Comparison of proton irradiated iron up to 100DPA at a range of proton energies. The background dose is given in sieverts, as the dose equivalent, and is the limit for members of the public per hour, when averaged over a year. The damage depth is the approximate maximum depth ions will reach.

It is clear to see from the predicted activities in table 7.9 that the higher energy beams would take longer to reach 100DPA, as they pass through the target leaving at a high energy, whilst also irradiating the target several orders of magnitude more than 5MeV and 10MeV. Detailed plots resulting from this Activity V2 simulation are in appendix ??.

Chapter 8

Methodology: Interatomic Potential Fitting

In order to perform MD ion damage simulations, an iron-palladium potential is needed. Experimental data is available for FCC palladium, but DFT is needed to compute the properties of FCC iron, as it does not exist in this phase at standard conditions without the addition of an austenite stabilizer. Additional DFT calculations will be used to compute the energy, stress and forces of slightly perturbed atoms which will in turn be used to fit the potentials. A new computer code will be created to fit the potentials.

8.1 Interatomic Potential Fitting

A summary of the process, step by step, towards a new potential has been outlined.

1. Collate experimental values (a_0 , e_{coh} , B_0 , elastic constants etc)
2. Generate or choose pseudopotentials for iron and palladium
3. Converge ecutwfc and ecutrho for these pseudopotentials
4. Converge k-points and smearing
5. Calculate relaxed positions and energies
6. Calculate equation of state where experimental data aren't available
7. Calculate elastic constants where experimental data aren't available
8. Randomise positions of atoms and calculate total energy and the forces between atoms with DFT
9. Insert vacancies, interstitials or both then calculate total energy and the forces between atoms with DFT
10. Select the type of potential to be used (FS, EAM, 2BEAM)
11. Select forms of the potential functions
12. Choose a start point
13. Fit potential

Several computer codes are developed to complete a number of the steps in the list.

8.2 Experimental & DFT Bulk Property Data

The bulk properties for Fe (BCC) and Pd are available on several websites and in published work. The lattice parameter, bulk modulus and elastic constants are available for Fe BCC and Pd FCC, but as pure Fe FCC is theoretical under normal conditions, this will need to be predicted using DFT.

A large amount of computing power is required to run the DFT calculations. If the input parameters are not accurate enough to begin with, it will either take a long time for the DFT calculation to run, or it will fail to converge completely. The lattice parameter for each element, using the density of that pure element in its state under normal conditions, is predicted for that element either for both the FCC and BCC structures (table 8.1).

Element	Atomic Mass	Density kg/m ³	Atoms/m ³	FCC (Bohr/Angstrom)	BCC (Bohr/Angstrom)
Al [140]	26.98	2700	6.02×10^{28}	7.66/4.05	6.08/3.22
Cr [141]	52.00	7140	8.27×10^{28}	6.89/3.64	5.47/2.89
Fe [141]	55.84	7874	8.47×10^{28}	6.83/3.61	5.42/2.87
Ni [142]	58.69	8908	9.09×10^{28}	6.67/3.53	5.30/2.80
Pd [143]	106.42	12023	6.83×10^{28}	7.34/3.88	5.83/3.08

Table 8.1: Predicted lattice parameters based on the density, atomic number and type of structure

Aluminium appears here because it has a simpler electronic structure and no magnetic properties to complicate the DFT calculations. These calculations for Aluminium complete much faster than for the other elements listed, so it was used throughout to develop and test the computer codes created.

8.3 DFT Calculations

8.3.1 Quantum Espresso

There are a choice of several different DFT programs to choose from, including VASP, Siesta and Quantum Espresso. The open source Quantum Espresso includes the PWscf binary that solves electronic structure calculations with plane wave pseudopotentials. It calculates the total energy of a system as well as the forces between atoms and the stress on the simulation box.

As the calculations contain Iron, it is important to take into account the magnetism caused by the filling of the d-shell. PWscf gives three modes: non-magnetic, collinear spin and non-collinear spin.

8.3.2 Pseudopotential Selection

The pseudopotentials were downloaded from the Quantum Espresso PSLibrary with url:

<http://www.quantum-espresso.org/pseudopotentials/pslibrary>

There are several categories of pseudopotential available. The pseudopotentials may be fully relativistic or not. The DFT calculations in this work will be non-spin or collinear spin and the element with the largest number of electrons will be Palladium, with electrons in the s, p, d shells. There will be no elements used in this work with f shell electrons, so a non-relativistic pseudopotentail will be used for each element.

There are a number of choices of exchange-correlation functional, depending on the element:

- pz: Perdew-Zunger (LDA)
- vwn: Vosko-Wilk-Nusair (LDA)

- pbe: Perdew-Burke-Ernzerhof (GGA)
- blyp: Becke-Lee-Yang-Parr (BLYP)
- pw91: Perdew-Wang 91 gradient-corrected

In the literature, LDA and BLYP are used for organic DFT calculations. LDA and GGA type pseudopotentials have been used to model solids and, in particular, the GGA type have been used to study metals as they are more reliable at calculating parameters such as a_0 (see section 5.6.10). Several elements were tested to compare the results of LDA and GGA pseudopotentials to the experimental values for the lattice parameter of each.

The settings used were:

- ecutwfc: 71
- ecutrho: 430
- k-points: 9 9 9 Monhurst-Pack grid offset
- smearing: 0.04 Ry Mazari-Vanderbilt cold smearing
- calculation type: non-polarised calculation
- pseudopotentials: PZ for LDA, PBE for GGA

The results were computed as follows:

Element	Crystal	a_0 exp. (ang)	a_0 LDA (ang)	a_0 GGA (ang)
Al	FCC	4.05	3.98	4.04
Fe	BCC	2.87	2.69	2.75
Pd	FCC	3.89	3.84	3.93
Cu	FCC	3.61	3.49	3.59

Table 8.2: Predicted lattice parameters based on the density, atomic number and type of structure

The simpler LDA pseudopotentials consistently underestimate the lattice parameter a_0 for the four metals tested, and on average they underestimate by 3%. The GGA calculations are within 1.5% of the experimental value.

8.3.3 Smearing Type

The material being studied here is a metal/metal alloy, and this has important consequences when setting up the calculation. The conduction and valence bands overlap in a metal, and the fermi energy passes through this overlap. When the occupied states are summed, integrating over all k-points k[144], those bands that cross the Fermi energy will drop to zero. There are two solutions used here to reduce this abrupt drop off: increase the number of k-points, or add a smearing to smooth the occupation function. Previous work has given the following suggested smearing types and values as a start point.

- Degauss values 0.01Ry Methessel-Paxton [145]
- Fermi-Dirac 0.01eV (0.00074Ry) [146]
- Marzari-Vanderbilt 0.01Ry [147]
- Degauss 0.03 + 0.05 [148]

- Marzari-Vanderbilt 0.05Ry [149]

This suggests that a range between 0.01Ry to 0.05Ry should be investigated as a sensible range of smearing energies. There are a choice of four smearing functions in QuantumEspresso.

- Gaussian
- Fermi-Dirac
- Methfessel-Paxton
- Marzari-Vanderbilt

Functions such as the Gaussian and Fermi-Dirac are not cold smearing, and will converge to the wrong energy whereas the Mazari-Vanderbilt and Methfessel-Paxton are designed to reduce and heating whilst smearing the electron wavefunction. An author of QuantumEspresso, N. Marzari, co-developed the cold smearing Marzari-Vanderbilt function, and this was selected as it had appeared a number of times in the literature and throughout the QuantumEspresso tutorials.

8.4 Parameter Convergence

The energy cutoff, charge density, k-point and smearing values must be selected carefully for any similar DFT calculation. There is a balancing act between the accuracy of the result of a particular calculation and computational time. This will also vary with the pseudopotential, complexity of the electron configuration of the atoms, whether or not the calculation is with or without spin, and so on.

The pseudopotentials and smearing type have already been selected, and the first set of calculations will not include magnetism. The parameters must work well with high symmetry geometries, such as a BCC or FCC crystal with the atoms in their exact position, but also where the atoms have been slightly perturbed from their exact position in the lattice.

8.4.1 Ecutwfc and Ecutrho

A primitive FCC or BCC is created and the atoms are slightly perturbed, breaking symmetry but also putting forces on each atom within the cell. A reasonably high number of k-points are used and the smearing value should be low, but the k-point values in particular will depend upon the computing resources you have available.

Pick a low ecutwfc and set the ecutrho equal to four times the value of the ecutwfc. Depending on the pseudopotential, the calculation may fail because the values are too small. If this is the case, continue increasing the value in small (perhaps 5Ry) increments until the calculation runs successfully.

Next, continue to increase the ecutwfc value whilst increasing the value of ecutrho such that $ecutrho = 4 \times ecutwfc$ in steady increments (5Ry). Once the calculated force and energy per 1Ry increment fall below a threshold set by the user, the values have converged. In this work, the convergence values were:

- Energy convergence per 1Ry: 1.0×10^{-6} Ry (1.36×10^{-5} eV)
- Force convergence per 1Ry: 1.0×10^{-5} Ry/Bohr (2.57×10^{-4} eV/Ang)

At this point, it may be possible to reduce the ecutrho value further while remaining within this convergence threshold, so the ecutwfc value is held fixed while the ecutrho value is reduced in steps of 5Ry.

Finally, attempt to reduce the value of ecutwfc further in 1Ry steps while holding the ecutrho value steady for as long as the convergence value remains within the set threshold.

8.4.2 K-Points and Smearing Value

A configuration with a similar size and number of atoms as those in the calculations should be created. In this work, the reference database and bulk properties will be calculated using 32 atom configurations in an approximately sized $7 \times 7 \times 7$ angstrom box. The Iron FCC k-points and smearing are calculated using a 32 atom FCC box with sides length 7.2 angstrom and the Palladium with a 32 atom FCC box with sides length 7.8 angstrom.

As with the ecutwfc and ecutrho convergence calculations, the atoms are slightly perturbed from their exact crystal positions, putting a force on the atoms and breaking the symmetry (as will be the case for the configurations used for fitting).

The convergence process is more complex for k-points and smearing. There might be a convergence that, at face value, looks better (a smaller convergence value and a decrease in the computation time and memory requirements) for combinations with a higher smearing value, but it is preferred to have a smaller smearing value with a higher number of k-points, while maintaining a reasonable computing time, as the greater the smearing the further the calculated value is from the actual calculated value (with no smearing and a very high number of k-points).

A 2D plot of energy and force values as a function of number of k-points and smearing may then be used to choose reasonable settings that balance the requirements.

8.4.3 Automating Convergence

The process requires the creation of many input files, and manually editing these files is time consuming and vulnerable to human error. Over 100 input files may be required to converge the ecutwfc and ecutrho values, with a further 50-100 required to generate the 2D k-point and smearing plots.

A python code, QECONVERGE, is developed to automate this process and plot the results automatically for the user (section 9.2).

8.5 Bulk Property Calculations

8.5.1 QEEOS Python Program

A Python code, QEEOS, (section ??) has been developed to calculate bulk properties of a structure using the DFT code Quantum Espresso. It is able to fit the equation of state for a cubic structure, and elastic constants for orthorhombic structures (which includes the subset of cubic structures). In brief, the user provides a template PWscf input file, and the program adjusts the crystal basis vectors, applying the homogeneous strain and 9 distortions required to calculate the equation of state and elastic crystals, as discussed earlier in this work (section 5.2.6). The output data and highlights of this code are detailed in the results (section ??) and appendix (??).

8.5.2 DFT Calculations: Austenitic Iron Properties

Experimental data was available for the bulk properties of pure BCC iron and pure FCC palladium. The properties of theoretical (under normal conditions) FCC iron are calculated using the QEEOS code with Quantum Espresso.

8.6 Reference Database

A reference database with a range of FCC configurations of iron, palladium and fe-pd alloy is needed with atom positions slightly perturbed from their perfect crystal positions. A computer code, QEFORFIT, is developed to automate this process (section ??).

8.7 DFT Errors, Convergence Failures and Repeat Calculations

8.7.1 Adjustment of DFT Parameters

Throughout the DFT segment of this work, there were a number of issues that either resulted in the calculations failing or completing and not converging. There were a number of causes for this and several solutions were used throughout the process to converge the SCF calculation.

Where the `ecutwfc` and `ecutrho` parameters are not converged there is a danger that values too low therefore containing too few plane waves in the basis set. This might cause problems with the SCF causing it not to converge. The same can be said about a failure to pick appropriate k-points and smearing.

There are a selection of mixing modes available in PWscf, giving a choice of Thomas-Fermi screening for homogeneous systems, local density dependent Thomas Fermi screening for inhomogeneous screening and Broyden mixing as the default. In this work, most success was found with plain Broyden mixing and a smaller than the default mixing factor (`mixing_beta = 0.1` rather than the default of 0.3).

8.7.2 A Lack of Memory

The DFT calculations are performed on a supercomputer with many processors and many cores per processor. The program uses two implementations of parallelization.

- OpenMP - many threads have access to the same shared memory
- OpenMPI/MPICH - many processes, each with its own memory

Whilst some favourable results have been shown using a hybrid of OpenMP and OpenMPI, it was more straightforward to use in OpenMPI mode only in this work. Unfortunately, the amount of memory per process overwhelmed the computing nodes for certain calculation. The job script was modified to reserve entire nodes and all the memory on each node, but to only use the cores on one of the two processors to halve the processes per node, but double the memory per process. It was quicker to do this than wait in the high memory queue on BlueBEAR, where the memory per process would be perhaps double than that required.

8.7.3 Scratch Restrictions

When the calculations first started, the scratch space was located on a large shared space on the BlueBEAR computer. Over time the drive space quota was restricted and the scratch was relocated to individual nodes. There were few issues with less complex calculations as the SCF temporary data was several GB in size with the scratch space on each node being approximately 120GB. Once a batch of collinear iron calculations had been submitted to the job queue, they would quickly fill the scratch drive and the job would terminate after just a few calculations, as the node had run out of scratch drive space.

Listing 8.1: Sbatch file

```
1 #!/bin/bash
2 #
3 #SBATCH --job-name=feforfit
4 #SBATCH --output=jobout.txt
5 #SBATCH --account=readmsd02
6 #
7 #SBATCH --ntasks 40
8 #SBATCH --nodes 1
9 #SBATCH --time 1800:00
10 #SBATCH --mem 180GB
11 #
12 #SBATCH --get-user-env
13 #SBATCH --export=NONE
14 #
15 unset SLURM_EXPORT_ENV
16
17 module purge; module load bluebear
18 module load bear-apps/2018a
19 module load iomkl/2018a
20 module load Python/3.6.3-ionmk1-2018a
21 module load matplotlib/2.1.1-ionmk1-2018a-Python-3.6.3
22
23 # Change to $PBS_O_WORKDIR
24 cd "$PBS_O_WORKDIR"
25 # Set the number of threads to 1
26 export OMP_NUM_THREADS=1
27 export PROC_COUNT=40
28 export PWSCF_SCRATCH=/scratch/bxp912
29 export PWSCF_PP=/rds/homes/b/bxp912/pp
30 export PWSCF_CACHE=/rds/homes/b/bxp912/pwscf_cache
31 export PWSCF_BIN=/rds/homes/b/bxp912/apps/qe-6.3/bin/pw.x
32 export PWSCF_SCRIPT=/rds/homes/b/bxp912/apps/qe-6.3/bin/pw.sh
33
34 python /rds/homes/b/bxp912/apps/python/qeforfit.py input.in > results.txt
```

Listing 8.2: Wrapper file

```
1 #!/bin/bash
2 rm -R $PWSCF_SCRATCH/*
3 mpirun -n $1 $PWSCF_BIN -in $2 > $3
```

As the convergence and equation of state/elastic constant python programs automatically created and submitted the PWscf jobs, they would need to be continually restarted after clearing out the scratch. To work around this a batch script (listing 8.1) and wrapper script (listing 8.2) for pw.sh were written. The batch script sets the environment variables required for the sbatch job scheduling program and the wrapper script, and the wrapper script then clears the scratch directory before running PWscf.

8.7.4 Caching Repeat Calculations

Several of the programs required many hundreds of DFT calculations. At first, if an error was encountered and the program had to start from scratch, computation time would be wasted. To solve this, a directory was set to cache input and output files for PWscf. A hash would be generated from the input file and this would be used as the cache file name. If a calculation was successful, it would save the input and output file. In future, if the same calculation was submitted again, the cached file would be returned. The cache could be copied from BlueBEAR to other computers to save on repeat submission to the BlueBEAR job scheduler.

8.7.5 Not Straying Too Far From Reality

It is important to provide the calculation with reasonable and realistic configurations and settings. Where a pure element such as Aluminium, Iron or Platinum was being investigated, the known experimental lattice parameters were used as a starting point (table 8.1 and table 8.2). If the parameters were too far from these values, for example inputting a lattice parameter of either 2 or 8 angs for Aluminium rather than 4, as a starting approximation, the DFT calculation might struggle to relax the structure, or to even successfully run or converge a calculation.

The FCC Iron structure lattice parameter was estimated from the density of FCC steel, and the lattice parameter for alloys could similarly be estimated within a reasonable margin to use as a start point for the calculation.

8.8 EAMPA: Potential Analysis and Fitting Code

A python-fortran based computer code was developed to fit interatomic potentials to bulk properties and DFT calculated forces and energies. It has been designed to take advantage of both Fortran and Python. Fortran is used to compute the neighbour lists, energies, stresses and forces, and Python is used to read and write data, produce plots and control the potential fitting. This allows the use of custom optimization subroutines as well as the use of those included in the SciPy module. Highlighted code is discussed in the results (section ??) and the appendix of this work (appendix ??).

8.9 Iron-Palladium Potentials

The experimental data and DFT computed bulk properties was used along with the database of reference configurations and the EAMPA potential fitting code to produce the potentials. Following this, the EAMPA code was then be used to check the bulk properties that the potentials give, as well as the cohesive energy and surface energy plots.

8.10 DL_POLY Contribution

DL_POLY is a Molecular Dynamics code developed by W. Smith, T.R. Forester and I.T. Todorov at Daresbury Laboratory in Warrington. It is written in Fortran and, before the modifications, included a number of potential types for metals including:

- Finnis Sinclair
- EAM
- EEAM

The Finnis Sinclair is a particular form of the EAM potential, and EEAM is a modification where, if the metal is an alloy, the density and embedding functional for each atom type are treated separately.

A meeting was held with Dr. Todorov at Daresbury Laboratory, where a brief overview of the relevant code was given. After this point, and corresponding by email, the two band EAM type was added (2BEAM). The code was altered with the addition of an array used to store the d-band density function and d-band embedding functional, with the s-band function/functional using the original density function and embedding functional arrays. The calculation of energy and forces on atoms was also altered to make use of these arrays where a two band embedded-atom method potential is used.

Listing 8.3: DL_POLY 4.05 mailshot extract

```
1 DL_POLY_4.05: New Release & Events - MAILSHOT 013
2
3 NEW FEATURES \& IMPROVEMENTS
4 -----
5 1. New two band (2B) EAM and EEAM potentials for metals (TEST45 and TEST46).
6
7 Acknowledgements
8 -----
9 Ben Palmer \@ University of Birmingham (UK) for contributing to the
10 development and testing of the 2BEAM for metals;
11
12 Regards,
13
14 Ilian Todorov
15 July 2013
```

Chapter 9

Results: DFT Reference Database

The key parameters required for Quantum Espresso are converged and selected in preparation for the upcoming calculations. A set of atom configurations for FCC crystals are created and DFT is used to computer the energy, forces and stresses on these crystals. FCC iron does not exist at normal room temperature and pressure, without the addition of a austenite stabiliser. A set of DFT calculations are used to compute the bulk modulus, lattice parameter and cohesive energy of pure Fe FCC. This data will be used to derive an Fe-Pd potential.

9.1 Introduction

While computer technology has advanced almost in accordance with Moore's law, the calculations involved to approximately simulate simple structures with Quantum Mechanics are barely accessible. Ideally, crystal structures with several hundred atoms, containing Fe, Cr, Ni and Pd would have been used to derive a potential describing all four elements.

The computer used did not have the resources required for such calculations, and so the model was simplified to Fe-Pd. The ferromagentism and antiferromagnetism of Fe and Cr were explored using collinear spin DFT calculations.

9.2 QECONVERGE Python Code

9.2.1 Purpose of Code

The process of converging cutoff parameters is one that may be automated. A program was developed in Python to automatically converge the ecutwfc and ecutrho values within a specified threshold.

The program reads an input file into memory, and this contains the settings for the convergence run. A template PWscf input file is also loaded, and this will have any required PWscf settings, such as the pseudopotentials, atom species and so on.

Once these files have been loaded, the first stage of the program runs to determine the ecutwfc and ecutrho values that are within the force and energy convergence thresholds provided by the user. The program creates a crystal based on the settings, for example an FCC, BCC or SC crystal supercell. The exact atomic positions are likely to be the optimal, relaxed, positions and so the overall forces on each atom will be zero. The program randomly varies the atomic positions, and this results in a configuration with forces between the atoms.

The wfc value starts with the user defined ecutwfc starting value, and this is increased until the change in energy and force both converge to within the specified thresholds. The ecutrho value is increased simultaneously, and is four times the ecutwfc value. In the second stage, the ecutrho value is decreased and the ecutwfc value is held constant while the energy and force remain within their respective convergence thresholds. Finally, the program attempts to decrease the ecutwfc value further.

Following the energy and charge density cutoff, the program then runs a number of calculations in order to produce a collection of colour density plots of energy cutoff vs density cutoff, plotting the convergence of total energy and total force. This allows the user to visualise the convergence surface and select cutoff values manually, if desired.

The final part of the program is to help the user decide upon degauss and k-point values. The converged ecutwfc and ecutrho values from the first part of the code are used, and the k-point start, end and increment amounts are set by the user, as is a list of smearing degauss values. 2D colour plots of force and energy convergence as smearing changes and k-point value changes are prepared and saved so the user may study these to determine the best combination for their application.

The randomised atom positions use a seed that may be set in the input file. This means that, although they are pseudo-rng generated, they are repeatable given the same random seed. Every successful PWscf calculation is logged and saved. If the program runs multiple times, the cached output files are used rather than recalculating every input file.

9.2.2 Source Code and Instructions

The source code and instructions on how to use the program are available to download from GitHub.

<https://github.com/BenPalmer1983/qeconverge>

9.3 Convergence of Key DFT Settings and Testing

9.3.1 Pseudopotential

Prior to generating a database of atom configurations, calculated forces, stresses and energies, key DFT settings needed to be selected and tested. The Python code QECONVERGE was written then used to converge the ecutwfc, ecutrho, k-points and degauss smearing values for each of the pseudopotentials required.

As discussed in previously (section 5.6.10) the GGA type pseudo-potential more reliably calculated the bulk properties of a material when compared to LDA/LSDA, and a copy of the earlier table is given in table 9.1.

Pseudo-potential	Element	a/angs	B_0/GPa
Experimental	Na	4.29[123]	6.3[123]
Na.pz-spn-kjpaw_psl.1.0.0	Na	4.06	8.72
Na.pbe-spn-kjpaw_psl.1.0.0	Na	4.20	7.67
Na.pbesol-spn-kjpaw_psl.1.0.0	Na	4.17	7.50
Experimental	Al	4.05[124]	76[124]
Al.pz-nl-kjpaw_psl.1.0.0	Al	3.98	78.6
Al.pz-n-kjpaw_psl.1.0.0	Al	3.98	78.6
Al.pbe-nl-kjpaw_psl.1.0.0	Al	4.04	91.3
Al.pbe-n-kjpaw_psl.1.0.0	Al	4.04	75.0
Al.pbesol-nl-kjpaw_psl.1.0.0	Al	4.01	87.7
Al.pbesol-n-kjpaw_psl.1.0.0	Al	4.01	79.0

Table 9.1: Experimental vs LSDA vs GGA - the DFT values were computed with a PWscf[125] using a 2x2x2 cell, 7x7x7 kpoints and ecutwfc=50

The following pseudo-potential files were used for each element:

- Aluminium: a plane wave GGA potential Pd.pbe-spn-kjpaw_psl.1.0.0.UPF
- Chromium: a plane wave GGA potential Cr.pbe-spn-kjpaw_psl.1.0.0.UPF
- Iron: a plane wave GGA potential Fe.pbe-spn-kjpaw_psl.1.0.0.UPF
- Palladium: a plane wave GGA potential Pd.pbe-spn-kjpaw_psl.1.0.0.UPF

For each pseudo-potential, the ecutwfc, ecutrho and kpoint and degauss settings were converged. Ideally, the k-points settings would have been increased, but doing so would have been prohibitive due to the amount of time taken to perform each calculation.

By the time calculations were being executed for a 32 atom supercell, one node of 20 processor cores would run for several days up to a week or more, requiring the calculation to be submitted to the job queue multiple times. Al is also considered in the results section, as it is a simpler atom to run trial calculations with, and to compare to experimental results.

9.3.2 Collinear Spin Calculations

The relaxed crystal lattice parameters were calculated for Fe, Cr and Pd. Cr will not be used in the Fe-Pd fitting calculations. However, there is a large increase in computation time between non-spin and collinear spin calculations. The ferromagnetic structure of BCC Fe, antiferromagnetic structure of FCC Fe and BCC Cr were investigated.

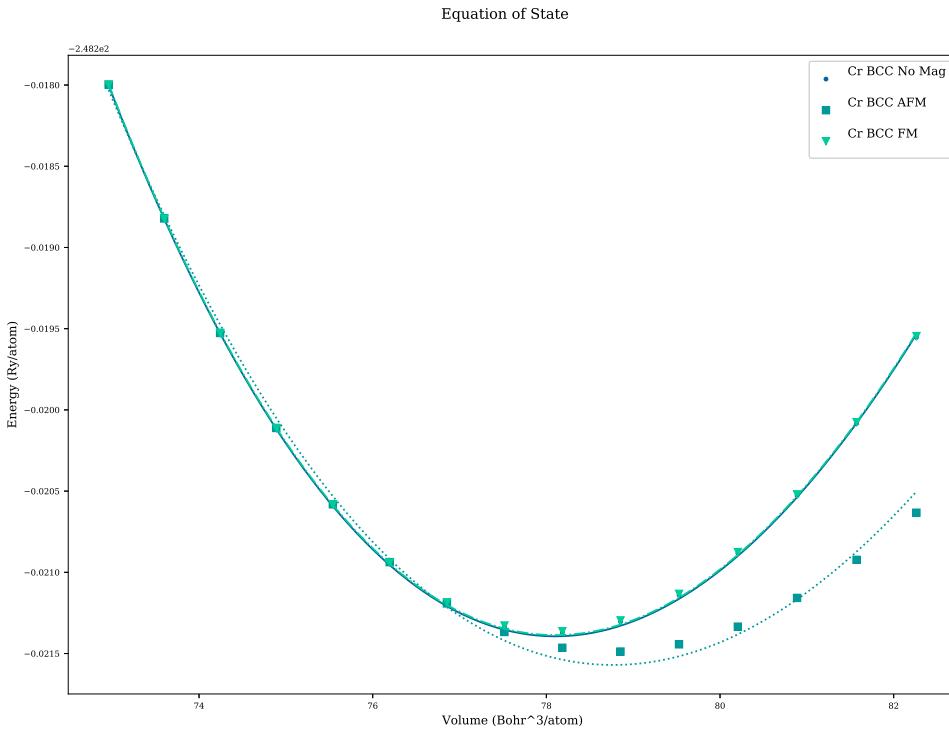


Figure 9.1: Equation of state fit through data points for Chromium BCC with no magnetism, ferromagnetic and anti-ferromagnetic configurations

Description	A_0 (ang)	V_0 (bohr ³)	B_0 (GPA)	E_0 (Ry) (DFT Only)
Exp.	2.91	83.23	160	-
AFM	2.86	78.77	217	-248.2216
FM	2.85	78.10	267	-248.2214
No Mag	2.85	78.102	267	-248.2214

Table 9.2: Chromium properties with and without collinear spin

The equation of state was calculated for Chromium BCC in three ways: (1) with magnetism switched off, (2) atoms set in a ferromagnetic configuration (collinear, all in the same direction), (3) atoms set in an antiferromagnetic configuration (collinear, atoms in the same cell with spin in opposing directions).

The DFT calculated values for the bulk modulus were larger than expected, but the antiferromagnetic calculation was much closer to the experimental value (table 9.2). The E_0 values do not reflect the actual energies, but show the relative calculated values. The antiferromagnetic configuration gives the lowest, optimum, energy (fig. 9.1).

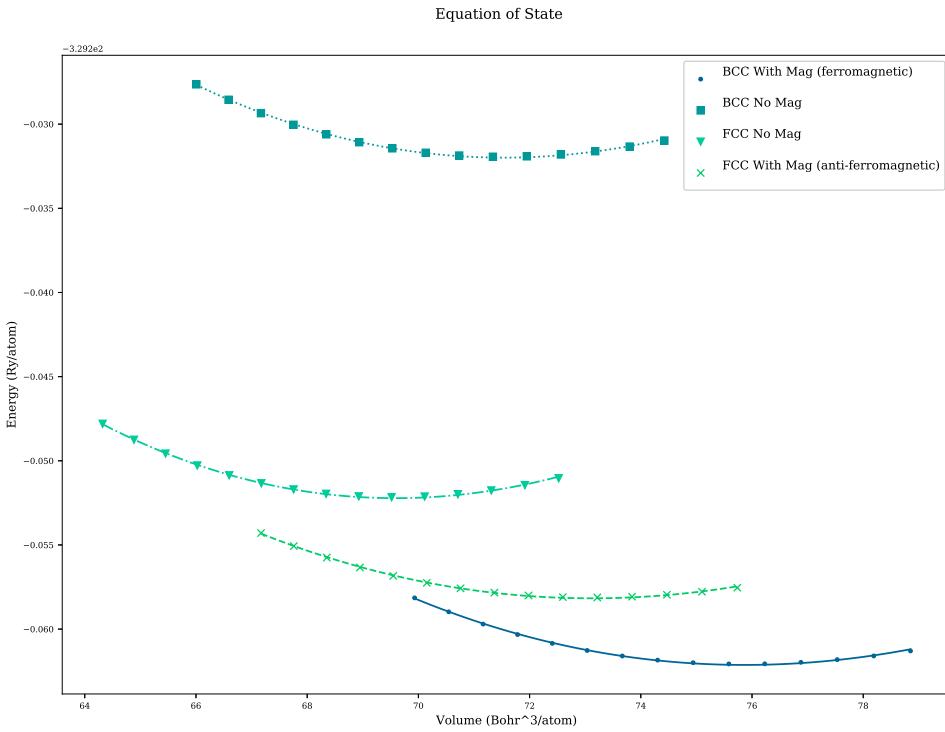


Figure 9.2: Equation of state fit through data points for Iron BCC (no magnetism, ferromagnetic) and Iron FCC (no magnetism and antiferromagnetic)

Description	A0 (ang)	V0 (bohr ³)	B0 (GPA)	E0 (Ry) (DFT Only)
Exp.	2.86	79.01	170	-
No Mag	2.77	71.53	286	-329.23
FM	2.82	75.9	239	-329.26
AFM	(failed)	(failed)	(failed)	(failed)

Table 9.3: BCC Iron properties with and without collinear spin

In the case of Iron, both BCC and FCC configurations were used. The antiferromagnetic configurations for both BCC and FCC failed in that the SCF calculations failed to converge. As expected, the optimal configuration was BCC with ferromagnetism, and second to this was FCC with ferromagnetism.

Despite the increase in computational time, and other demands on resources such as scratch space and RAM per node, it was clear that collinear spin calculations were required for iron. The lattice parameter value was within 2% of the experimental value with the BCC structure set in a ferromagnetic state. Whilst the bulk modulus is only within 41% of the experimental value, this is an improvement over the non magnetic calculation that was almost 70% away from the experimental bulk modulus (table 9.3, fig. 9.2).

9.3.3 Convergence Results

The QECONVERGE program was used to suggest the parameters to use, given the convergence threshold of $1.0^{-5} RY/Bohr$ for forces and $1.0^{-6} RY$ for energy.

Element	Pseudopotential	Ecutwfc	Ecutrho	K-points	Degauss
Al	Al PBE KJPAW	50	200	11	0.04
Fe	FE PBE KJPAW	71	431	9	0.04
Pd	PD PBE KJPAW	71	431	9	0.04

Table 9.4: DFT Settings - pseudo-potentials, ecutwfc, ecutrho, k-points, smearing

These settings in table 9.4 were used throughout the remainder of the work. The maximum values were selected from Fe and Pd as they were to be used for calculations of only Fe, only Pd and an alloy of the two.

The other settings used in PWscf throughout are listed in table 9.5. Several of these values were arrived at by trial and error. It was recommended by the authors of Quantum Espresso to adjust parameters such as the mixing beta if a calculation fails to converge. The mixing mode was also changed from time to time, but overall the plain mode seemed to work best in most cases.

Parameter	Value
etot_conv_thr	0.0001
forc_conv_thr	0.001
conv_thr	1.0D-6
diagonalization	david
mixing_beta	0.1
mixing_mode	plain

Table 9.5: DFT Settings - other settings

The choices of parameters are supported by the convergence plots produced by the QECONVERGE code in figures ?? and ?? (appendix ??).

9.4 Preliminary Calculations

9.4.1 Relaxed Crystal Calculations

The cohesive energy is an important value in relation to the interatomic potential. The energies calculated by DFT code depend on many factors including the energy under which plane-wave are cut-off, the degauss value, k-point settings and the SCF convergence parameters. What is more important in the DFT calculations is the difference in energy between calculations. As the cohesive energy is known, a DFT calculation may be performed for each species of atom to determine the relaxed energy. This value may then be used to calibrate other DFT calculations, so they are given relative to the energy of each atom spaced infinitely far from one another, with a binding energy of 0eV. The relaxed energies, the settings used to calculate them and the adjustment to convert them to the “real” energies are given in table ??.

Measurement	Al FCC	Fe BCC	Fe FCC	Pd FCC	Ru HCP
Element	Aluminum	Iron	Palladium	Ruthenium	Ruthenium
Structure	FCC	FCC	FCC	HCP	FCC
Ecutwfc (Ry)	50	71	71	71	71
Ecutrho	200	430	430	430	430
K-points	11 11 11	9 9 9	9 9 9	9 9 9	9 9 9
Smearing (Ry)	0.04	0.04	0.04	0.04	0.04
NSpin	0	2	2	0	0
Starting Magnetism	None	FM	AFM	None	None
No. Atoms	32	16	32	32	32
Energy (Ry)	-1264.08749654	-5268.18846365	-10536.25040753	-16384.95803030	-6906.78722345
Energy/Atom (Ry)	-39.502734267	-329.261778978	-329.257825235	-512.0299338447	-431.674201466
Energy/Atom (eV)	-537.462275214	-4479.836349416	-4479.78255982	-6966.524743211	-431.665652533
Known Cohesive Energy (eV)	-3.36	-4.316	-4.26 (Calculated)	-3.91	-6.74
Adjustment/atom (eV)	534.102275214	4475.520349416	4475.520349416	6962.614743211	5866.47338
Relaxed a_0 (Bohr)	15.265	10.594	12.937	14.845	-6.62
Relaxed c (Bohr)	None	None	None	None	None
U_{xx}	1.000	1.000	1.000	1.000	1.000
U_{yy}	1.000	1.000	1.054	1.000	1.000
U_{zz}	1.000	1.000	1.000	1.000	1.000

Table 9.6: Relaxed energies calculated in Quantum Espresso

9.4.2 Relaxed FCC Iron

The configuration for Fe has been referred to throughout this work as FCC, but the relaxed collinear spin-polarized DFT calculation predicts that the structure is slightly face centered tetragonal (FCT). The optimal magnetic setting is antiferromagnetic (fig. ??) and, with the spin aligned in the z-direction, the y-axis of the unit cell is approximately 5% larger than the x and z axis.

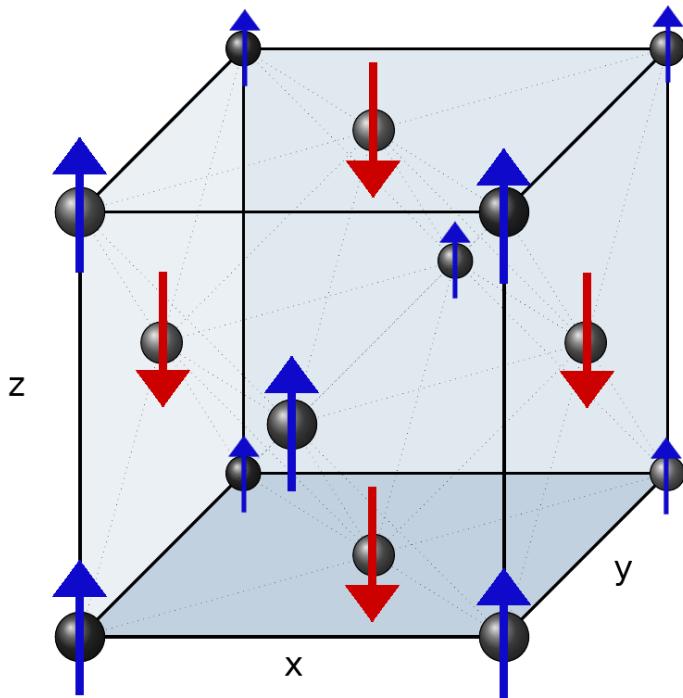


Figure 9.3: Magnetic alignment along the z-axis, antiferromagnetic Fe fcc

The atoms in the (0,1,0) plane are majority spin-up electrons, along the z-axis, and those in the (0,2,0) plane are spin-down, along the z-axis (fig. ??). Following the relaxation calculation, all the atoms were in their standard FCC positions, and the magnetic moment for each atom was 1.4901 Bohr magneton and -1.4901 Bohr magneton for the spin up and down respectively.

9.4.3 Iron-Palladium Atom Positions

The stainless steel being studied is to be doped with 1% or less of a platinum group metal. Ideally, a 256 4x4x4 FCC supercell would be used, with 2 or 3 randomly placed Pd atoms, and the remainder Fe. This calculation would require more time and memory than was available, especially for repeat calculations, and a compromise of 1 Pd and 31 Fe was used instead.

Listing 9.1: Relaxed Fe Pd Configuration

```
CELL_PARAMETERS (alat= 13.017452732)
 1.000000000  0.000000000  0.000000000
 0.000000000  1.054129443  0.000000000
 0.000000000  0.000000000  0.999992919

ATOMIC_POSITIONS (crystal)
Pd      0.000000000  0.000000000  0.000000000
Fe2    0.255184314  0.255092166  0.000000000
```

```

Fe3    0.257059507 -0.000000000 0.257051795
Fe4    0.000000000 0.255082799 0.255177990
Fe1    0.000000000 0.000000000 0.500000000
Fe2    0.250476117 0.250764583 0.500000000
Fe3    0.257059507 0.000000000 0.742947204
Fe4    0.000000000 0.255082799 0.744821008
Fe1    0.000000000 0.500000095 0.000000000
Fe2    0.255184314 0.744908119 0.000000000
Fe3    0.250799614 0.500000095 0.250799503
Fe4    0.000000000 0.744917486 0.255177990
Fe1    0.000000000 0.500000095 0.500000000
Fe2    0.250476117 0.749235702 0.500000000
Fe3    0.250799614 0.500000095 0.749200497
Fe4    0.000000000 0.744917486 0.744821008
Fe1    0.499999800 0.000000000 0.000000000
Fe2    0.744815185 0.255092166 0.000000000
Fe3    0.742940093 0.000000000 0.257051795
Fe4    0.499999800 0.250771574 0.250467682
Fe1    0.499999800 0.000000000 0.500000000
Fe2    0.749523382 0.250764583 0.500000000
Fe3    0.742940093 -0.000000000 0.742947204
Fe4    0.499999800 0.250771574 0.749531317
Fe1    0.499999800 0.500000095 0.000000000
Fe2    0.744815185 0.744908119 0.000000000
Fe3    0.749199985 0.500000095 0.250799503
Fe4    0.499999800 0.749228711 0.250467682
Fe1    0.499999800 0.500000095 0.500000000
Fe2    0.749523382 0.749235702 0.500000000
Fe3    0.749199985 0.500000095 0.749200497
Fe4    0.499999800 0.749228711 0.749531317
End final coordinates

```

The start configuration is FCT Iron, having a slightly longer y axis than x or z, and the positions of the atoms and the structure were relaxed in Quantum Espresso.

9.5 QEEOS Python Code

9.5.1 Purpose of Code

A code was developed to automate the process of calculating the equation of state and elastic constants of a material using Quantum Espresso. It requires an input file and a starting PWscf input file.

- the input configuration is relaxed using the vc-relax option in PWscf
- configuration files are created to compute the equation of state, and PWscf is used to calculate the energies of these configurations
- to compute the elastic constants, nine distortions are applied to the relaxed configuration with the required number of steps for each strain applied to each distortion, and these are processed with PWscf
- once all DFT work has completed, the Birch-Murnaghan equation of state is fit to the first set of energies, and the nine orthorhombic elastic constants are fit to the results of the second set of energies

9.5.2 Source Code and Instructions

The source code and instructions on how to use the program are available to download from GitHub.

9.6 Calculated Elastic Properties

The planewave cutoff and k-points were converged to within the required parameters for energy and force but it is known that, in general, LDA and GGA pseudopotentials under and over estimate the lattice parameters. The bulk modulus and elastic constants were calculated for FCC Aluminium and BCC Iron in order to compare to experimental values.

A summary of the main data points is given here for FCC Al, BCC Fe, FCC Pd and FCC Fe, and the first three listed are compared to experimental data. A full output from the computer code is included in the appendix (chapter ??).

9.6.1 FCC Aluminium

The parameters used for the Al calculations were arrived at earlier using the convergence code. The remainder of the parameters are either default settings, or were selected following trial and error (for example, reducing the mixing beta to help achieve convergence). These settings are given in table ??.

Setting	Value
ecutwfc (Ry)	50
ecutrho	200
smearing (Ry)	0.04
k-points	11 11 11 1 1 1
nspin	0 (Non-magnetic)
pseudopotential	Al.pbe-nl-kjpaw-psl.1.0.0.UPF
etot_conv_thr	0.0001
forc_conv_thr	0.001
conv_thr	1.0D-6
diagonalization	david
mixing_beta	0.1
mixing_mode	plain

Table 9.7: Aluminium DFT settings

The results in table ?? show a good agreement between the experimental values for Al and the calculated values. The lattice parameter is a particularly good fit, being within 1% of the experimental value. The shear modulus does not match as well, but is within 20% of the experimental value. The RSS measurement of fit between the values was 2.54×10^2 .

	Al Experimental	Al DFT (this work)
Structure	Face Centered Cubic	Face Centered Cubic
a_0 (Angs)	4.05 Angstrom	4.04 Angstrom
Nearest Neighbour	2.86 Angstrom	2.86 Angstrom
Basis vectors	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$
E_{coh} (eV)	-3.36 eV	Not Calculated
Bulk Modulus B_0 (GPA)	76	77.6
Bulk Modulus $B_{0,r}$ (GPA)	-	75.4
Bulk Modulus $B_{0,g}$ (GPA)	-	75.4
Young's modulus E (GPA)	70	81.4
Shear Modulus G (GPA)	26	30.8
Poisson Ratio ν	0.35	0.32
Elastic Constants (GPA)	$\begin{bmatrix} 114 & 62 & 62 & 0 & 0 & 0 \\ 62 & 114 & 62 & 0 & 0 & 0 \\ 62 & 62 & 114 & 0 & 0 & 0 \\ 0 & 0 & 0 & 32 & 0 & 0 \\ 0 & 0 & 0 & 0 & 32 & 0 \\ 0 & 0 & 0 & 0 & 0 & 32 \end{bmatrix}$	$\begin{bmatrix} 110.9 & 57.7 & 57.7 & 0 & 0 & 0 \\ 57.7 & 110.8 & 57.5 & 0 & 0 & 0 \\ 57.7 & 57.5 & 110.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 34.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 34.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 34.0 \end{bmatrix}$

Table 9.8: FCC Aluminium: Experimental Values vs DFT Calculated Variables

9.6.2 BCC Iron

The equation of state and elastic constants were calculated for BCC Fe, with no-spin and collinear spin. The settings used for the calculations are given in table ??.

Setting	Value
ecutwfc (Ry)	71
ecutrho	430
smearing (Ry)	0.04
k-points	9 9 9 1 1 1
nspin	0 (Non-magnetic) and 2 (Collinear Spin)
pseudopotential	Fe.pbe-spn-kjpaw-psl.1.0.0.UPF
etot_conv_thr	0.0001
forc_conv_thr	0.001
conv_thr	1.0D-6
diagonalization	david
mixing_beta	0.1
mixing_mode	plain

Table 9.9: Iron DFT settings

	Fe Experimental	Fe DFT No Magnetism (this work)	Fe DFT Ferromagnetic (this work)
Structure	Body Centered Cubic	Body Centred Cubic	2.80 Angstrom
a_0 (Angs)	2.86 Angstrom [femendelev]	2.75 Angstrom	2.42 Angstrom
Nearest Neighbour	2.48 Angstrom [femendelev]	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$
Basis vectors	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$	Not Calculated	Not Calculated
E_{coh} (eV)	-4.32 [femendelev]	239.2	239.2
Bulk Modulus B_0 (GPA)	170	287.3	205.3
Bulk Modulus $B_{0,r}$ (GPA)	-	257.5	205.3
Bulk Modulus $B_{0,g}$ (GPA)	-	253.7	211.1
Young's modulus E (GPA)	211	-11960.7	79.4
Shear Modulus G (GPA)	82	-643.1	
Poisson Ratio ν	0.29	8.29	0.33
	$\begin{bmatrix} 243 & 145 & 145 & 0 & 0 & 0 \\ 145 & 243 & 145 & 0 & 0 & 0 \\ 145 & 243 & 243 & 0 & 0 & 0 \\ 0 & 0 & 0 & 116 & 0 & 0 \\ 0 & 0 & 0 & 0 & 116 & 0 \\ 0 & 0 & 0 & 0 & 0 & 116 \end{bmatrix}$	$\begin{bmatrix} 63.0 & 329.2 & 315.3 & 0 & 0 & 0 \\ 329.2 & 176.6 & 316.4 & 0 & 0 & 0 \\ 315.3 & 316.5 & 121.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 176.9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 180.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 180.0 \end{bmatrix}$	$\begin{bmatrix} 250.1 & 182.3 & 183.3 & 0 & 0 & 0 \\ 182.3 & 249.1 & 182.9 & 0 & 0 & 0 \\ 183.2 & 182.9 & 251.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 139.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 139.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 139.3 \end{bmatrix}$
Elastic Constants (GPA)			

Table 9.10: BCC Iron: Experimental Values vs DFT Calculated Variables

The non-magnetic Iron calculation was particularly poor. Although the lattice parameter was within 5% of the experimental value, the structure was unstable and had negative values for both the Young's modulus and shear modulus (table ??). The RSS measurement of fit between the values for the was 2.01×10^5 .

With collinear spin enabled, the structure is stable. The lattice parameter is still slightly under the experimental, but is now within almost 2% of the value. The bulk modulus predicted by fitting the Birch-Murnaghan equation of state is more than 40% the experimental value, but the Reuss and Voigt values derived from the calculated elastic constants are just 20% difference to the experimental value. The latter two values are calculated using the elastic constants, rather than the equation of state. This highlights the importance of using collinear spin calculations, despite the cost in computing time. The RSS measurement of fit between the values was 1.09×10^4 giving an improvement of at least one magnitude.

9.6.3 FCC Palladium

The equation of state and elastic constants were calculated for FCC Pd with no-spin. The settings used for the calculations are given in table ??.

Setting	Value
Ecutwfc (Ry)	71
Ecutrho	430
Smearing (Ry)	0.04
K-points	9 9 9 1 1 1
Nspin	0 (Non-magnetic)
Pseudopotential	Pd.pbe-spn-kjpaw_psl.1.0.0.UPF
etot_conv_thr	0.0001
forc_conv_thr	0.001
conv_thr	1.0D-6
diagonalization	david
mixing_beta	0.1
mixing_mode	plain

Table 9.11: Palladium DFT settings

There is a good agreement between the computed and experimental values for Pd, without the need to move from non-spin to collinear spin calculations (table ??). However, in the alloy calculations the presence of Fe will dictate the necessity for collinear spin. The RSS measurement of fit between the values was 4.12×10^3 .

	Palladium FCC	Palladium FCC (this work)
Structure	Face Centered Cubic	Face Centered Cubic
a_0 (Angs)	3.89	3.92
Nearest Neighbour	2.75	2.77
Basis vectors	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$
E_{coh} (eV)	-3.91	not calculated
Bulk Modulus B_0 (GPA)	195.5	184.4
Bulk Modulus $B_{0,r}$ (GPA)	-	173.7
Bulk Modulus $B_{0,v}$ (GPA)	-	173.7
Young's modulus E (GPA)	121	153.1
Shear Modulus G (GPA)	44	56.6
Poisson Ratio ν	0.39	0.35
Elastic Constants (GPA)	$\begin{bmatrix} 234 & 176.1 & 176.1 & 0 & 0 & 0 \\ 176.1 & 234 & 176.1 & 0 & 0 & 0 \\ 176.1 & 176.1 & 234 & 0 & 0 & 0 \\ 0 & 0 & 0 & 71.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 71.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 71.2 \end{bmatrix}$	$\begin{bmatrix} 218.5 & 151.4 & 151.4 & 0 & 0 & 0 \\ 151.4 & 218.5 & 151.3 & 0 & 0 & 0 \\ 151.4 & 151.3 & 218.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 80.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 80.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80.3 \end{bmatrix}$

Table 9.12: FCC Palladium: Experimental Values vs DFT Calculated Properties

9.6.4 FCC Iron

The gamma phase of pure iron does not exist for experimental measurements to be made, and the results here will be used to fit the FCC iron potential in chapter ???. The settings used for the calculations are given in table ??.

Setting	Value
ecutwfc (Ry)	71
ecutrho	430
smearing (Ry)	0.04
k-points	9 9 9 1 1 1
nspin	2 (Collinear Spin)
pseudopotential	Fe.pbe-spn-kjpaw_psl.1.0.0.UPF
etot_conv_thr	0.0001
forc_conv_thr	0.001
conv_thr	1.0D-6
diagonalization	david
mixing_beta	0.1
mixing_mode	plain

Table 9.13: Iron DFT settings

The structure is stable under the orthorhombic stability conditions given in section 5.2.7. Whilst there are no experimental measurements to check against, the values are sane when compared to the other calculations performed here (table ??). The C_{44} , C_{55} and C_{66} values are slightly large when compared with those for BCC, but further DFT calculations would need to be performed to investigate this.

Iron FCC (this work)						
	Face Centered Cubic	Face Centered Cubic				
Structure						
a_0 (Angs)	-		3.42 Angstrom			
Nearest Neighbour	-		2.86 Angstrom			
Basis vectors	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$		$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$			
E_{coh} (eV)	-		-4.26			
Bulk Modulus B_0 (GPA)	-		226.1			
Bulk Modulus $B_{0,r}$ (GPA)	-		217.3			
Bulk Modulus $B_{0,v}$ (GPA)	-		226.6			
Young's modulus E (GPA)	-		356.8			
Shear Modulus G (GPA)	-		144.8			
Poisson Ratio ν	-		0.23			
Elastic Constants (GPA)	-		$\begin{bmatrix} 364.6 & 141.6 & 233.8 & 0 & 0 & 0 \\ 141.6 & 298.7 & 130.4 & 0 & 0 & 0 \\ 233.8 & 130.4 & 364.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 186.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 266.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 186.3 \end{bmatrix}$			

Table 9.14: Iron FCC: DFT Calculated Properties

9.6.5 FCC Ruthenium

Ruthenium at standard conditions exists as hexagonal close packed, but as this work is focused on FCC steel doped with PGMs, the properties are computed for FCC Ruthenium. The settings used for the calculations are given in table ??.

Setting	Value
ecutwfc (Ry)	71
ecutrho	430
smearing (Ry)	0.04
k-points	9 9 9 1 1 1
nspin	2 (Collinear Spin)
pseudopotential	Ru.pbe-spn-kjpaw-psl.1.0.0.UPF
etot_conv_thr	0.0001
forc_conv_thr	0.001
conv_thr	1.0D-6
diagonalization	david
mixing_beta	0.1
mixing_mode	plain

Table 9.15: Ruthenium DFT settings

Ruthenium FCC (this work)						
Structure	Face Centered Cubic	Face Centered Cubic				
a_0 (Angs)	-	3.81 Angstrom				
Nearest Neighbour	-	2.69 Angstrom				
Basis vectors	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$				
E_{coh} (eV)	-	-				
Bulk Modulus B_0 (GPA)	-					
Bulk Modulus $B_{0,r}$ (GPA)	-					
Bulk Modulus $B_{0,v}$ (GPA)	-					
Young's modulus E (GPA)	-					
Shear Modulus G (GPA)	-					
Poisson Ratio ν	-					
Elastic Constants (GPA)	-	$\begin{bmatrix} 364.6 & 141.6 & 233.8 & 0 & 0 & 0 \\ 141.6 & 298.7 & 130.4 & 0 & 0 & 0 \\ 233.8 & 130.4 & 364.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 186.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 266.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 186.3 \end{bmatrix}$				

Table 9.16: Ruthenium FCC: DFT Calculated Properties

9.7 QEFORFIT Python Code

A program was created in Python that generates input files for PWscf. The user specifies a template file then parameters such as the crystal type, lattice parameters, defects, interstitials and vacancies. The user may choose whether or not the atoms should be moved from their perfect crystal location by a small amount and whether the lattice parameter should be varied. The program then creates the user specified number of input files for PWscf.

9.7.1 Source Code and Instructions

The source code and instructions on how to use the program are available to download from GitHub.

<https://github.com/BenPalmer1983/qeforfit>

9.7.2 Gaussian Random Distribution

The positions of the atoms are moved from the perfect crystal locations by a random amount. The minimum and maximum amount to be moved is specified by the user, and a random value between these is selected. The distribution is not flat but takes the shape of a Gaussian curve (fig ??), so there is less chance that the atom will be moved to either extreme.

The Gaussian randomisation class is detailed in the appendix (??). As the atoms are perturbed further and further, the energy and therefore temperature of the system increases. This may change depending on the element and crystal structure, but a maximum displacement for aluminium atoms of 0.1 angstrom increases the energy and temperature to approximately 40K (fig. ??).

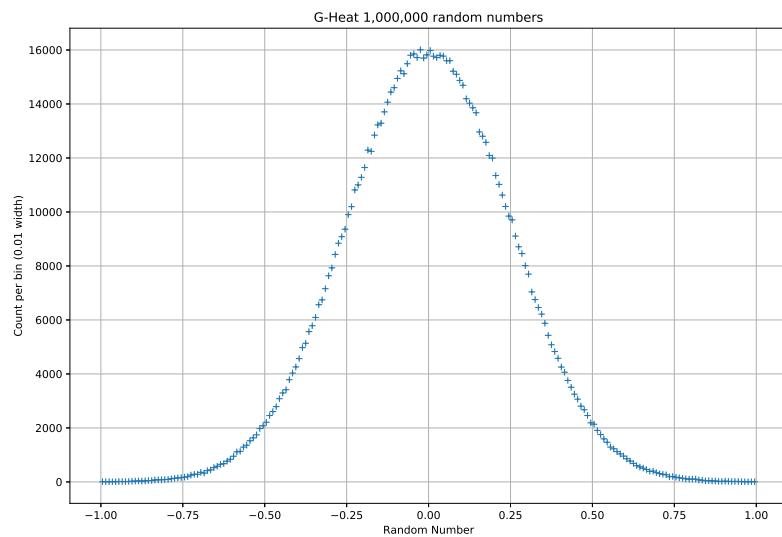


Figure 9.4: Gheat distribution function

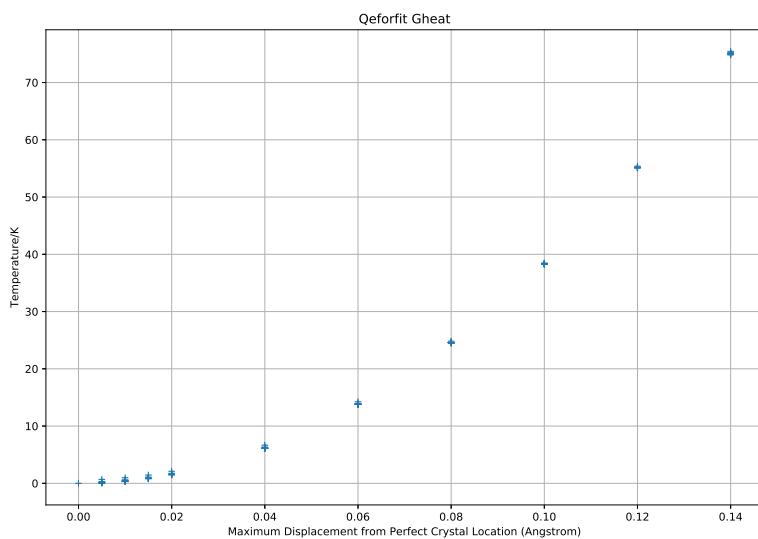


Figure 9.5: Maximum displacement from perfect crystal position against the increase in temperature of the system

9.7.3 DFT Configuration Database

The QEFORFIT code was used to create files that contributed towards the DFT configuration database. The broken symmetry caused by the random perturbation of atoms in the configurations caused the calculation to take a long time to complete. The calculations were also very resource intensive, using up to 100GB of RAM and 20-40 cores for up to a day per configuration.

In addition to this, quite often the SCF convergence would either fail, or complete the maximum number of iterations (which was set high, at 100) without converging, which would also be considered a fail (listing ??).

Listing 9.2: Sample output file to show resource usage and sample error codes

```
1 -----
2 WARNING: There was an error initializing an OpenFabrics device.
3
4 Local host: bear-pg0211u07a
5 Local device: mlx5_0
6 -----
7
8 Program PWSCF v.6.3 starts on 10Nov2020 at 7:58:30
9
10 This program is part of the open-source Quantum ESPRESSO suite
11 for quantum simulation of materials; please cite
12 "P. Giannozzi et al., J. Phys.:Condens. Matter 21 395502 (2009);"
13 "P. Giannozzi et al., J. Phys.:Condens. Matter 29 465901 (2017);"
14 URL http://www.quantum-espresso.org",
15 in publications or presentations arising from this work. More details at
16 http://www.quantum-espresso.org/quote
17
18 Parallel version (MPI & OpenMP), running on 40 processor cores
19 Number of MPI processes: 40
20 Threads/MPI process: 1
21
22 MPI processes distributed on 1 nodes
23 R & G space division: proc/nbgrp/npool/nimage = 40
24
25 ---
26
27 Estimated max dynamical RAM per process > 2.11 GB
28
29 Estimated total dynamical RAM > 84.34 GB
30 Generating pointlists ...
31
32 ---
33
34 Writing output data file zDmzKgmzhrarwYka.save/
35 -----
36 Primary job terminated normally, but 1 process returned
37 a non-zero exit code.. Per user-direction, the job has been aborted.
38 -----
39 -----
40 mpirun detected that one or more processes exited with non-zero status, thus causing
41 the job to be terminated. The first process to do so was:
42
43 Process name: [[18842,1],0]
44 Exit code: 38
45 -----
```

Due to the large amount of resources required and the regular failure to converge, there were a reduced number of configurations: 1 for Iron, 2 for Iron-Palladium, 10 for Palladium (list. ??).

The Iron and Palladium PWscf output files produced by the QEEOS code, that were used to calculate the equation of state and elastic constants, were added to the database of files, giving a total of 154 configurations for the Iron-Palladium potential fitting.

Chapter 10

Results: Interatomic Potential Fitting

A computer code was developed to fit interatomic potentials to DFT calculated energy, force and stress data. It also uses bulk property data including the bulk modulus, lattice parameter, cohesive energy and elastic constants. It was written in Python and Fortran and has been designed to be easy to add new fitting data, such as cohesive energy and surface energy plots.

The code has been used with the results from chapter 9 to fit two potentials for FCC Fe-Pd. The cohesive energy and surface energy plots were calculated using these potentials.

10.1 EAMPA Python Code

10.1.1 Purpose of Code

This code was developed to analyse interatomic potentials and fit their parameters to data. It currently has the following features:

- calculate energy, forces and stress of an ensemble of atoms
- calculate bulk properties of a potential for pure BCC or FCC structures
- fit parameters using simulated annealing, genetic algorithm, Nealder-Meade, Levenberg-Marquardt, BFGS
-

10.1.2 Source Code and Instructions

The source code and instructions on how to use the program are available to download from GitHub.

https://github.com/BenPalmer1983/eampa_v3

10.1.3 Modular Design

After several iterations, a Python-Fortran version was developed. The Python code was designed to handle the user input, output and the fitting process. The Fortran modules were designed with a separate module for each task: energy/force/stress, bulk properties, surface energy and defect calculations. The surface energy and defect module is incomplete. More modules may be easily added at a later date.

10.1.4 Potential Types

The code has been designed specifically for many body potentials. These include Finnis-Sinclair, embedded-atom method, two band embedded-atom method but not the modified embedded-atom method. It may be reduced to pair potentials only by setting the density function and embedding functional to “zero” (see appendix ?? for function details).

10.1.5 Energy, Force and Stress Calculations

The energy, forces and stresses are calculated in a library created with F2PY. The method is discussed in section 5.3.6 and the subroutine used is included in the appendix (section ??).

10.1.6 Bulk Property Calculations

The bulk property calculations are also performed in Python using a library created with F2PY. In order to speed up the calculation, the equation of state and elastic constant configurations use the known lattice parameter set by the user. The methods used are discussed in chapter 5 and the subroutine that creates the configurations, with which the module calculates the energies used to plot the equation of state and other distortions, is included in the appendix ??.

10.1.7 RSS - How Well The Potential Fits Experimental & DFT Data

The fitting process measures how well a potential fits the data when the initial parameters are given and every time a new set of parameters are proposed. They are weighted and computed in the efs and bp modules. They are the residual squared sum (RSS) of the differences between calculated energies, forces, stress, lattice parameter, cohesive energy, bulk modulus and elastic constants and the known values. The sum of these values measures how well the potential fits.

10.1.8 Analytic Potential Fitting

In earlier versions, the code used tabulated potentials only. Any analytic potentials were converted to tabulated. This was for two reasons: the functions could be easily exported and used by other codes that may not have the same analytic functions, and interpolation between tabulated data was faster to compute the value and gradient of a function at a certain point than several of the analytic functions (in particular the knot-to-knot splines).

A negative aspect is that, whilst the pair and density functions have a fixed range (from 0 to the cutoff radius), the range for the embedding functional depends on the density function, and the range of densities computed at each atom location. Rather than compute this first, I re-wrote the code to use analytic functions only, with the option of outputting as a tabulated function for use in other programs. A number of Python scripts were created to fit pre-existing tabulated potential functions into analytic functions to use as a starting point for the fitting procedure.

10.1.9 Overview of the Potential Fitting Process

The input files are read in by Python that then iterates over the fitting process while changing the potential. The Fortran modules calculate the energy, force and stress of the loaded configurations as well as the bulk property calculations for pure crystals of the atomic species and structures specified by the user. To save on

processing time, the neighbour lists for all configurations, including those used to calculate the equation of state and elastic constants, are only calculated once.

The step by step flow of the program while fitting parameters of a potential is as follows:

1. Read input files
2. Load Fortran Modules
3. Load potentials into Python and Fortran
4. Load configurations into Python and Fortran
5. Allocate memory
6. Create neighbour list for all configurations
7. Create eos and ec configurations and neighbour lists
8. Begin fitting loop in Python
9. Loop starts - loop through potential function parameters
 - (a) Update potential and load parameters to Fortran modules
 - (b) Loop starts - loop through configurations
 - i. Calculate energy, force, stress using pre-calculated neighbour lists
 - (c) End loop
 - (d) Loop starts - bulk property calculations for each pure element
 - i. Calculate eos and ec values and bulk properties using pre-calculated neighbour lists
 - (e) End loop
 - (f) Calculate rss between these values and known/dft
 - (g) Save parameters if best
 - (h) Make new parameters
10. End loop
11. Output results

Once the fitting process has completed, the best set of parameters are stored for the user along with a plot of the potential functions, equation of state and elastic constant energy-strain distortion plots (where these are used in the fitting process).

10.1.10 Function Types

The function types available for pair, density and embedding are listed in the appendix (??). More function types may be easily added by editing the Fortran fnc module and recompiling the efs and bp modules.

Most of the functions are basic mathematical functions that use built in operators and functions. Several are more complex, such as the cubic spline, quintic spline and knot-to-knot splines.

10.1.11 Cubic and Quintic Splines

These splines have been discussed in section 5.3.4. Several papers have previously used summed polynomial splines that employ the Heaviside-step function to cut off these functions at the desired points. The cubic term, $(r - r_i)^3$, ensures the function and its first and second derivatives equal zero and are continuous at the cutoff points.

$$V(r) = \sum_i^N a_i (r - r_i)^3 H(r_i - r)$$

where

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \quad (10.1)$$

This was implemented in Fortran so that an efficient vectorised version could be used by Python. The Fortran code for the spline function is in the appendix ??.

10.1.12 Knot to Knot Splines

A Fortran module was created for the knot-to-knot spline function. It takes a set of knots and splines between the knots pair at a time using the x , y , y' values to construct a set of linear equations to solve.

$$\begin{aligned} c_0 + c_1 x_A + c_2 x_A^2 + c_3 x_A^3 &= f(x_A) \\ 0 + c_1 + 2c_2 x_A + 3c_3 x_A^2 &= f'(x_A) \\ c_0 + c_1 x_B + c_2 x_B^2 + c_3 x_B^3 &= f(x_B) \\ 0 + c_1 + 2c_2 x_B + 3c_3 x_B^2 &= f'(x_B) \end{aligned} \quad (10.2)$$

$$\begin{bmatrix} 1 & x_A & x_A^2 & x_A^3 \\ 0 & 1 & 2x_A & 3x_A^2 \\ 1 & x_B & x_B^2 & x_B^3 \\ 0 & 1 & 2x_B & 3x_B^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} f(x_A) \\ f'(x_A) \\ f(x_B) \\ f'(x_B) \end{bmatrix} \quad (10.3)$$

A subroutine was created to calculate the coefficients for the 3rd order or 5th order polynomial to spline the two knots. This subroutine and the remainder of the module are available to view and download on the GitHub repository for this project. The Fortran code for a cubic and cubic with ZBL inner core knot-to-knot spline are included in the appendix (section ?? and section ??).

10.1.13 Simulated Annealing

The simulated annealing algorithm was considered for optimisation, and may be implemented as an option in the future. The algorithm looks for better solutions and replaces the current value with better values. It also

accepts worse solutions with a probability that decreases with “temperature”. Similar to the way atoms find optimal positions by annealing and cooling, the algorithm starts with a high temperature and it reduces this temperature over time. It is used to search for the global optimum.

The algorithm used is given in the appendix (section ??).

10.1.14 Genetic Algorithm

The global optimum is searched for using a genetic algorithm. The user chooses the parameters for the algorithm and it attempts to search for the optimum parameters to fit the potential functions to the data.

The user settings include the following:

- population size
- “fresh” (new randomly generated) parameters
- number of generations
- number of extinction events
- how often the top solutions are enhanced with linesearch and gradient descent

The subroutine that breeds the parameters randomly combines the parameters from two parent solutions to create two children parameter sets. The parents are either both from the existing pool of solutions, or a parent from the existing pool with a parent from a fresh set, and the fresh set of parameters changes with each generation (fig. ??).

Mixing Parameters from Parents to make Children

Parent A	P1	P2	P3	P4	P5	P6	P7	P8
Parent B	P1	P2	P3	P4	P5	P6	P7	P8
Child A	P1	P2	P3	P4	P5	P6	P7	P8
Child B	P1	P2	P3	P4	P5	P6	P7	P8

Figure 10.1: Parameter Breed Event

There is a chance that the child solutions will mutate slightly from the two parent combination, and this chance is set by the user.

During the breed event, the program will check whether or not the two children already exist within the solution pool. If they are allowed to also enter the pool, the solutions may quickly be filled with clones during the breeding process. The breed event automatically forces any clones to be varied so they become a different and distinct solution.

Every now and then an extinction event will occur that kills off any poor solutions and replaces them with variations of the better solutions. The user controls the frequency of the extinction event and the probability that a solution will be removed.

Top solutions are enhanced, at a frequency picked by the user, using a local optimisation algorithm: line-search and gradient descent. It does require many calculations and adds a substantial amount of time to the procedure,

but it uses the gradient of solutions to find more optimal solutions more efficiently than randomly generated or bred solutions.

The algorithm used is given in the appendix (section ??).

10.1.15 Newton Gauss Optimisation

The Newton Gauss optimisation estimates the second order derivatives matrix, the Hessian, by multiplying the Jacobian and its transpose. It is not yet implemented in the code, but may be in the future. The algorithm is used to find a local optimum, but it can be unstable or return errors.

$$\begin{aligned}\vec{J}^T \vec{J} \vec{p} &= \vec{J}^T \vec{r} \\ \vec{r} &= \vec{y} - f(\vec{x}, \vec{p})\end{aligned}\tag{10.4}$$

10.1.16 Levenberg-Marquardt Optimisation

The Levenberg-Marquardt algorithm is similar to the Newton-Gauss method, but the estimate for the Hessian does not need to be calculated for each loop of the algorithm. The parameter, λ , is changed and the Hessian is updated periodically. This is not yet implemented in the code, but may be added in the future.

$$\begin{aligned}\left(\vec{J}^T \vec{W} \vec{J} + \lambda \text{diag} \left(\vec{J}^T \vec{W} \vec{J} \right) \right) \vec{p} &= \left(\vec{J}^T \vec{W} \vec{r} \right) \\ \vec{r} &= \vec{y} - f(\vec{x}, \vec{p})\end{aligned}\tag{10.5}$$

10.1.17 Python SciPy Optimization Functions

Additional optimisation functions have been used by importing the SciPy module into the program, and these may be used in conjunction with those listed above. Several have been used including the bin-hopping, Nealder-Meade and BFGS. The bin-hopping was unreliable and crashed on a number of occasions, so I used the simulated annealing and genetic algorithms programmed specifically for this work. The Nealder-Meade and BFGS were both used following the run of global optimisation and increased the efficiency of improving the parameters.

10.1.18 Minimising Runtime

Python, Fortran & OpenMP

When fitting a potential, for each trial set of functions, the program must calculate the energy, forces and stress for each of the configurations and tens or hundreds of energy only calculations for the equation of state and elastic constant calculations. Python is a well used programming language in many areas of Science and it has the benefits of many modern languages. For certain specific tasks, it is slow in comparison to Fortran.

The energy, stress and force calculations were programmed in Fortran and use OpenMP to calculate the many configurations in parallel. On a single thread, the fitting process could take weeks, but on a 20 core node the calculation time was reduced to under a couple of days.

Function Caching

The first version of the code used tabulated potentials, and this removed the task of programming different functions (and the wide range of complexities amongst those functions) and replaced it with a quick search and interpolation of tabulated data. However, the final version was developed to use and fit with analytic potentials. Several of the optional functions, including the knot-to-knot spline function, drastically increased the computational time.

The bulk property module in particular uses configurations with atoms in regular places, with there being a small set of unique atom separation values. Rather than generate tabulated data for each function, to then interpolate, the input and output values are cached. Once the value is calculated, it is quickly retrieved for subsequent calculations with the same input value. This has increased computation speed of the analytic potentials to a speed similar to using tabulated functions.

10.1.19 Interpolation for Non Analytic Potentials

In the current version, only analytic potentials are available for fitting. There are several subroutines ready if tabulated functions are reintroduced. These subroutines use Lagrange polynomials to interpolate values of the function (eq. ?? ??).

$$D = \{(x_0, y_0) (x_1, y_1) (x_2, y_2) \cdots (x_n, y_n)\} \quad (10.6)$$

$$p_n(x) = \sum_{i=0}^n y_i L(i, x) \text{ where } l(i, x) = \prod_{j=0, j \neq i}^n (x - x_j) \quad (10.7)$$

There are typically hundreds or thousands of data points, so to interpolate, the closest few points are used. Throughout the computer code four point interpolation was the preferred method, to balance computational speed with a well fitting polynomial.

The gradient of the potential functions are also computed using Lagrange polynomials (eq. ??).

$$q_n(x) = \sum_{i=0}^{i=n} y_i g(i, x) \quad (10.8)$$

$$\text{where } g(i, x) = \left(\prod_{j=0, j \neq i}^{j=n} \frac{1}{x_i - x_j} \right) \times \left(\sum_{k=0, k \neq i}^{k=n} \prod_{j=0, j \neq k, j \neq i}^{j=n} (x - x_j) \right)$$

A pseudocode for both interpolation of the function points and derivatives are available in the appendix (section ?? and ??).

10.2 Iron-Palladium Potentials: Data and Initial Potential Parameters

The developed code was used to fit a potential for Fe, Pd and their alloy. A number of problems and limitations were encountered, and these will be discussed in more detail throughout this chapter and the conclusion.

10.2.1 Bulk Properties for Fitting

The fitting procedure is for both Fe and Pd, but the FCC allotrope of Fe that is used in this work does not exist, at room temperature and pressure. Experimental data was available for Pd, but DFT generated data was required for Iron.

Palladium [FCC]

The input parameters for Pd are given in table ??.

Property	Value used in potential fitting
Element	PD
Structure	Face Centered Cubic
a_0	3.89 Angstrom [143]
Nearest Neighbour	2.75 Angstrom [143]
Basis vectors	$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$
E_{coh}	3.91 eV [100]
B_0	195.5 GPA [100]
Elastic Constants	$\begin{bmatrix} 234.1 & 176.1 & 176.1 & 0 & 0 & 0 \\ 176.1 & 234.1 & 176.1 & 0 & 0 & 0 \\ 176.1 & 176.1 & 234.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 71.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 71.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 71.2 \end{bmatrix}$

Table 10.1: Pd input parameters for fitting

Iron [FCC]

The input parameters were calculated by DFT, with an FCT structure rather than FCC, and the calculated bulk modulus, elastic constants and other related properties are in table ??.

Property	Value used in potential fitting
Element	Fe
Structure	Face Centered Cubic
a_0	3.59 Angstrom
Nearest Neighbour	1.85 Angstrom
Basis vectors	$\begin{bmatrix} 0.96 & 0.0 & 0.0 \\ 0.0 & 1.00 & 0.0 \\ 0.0 & 0.0 & 0.96 \end{bmatrix}$
E_{coh}	-4.32 eV
B_0 (GPA)	226.1
E (GPA)	356.8
G (GPA)	144.8
Poisson Ratio η	0.23
Elastic Constants	$\begin{bmatrix} 364.6 & 141.6 & 233.8 & 0 & 0 & 0 \\ 141.6 & 298.7 & 130.4 & 0 & 0 & 0 \\ 233.8 & 130.4 & 364.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 186.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 266.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 186.3 \end{bmatrix}$

Table 10.2: Fe input parameters for fitting

10.2.2 Reference Database

A reference database of 72 configurations for Fe, 80 configurations for Pd and 152 combined for Fe-Pd was created using the DFT code PWscf. This process and the python code used is detailed in section ??.

10.2.3 Potential Functions & Functionals

There are a large number of functions and functionals used in the literature, and the potentials provided by H. Sheng are numeric[102] (section 5.3.5). In this work, it was decided to use a knot-to-knot spline for the pair potential that is attached to a ZBL potential for small separations, with a fixed node at the cutoff, with a zero value and gradient. In the first instance the radial electron density was calculated using DIRAC and from this a knot-to-knot spline was fitted with a forced cutoff to $\rho(r_{cut}) = 0, \rho'(r_{cut}) = 0$. This was then replaced with a knot-to-knot spline based on the Sheng density potentials and the reason for this will be discussed in section ??.. Finally, the embedding functionals take the form $f(x) = p(1) \times \sqrt{r} + p(2) \times r^2$ as used by Mendeleev[femendelev][feacklandmendelev], Ackland[feacklandmendelev], Olsson[98] and Wallenius[98].

DIRAC Density Plots

Whilst the potentials in this work are matched to DFT calculated forces, energies and bulk properties of the materials, using the radial electron density gave a starting point for each element. For an alloy, the density functions cannot have an arbitrary scale, as the density of multiple elements will be summed to calculate the energy of an atom embedded in that density. For this reason, it was also reasonable to start with density distributions computed by DIRAC, each relative in scale to the other (fig. ?? and fig. ??).

Unfortunately, as may be expected, there were a number of issues in using the density distribution of a single atom. The atomic separation for both iron and palladium, for a relaxed crystal, begins at approximately 2-2.5 angstrom, but most of the electrons, from all shells, are distributed much closer. The plot does not take into consideration the valence electrons shared in the crystal, bonding the metal atoms together.

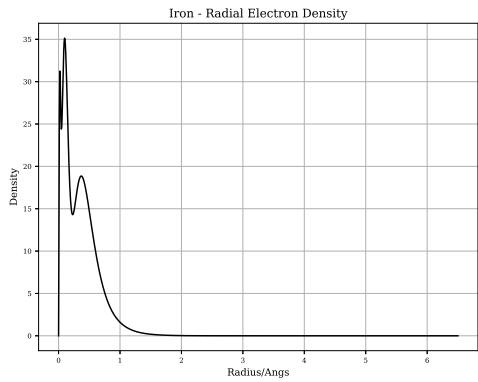


Figure 10.2: DIRAC calculated radial electron density for Iron

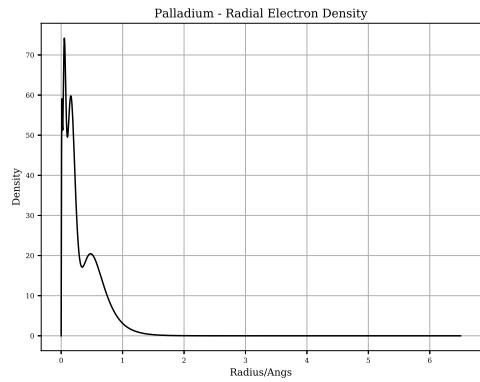


Figure 10.3: DIRAC calculated radial electron density for Palladium

From the point of view of the potential, the pair potential will dominate at small separations due to the ZBL function. The density function will be important at separations of 2 angstrom up to the cutoff radius (5-6 angstrom).

For this reason, the idea to use the DIRAC computed densities was abandoned in favour of using the Pd Sheng potential.

Sheng Potentials

The Sheng potentials available are numeric and so the functions used in this work were first fit to the tabulated data[29]. As the Sheng Pd is a potential for a FCC crystal, this was a starting point for both Fe and Pd (which is quite obvious for the case of Pd). Plot for the Pd potential are given in figs. ?? ?? ?? ??.

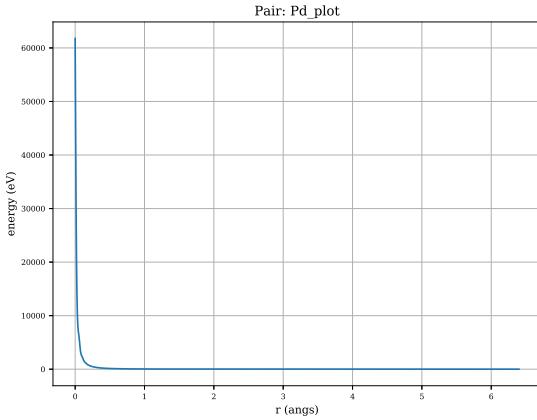


Figure 10.4: Sheng palladium pair function

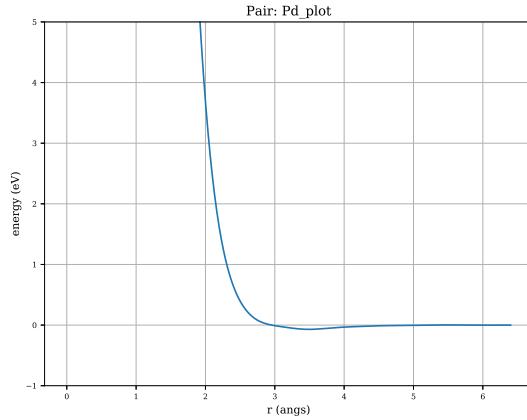


Figure 10.5: Sheng palladium pair function (zoomed in)

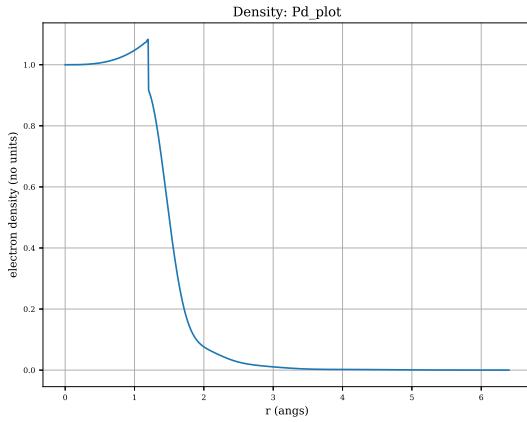


Figure 10.6: Sheng palladium density function

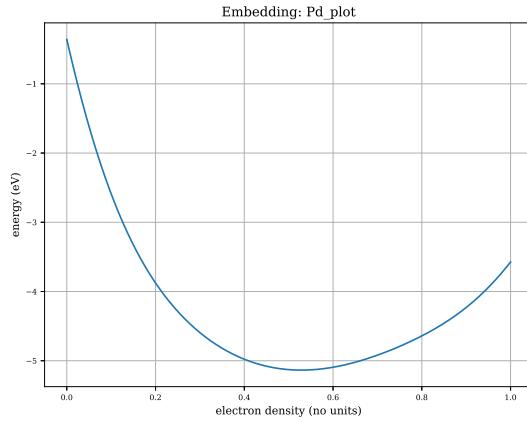


Figure 10.7: Sheng palladium embedding function

Preliminary Choices for the Potential Functions

The form of each function was decided by the function type, past forms from the literature and testing.

The pair potentials were a cubic knot-to-knot spline that was originally fit to the Sheng potential for Pd. The spline started with 30-40 knots, but several of those at small separations were removed, as varying them had little effect or no effect on the calculation of bulk properties or the energy, forces and stresses of the prepared atom configurations. The spline is forced to end with a value and derivative equal to zero at the cutoff radius.

Originally, the density function was a cubic knot-to-knot spline of the DIRAC produced density plots, but most of the density data points were at a small separation and again had little to no effect on the potential. The Sheng Pd density was instead used as a starting point, and the density close to 0 was also reduced so the ZBL pair would dominate at small separations.

$$F(\rho) = a \times \sqrt{\rho} + b\rho^2 + \rho^4 \quad (10.9)$$

The embedding functional was selected as a type used by Mendelev and Ackland for Fe (eq. ??), with a Finnis-Sinclair negative square root, a positive or negative squared term and a positive quartic term to ensure a low or negative energy while at small electron densities, but a positive energy as the electron density continues to increase.

Both the Fe and Pd potentials were derived separately as pure elements, and then again as an alloy.

10.3 Iron-Palladium Potentials: First Potential

The Fe potential consisted of a cubic knot-to-knot spline for the pair and density functions, and the Mendelev and Ackland type embedding functional as described in section ???. These functions were fit to the Sheng Pd plot to give the potential a starting point, and to ensure the density plot was a similar magnitude to the Pd potential that is also being derived, to give a reasonable starting point for the alloy fitting.

Similarly to the Fe potential, the Pd potential consisted of a cubic knot-to-knot spline were used for the pair and density with a 2 parameter embedding functional similar to that used by Mendelev and Ackland. The Sheng Pd potentials were used as a starting point. The parameters of the three functions were fit to the Sheng plots before then being fit to the DFT data and experimental bulk properties.

A total of 72 configurations were used in the fitting of Fe, but unfortunately only one of these had slightly randomised atomic positions. This was due to an increase in SCF cycles and a number of convergence failures with the slightly perturbed atom positions. The Pd potential was fit using the known experimental parameters and 80 DFT configurations, including 10 with randomised atom locations, slightly perturbed from the perfect crystal locations.

Finally, the two already derived potentials (above) were combined. A seventh potential was required to finish the potential, this being the Fe-Pd pair potential. The Fe-Fe pair potential was used as a starting point. All seven potentials were then fit to the Fe, Pd and Fe-Pd DFT configurations as well as both the Fe and Pd FCC bulk properties.

Throughout, the simulated annealing and genetic algorithms were use in combination to optimise the potential parameters. No local optimization methods were used with this potential.

Fe-Pd Potential Properties

The properties are calculated throughout the fitting process, but the final properties for both iron and Pd are presented in tables ?? and ??.

Parameter	Experimental/DFT	This Potential	Error %
a_0	3.42	3.58	4.7
e_0	-4.27	-4.27	0.0
B_0	222.0	238.8	7.6
C_{11}	365.6	356.8	2.4
C_{22}	298.7	296.1	0.9
C_{33}	364.0	356.8	2.0
C_{44}	186.3	180.9	2.9
C_{55}	266.8	263.2	1.3
C_{66}	186.3	180.9	2.9
C_{12}	141.6	139.1	1.8
C_{13}	233.8	241.7	3.4
C_{23}	130.4	127.3	2.4

Table 10.3: Experimental/dft properties for FCC iron vs those of the fe-pd potential

Parameter	Experimental/DFT	This Potential	Error %
a_0	3.89	3.95	1.5
e_0	-3.91	-3.91	0.0
B_0	180.0	171.3	4.8
C_{11}	234.0	248.9	6.4
C_{12}	176.0	167.2	5.0
C_{44}	71.2	67.6	5.1

Table 10.4: Experimental/dft properties for FCC palladium vs those of the fe-pd potential

Fe-Pd Property Plots

The equation of state and elastic constants were used to fit the potential, and for both iron and palladium the fit reasonable well. The data points produced for the Fe EOS do curve down slightly more than might be expected, but the data points for the Pd EOS do better fit the curve of the equation of state (appendix ??).

To test the cohesive energy a short python code was written to generate 4x4x4 FCC supercells with a lattice parameter ranging 9.58 angs to 36.9 angs for iron and 10.9 angs to 42.0 angs for palladium. An energy calculation was performed for each element, using the derived potential, and the results were plotted (appendix ??, figs. ??, ??, fig:pdv1cohesive, ??).

The cohesive energy plots for both iron and palladium take on a recognisable shape and, over all they do give the predicted cohesive energies. Neither is very smooth and to improve the reproduction of the correct energy at varying lattice parameters the energies should be computed using DFT and used during the potential fit.

The surface energies were also calculated. A small python program was created to generate a set of 4x4x4 FCC atoms. The position and separation of the atoms was fixed relative to one another, and anchored to the 0,0,0 coordinate, whilst the unit vector in the z axis was increased. This represents the transition from a bulk material to a slab with two surfaces in the xy-plane.

$$E_s = \frac{E_{total} - N_a \times E_{bulk}}{2A_{surface}}$$

where E_s = surface energy
 and E_{total} = total energy of slab
 and N_a = atoms in bulk
 and E_{bulk} = atom energy in bulk
 and $A_{surface}$ = surface area of slab in xy-plane

(10.10)

The calculated surface energies were $0.205\text{eV}/\text{ang}^2$ ($3,290\text{mJ}/\text{m}^2$) for iron and for $0.106\text{eV}/\text{ang}^2$ ($1690\text{mJ}/\text{m}^2$) palladium. The plots for these calculations are in appendix ?? (figs. ??, ??). Whilst the value for palladium is close to the experimental and Sheng values, ($2,000\text{mJ}/\text{m}^2$ and $1,747\text{mJ}/\text{m}^2$ respectively [29]), the calculations were performed without relaxing the atoms at each stage. This will be discussed further in chapter ??.

10.4 Iron-Palladium Potentials: Second Potential

A second attempt was made to derive an iron-palladium and several changes were made. First, the neighbour separations were examined for both palladium and iron, as shown in fig. ?? and fig. ???. Knot-to-knot splines offer a high degree of flexibility, but there is a problem. If the movement of one, or more, knots does not have

a direct effect on any of the bulk property or energy, stress, force calculations it will move during optimization resulting in what looks to be a very badly behaved function. It also unnecessarily increases the parameter space for optimization.

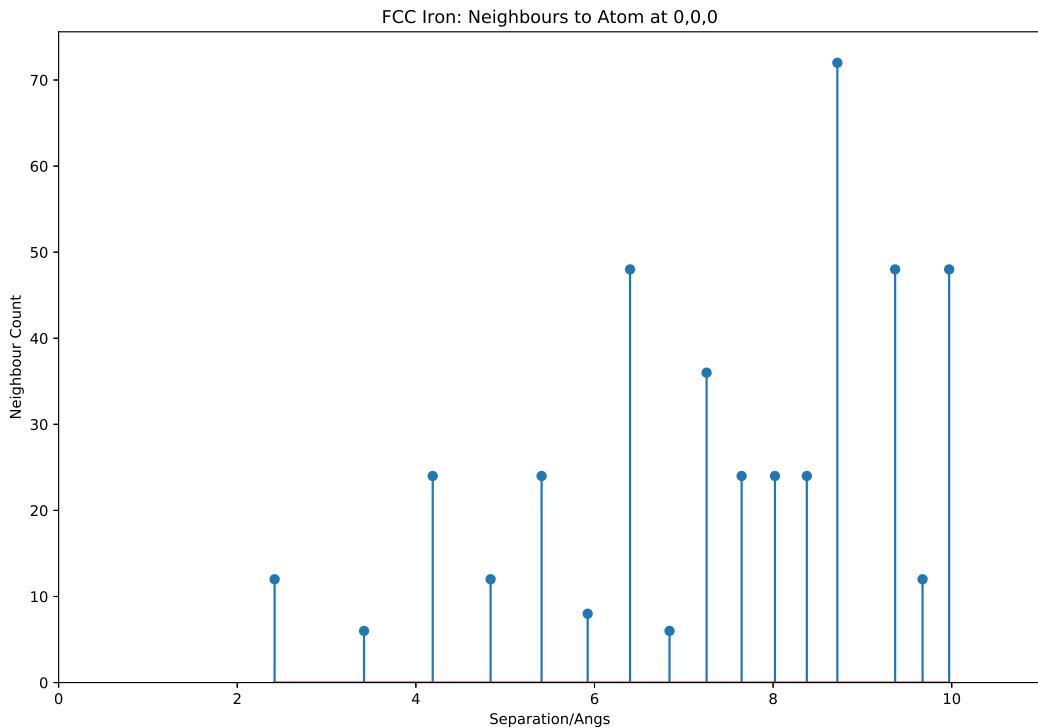


Figure 10.8: FCC Iron Neighbour Separation

The number of knots for the pair functions was reduced to 7, including the start and end knot. This would help reduce the number of superfluous knots, decreasing the parameter space, with the aim of also ending with a smoother function. A major change was also made to the cubic knot function. The entire function would be the ZBL potential summed with a cubic knot spline, to give a function with a continuous value and derivative (eq. ??).

$$v(r) = zbl(r, z_a, z_b) + spline(r, \vec{p}) \quad (10.11)$$

Another change implemented was to the way the parameters for the cubic knot spline functioned. Originally, the x positions for the knots were set by the user and the y positions were varied by the fitting program. The derivative at each point, needed to calculate each knot to knot spline, was interpolated. In the latest version the user specifies x, y(x) and y'(x) for the end knot (in this work, 6.5 0.0 and 0.0) and the x value of the start knot (in this work, 0.0). The remaining x, y(x) and y'(x) are all parameters that are varied by the fitting code.

The spline used previously for the density function was replaced with a 2 parameter Slater type function. The embedding functional remained the same. The starting potentials for both iron and palladium were fit to the Sheng potential for palladium and the same bulk properties and DFT configurations were used as for the first potential.

The fitting code was modified to include the SciPy optimization library. As a result, the potentials were fit using the simulated annealing and genetic algorithm for global optimisation, then Nelder-Mead and Broyden

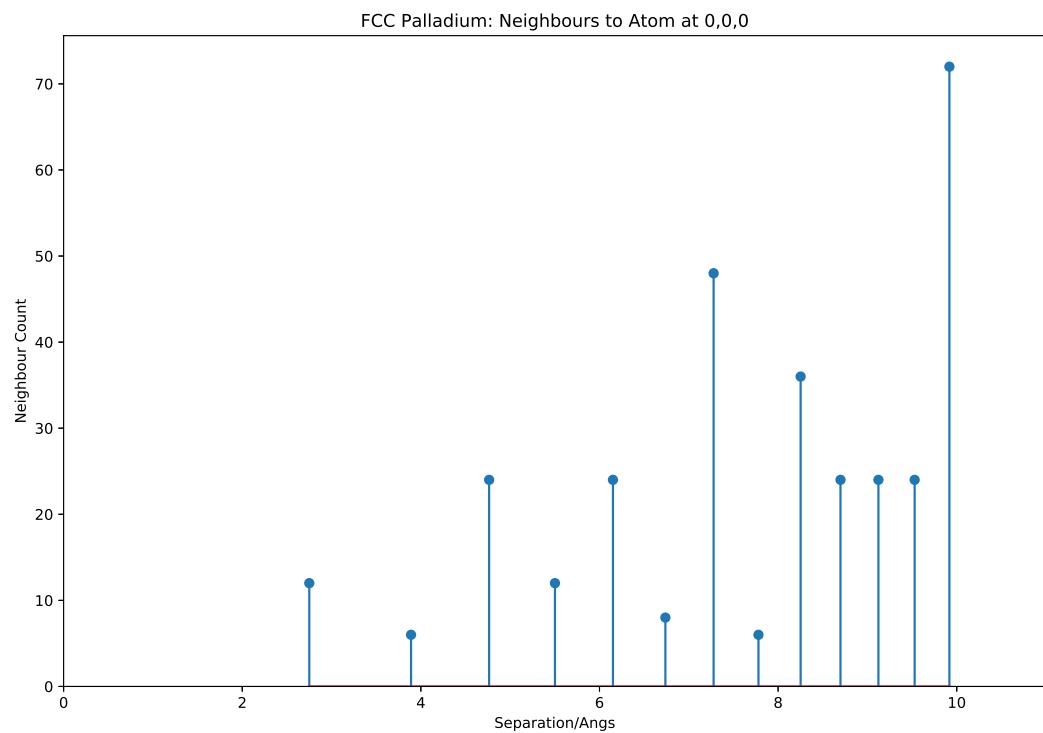


Figure 10.9: FCC Palladium Neighbour Separation

Fletcher Goldfarb Shanno to optimise the parameters locally.

Fe-Pd Calculated Properties

The properties are calculated throughout the fitting process, but the final values for both Fe and Pd are presented in tables ?? and ??.

Parameter	Experimental/DFT	This Potential	Error %
a_0	3.42	3.40	0.6
e_0	-4.27	-4.29	0.5
B_0	222.0	229.1	3.5
C_{11}	365.6	365.9	0.1
C_{22}	298.7	298.6	0.0
C_{33}	364.0	365.9	0.5
C_{44}	186.3	187.6	0.7
C_{55}	266.8	258.4	3.1
C_{66}	186.3	187.6	0.7
C_{12}	141.6	141.1	0.4
C_{13}	233.8	238.1	1.8
C_{23}	130.4	124.8	4.3

Table 10.5: Experimental/dft properties for FCC iron vs those of the fe-pd potential

Parameter	Experimental/DFT	This Potential	Error %
a_0	3.89	3.87	0.5
e_0	-3.91	-3.91	0.0
B_0	180.0	171.3	4.8
C_{11}	234.0	232.9	0.5
C_{12}	176.0	176.7	0.4
C_{44}	71.2	68.8	3.4

Table 10.6: Experimental/dft properties for FCC palladium vs those of the fe-pd potential

Fe-Pd Property Plots

The equation of state and elastic constants were used to fit the potential, and for both iron and palladium they fit reasonable well. The largest error is for the bulk modulus of Pd with a value 4.8% less than the actual bulk modulus.

The cohesive energies were calculated similarly to the first version, using the prepared configurations to expand the cell until the atoms are isolated from their neighbours. The energies were plotted at various values of lattice parameter (appendix ?? figs. ??, ??, fig:pdv2cohesive, ??). Both the Fe and Pd plots are quite poor with a smooth but erratic trend in data points with a sharp drop for the largest value of a_0 . This supports using DFT data to accurately calculate the cohesive energy plot data points and use these in the potential fitting process.

The calculated surface energies were $0.506\text{eV}/\text{ang}^2$ ($8,110\text{mJ}/\text{m}^2$) for iron and for $0.058\text{eV}/\text{ang}^2$ ($932\text{mJ}/\text{m}^2$) palladium. The energy for palladium is under half that expected. The value for iron in comparison is rather large, but the surface energy for FCC iron would need to be calculated using DFT to determine whether or not it is a sane value or not. The plots for the surface energy calculations are contained in appendix ?? (figs. ??, ??).

Chapter 11

Conclusions

It is known that Cr provides protection to stainless steel through a passive protective layer, but this becomes ineffective to protect the steel under irradiation, as Cr is depleted from the grain boundary. Platinum group metals also provide a mechanism to protect steel from corrosion, and experimentation or simulations may be used to investigate whether they can maintain high enough concentrations at the surface to continue to protect the steel against corrosion under irradiation. If either of the two chosen PGMs, Pd or Ru, are depleted at the grain boundary under irradiation as well as depletion of Cr, the steel will become susceptible to corrosion.

Experimental testing is cheaper, more controllable and more rapid at causing damage in a targeted volume if an ion beam is used rather than a neutron source. Either source will cause the target material to become radioactive. It is important to be able to calculate how radioactive, and how long after irradiation (to the desired damage, in DPA) it will be until the material is safe to be handled.

Computer simulations to simulate radiation damage will be much too large and complex for first-principles calculations. They will need either classical molecular dynamics or kinetic monte carlo simulations to encapsulate the much larger simulation volumes, to contain the atoms effected by the damage cascade, and to represent the grain boundary. This requires derivation of an appropriate interatomic potential using experimental data and first-principles data where it is not available, or is either impossible or impractical to measure.

11.1 Inter Granular Stress Corrosion Cracking

Austenitic stainless steel is particularly susceptible to inter granular stress corrosion cracking. It has good properties, including a good resistance to corrosion due to its high Cr and Ni content. Under irradiation, the Cr at grain boundaries is depleted with the formation of Cr carbides. Three requirements for inter granular stress corrosion cracking to occur are:

- a susceptible material (austenitic stainless steel)
- stress, welds, swelling, pressure, any applied stress (high pressure in reactor environment, swelling due to irradiation)
- a corrosive environment (radiolysis of water)

The passive Cr layer, under normal conditions, protects the steel from corrosion. The steel is made up from small grains, and the surface of these grains of crystalline metal are where the protective layer covers. During irradiation, Cr is depleted, forming Cr carbides between the grain boundaries. As the percentage of Cr drops, the surface becomes prone to corrosion.

Small quantities of platinum group metals may be added to a steel to also increase the corrosion resistance of the steel. Platinum group metals are rare and much more costly than Fe, Ni, Cr. Where irradiation is not present, it has been shown experimentally that the benefit due to Pd is lost due to the formation of Pd-Mn nanoparticles, where Mn is present in the steel, although no such particles are present when Pd was replaced by Ru. Pd does provide corrosion protection via cathodic modification when Mn is reduced to very small amounts as it no longer forms Pd-Mn precipitates during sensitisation. The outstanding question is whether or not Pd or Ru are depleted, enriched or are unaffected at the grain boundary when under irradiation.

11.2 Activity Code

Testing a material inside a nuclear reactor is an expensive experiment. There are many more ion sources around the world than high flux neutron sources, and ion beams are easier to direct due to the charge of the ion versus the neutral charge of the neutron. The ion energies required to create similar sized damage cascades in the material are high enough to also have a chance of transmuting the target nuclei. These transmuted nuclei are most likely going to be radioactive. To reach damage dose levels comparable to a component in a Gen IV reactor by the end of its life (up to 200 DPA), the target material will become dangerously radioactive.

Bateman's equation may be used to calculate the amount of an isotope in a decay chain of several isotopes, and thus how radioactive an isotope would be after a certain time. Major modifications were required to be made to the equation to also include branching factors and source rates for isotopes generated by an ion beam.

11.2.1 Modified Equation

Whilst the equation was first tackled by solving the differential equations head on, it became obvious that using Laplace transforms would be the logical choice; this was also the route Bateman followed when deriving the original equations. Once transformed, a numerical algorithm (Gaver-Stehfest) was programmed and used to solve, with some success. It isn't an exact solution, and the errors incurred often resulted in negative amounts of an isotope, and this was unacceptable. More progress was made towards solving the problem analytically, and by using partial fractions, obviating the need for a numerical method.

11.2.2 Activity Code

The Activity code was created to automate the process of calculating the activity of an ion irradiated target. The program:

- reads in simulation details (target composition, thickness, duration etc)
- reads in ion cross section data from the TENDL database
- processes SRIM exyz data points
- calculates the amount of each isotope by the end of the simulation
- calculates the activity of each isotope and the dose that an average human would be exposed to

A number of simplifications are made to the model but it predicted the radioactivity of certain peaks reasonably well with a sample of proton irradiated pure Fe. Given more time a range of alloys would also be irradiated by a proton beam and, once their activity is measured at several time intervals after irradiation, they would be compared to the activity predicted by the Activity code.

The first version of the Activity code was replaced by a second version that used Python rather than Fortran. The first version was more cumbersome for new users and it crashed for certain isotope, and these problems were fixed in the new version.

11.2.3 More Experimental Data Required

I have not had a great deal of experience practically measuring gammas, and I would expect there to be errors introduced due to my own inexperience. The range of beam energies, elements and beam duration are also very restricted, to one in each case (Fe and 36MeV protons). A wider range of Activity code.

Not taking the intensity into account and comparing the gamma peaks between the experimental work and the computer code, a number of peaks align, although their intensities differ. In particular, the peak near 511keV is much more intense in the experimental work (fig. ??), however the detector was calibrated to detect the 931keV peak.

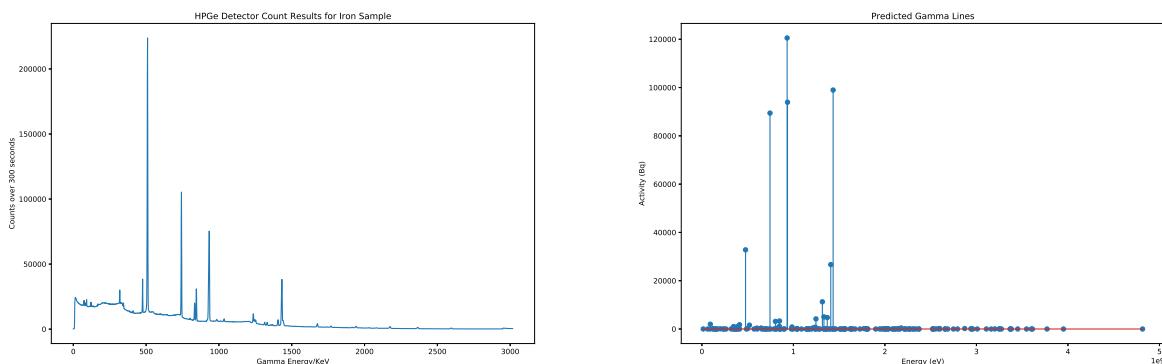


Figure 11.1: Measured gamma lines (36MeV protons) and predicted gamma lines (28MeV protons)

Arguably, the cross section data used by the Activity code has already been evaluated against experimental results. However, a slight change in cross section database has resulted in differing activities. The measurement of gammas and the dose rate at a fixed distance from irradiated targets, under a range of conditions as mentioned, would help to increase confidence in the code or give an indication of adjustments that need to be made to improve the accuracy of the code.

11.2.4 100DPA Activity Predictions

Using dose rates calculated with SRIM, the code was used to predict how radioactive Fe would become at a dose rate of 100DPA. It assumes the beam source is the Scanditronix MC40 using protons at maximum current for an uninterrupted duration of 134-391 days. In the worst case scenario, 25MeV protons result in over 20 grays per hour if irradiated to the required damage dose. However, by reducing the beam energy to 15MeV radioactivity of the target is reduced by a factor of 1,000. It may be reduced further by lowering the energy to 5 or 10MeV, but for a 0.5mm piece of steel, the ions would not have enough energy to pass through and the damage would be disproportionately concentrated in the top 100-250 microns.

11.2.5 Original Contribution

The modified Bateman equation as derived is an original contribution to knowledge. The activity code is also an original contribution, and its use shows how adjusting beam parameters will reduce the amount of radioactive material created and the risk to those handling irradiated samples.

11.3 Iron-Palladium Potential

11.3.1 Restricting the Potential to Iron-Palladium

It would be too complex a task to develop an interatomic potential that includes Fe, Pd, Cr, Ni and any other elements that make up a typical sample of austenitic stainless steel; Fe and Pd are the starting point, with the possibility of adding more elements to the potential in the future.

Pd exists in the FCC state at normal conditions, but Fe is BCC. The relatively high percentage of Ni in the steel stabilises the austenitic FCC structure. The bulk properties, such as the lattice parameter, bulk modulus, elastic constants and so forth, are experimentally known for Pd. As Fe doesn't exist in the gamma phase at normal conditions, DFT calculations were used to find the bulk properties of gamma Fe.

11.3.2 Collinear Spin DFT

With most models, there will be a trade off between simplicity, accuracy and the computational cost of the model. The parameters that had the largest impact on how long a DFT calculation will take to compute are the planewave energy cutoff and number of k-points used to integrate the Brillouin zone. These should be reduced as much as possible, while keeping the energy and force results within a tolerance determined by the user.

A choice in the complexity of the calculation also had to be decided upon, and this was whether to use a non-spin, collinear spin or non-collinear spin calculation. Pure Fe under normal conditions has a BCC structure and energetically favours a ferromagnetic configuration. Cr also has a BCC structure under normal conditions, but it's optimum structure magnetically is antiferromagnetic. Gamma phase Fe (FCC) is also antiferromagnetic, and so it will be important to include magnetism in the DFT calculations.

The options in increasing complexity, and computational time, are:

- non-magnetic
- collinear - suitable for high symmetry, ferromagnetic/antiferromagnetic
- non-collinear - spin may not be aligned in the same direction

For the Fe and Pd calculations, collinear spin was used and the starting configurations of the atoms were set up in accordance with the known configurations i.e. BCC Fe was configured such that the spins were aligned in the same direction, and FCC were set up in an antiferromagnetic setting. The collinear DFT calculations showed that Fe FCC, where magnetism is taken into account, is slightly tetragonal with the atoms arranged in a FCT structure.

A Python (QECONVERGE) code was developed to automatically converge the aforementioned parameters. Whilst automation worked well for the planewave cutoff parameters the choice of smearing type, the amount of smearing and number k-points was more difficult to automate. It was more useful to select a range of k-point values and smearing values and produce 2D force and energy plots so the user could make a decision based on these and choose the parameters that best suit their requirements.

During testing, a much larger number of k-points were needed for the predicted properties of Aluminium to replicate the known properties reasonably well. Unfortunately, the number of k-points used for Fe and Pd was constrained by the amount of time and the RAM available on the compute nodes. Ideally, more k-points would have been used, but this was not possible.

11.3.3 Bulk Properties of Orthorhombic FCC Iron

As pure Fe does not exist in its gamma phase at room temperature, DFT calculations were used in place of experimental data. As eluded to in the previous section, collinear spin DFT was selected. Although a smaller number of k-points were used, the calculation lasted for approximately 30 days from start to end.

Rather than calculate the input files manually, a program was created to automate this process (QE_EOS). It was designed to cache input and output files for the DFT program, PWscf, such that it would load an output file from the cache if it had already successfully run, to reduce the calculation time.

The relaxed (energy minimised) shape of the antiferromagnetic FCC Fe was not cubic, but orthorhombic. The equation of state calculation is specifically for a cubic crystal, but the elastic constants were calculated for the orthorhombic crystal. From the 9 calculated elastic constants the melting point, bulk modulus, shear modulus and Young's modulus were calculated (section ?? and appendix ??). The resulting structure was stable and were well within a magnitude of the known values for BCC Fe. These data are an original contribution to science.

11.3.4 Palladium-Iron Configurations as Fitting Data

To be able to fit the Fe-Pd potentials, the bulk properties of both pure FCC Fe and FCC Pd were either calculated or taken from experimental values. The potentials were then trained to fit this data. As well as this, a number of configurations of Fe, Pd and Fe-Pd were generated with their atoms slightly displaced from their lattice positions. This would give a wider range of atom separations and force values for the fitting program to use.

The force data were entirely from first-principles calculations, and ideally there would have been more configurations and computed with a higher number of k-points. In addition, a wider range of vacancies, interstitials and mixing concentrations would have been desirable. The atomic percentage on Pd or Ru doped is half a percent which would be better represented with a one PGM atom in a 4x4x4 FCC 256 atom supercell rather than the 32 atom cells used, but again this was a constraint imposed by computational resources.

11.3.5 Iron-Palladium Potential

The first derived Fe-Pd potential reproduces the properties of the material with a minimum error of 0.0% for the cohesive energy for both Fe and Pd, and a maximum error of 8% for the bulk modulus of Fe.

The cohesive energy and surface energy plots have a number of irregular bumps (fig ??), and it is hard to say whether or not these are correct without also computing them using DFT.

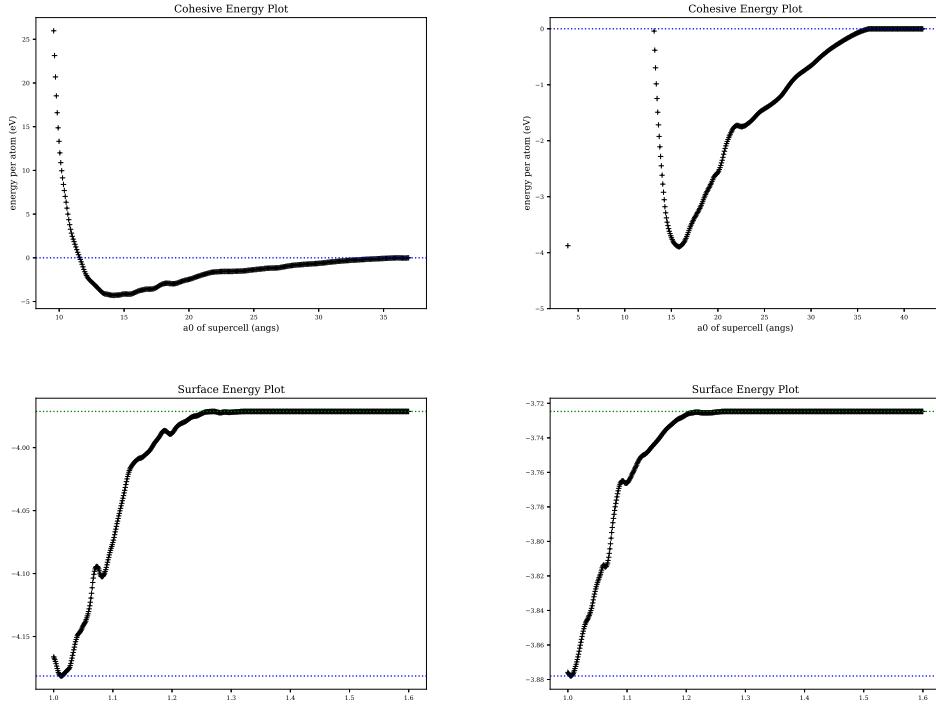


Figure 11.2: Cohesive energy and surface energy plots - detailed plots in the appendix ?? and ??

The second derived Fe-Pd potential reproduces the properties of the material with a minimum error of 0.0% for the cohesive energy of Pd and the C_{22} elastic constant for Fe, and a maximum error of 4.8% for the bulk modulus of Fe.

Despite having a better fit to the properties of the material used, the cohesive energy and surface energy plots are less well behaved than those in the first version (fig. ??). The optimisation algorithms blindly fit potentials to the data available, and it is apparent that more DFT computed energies are required for a variety of configurations, including splitting the bulk into two surfaces and expanding atoms from bulk into isolated atoms.

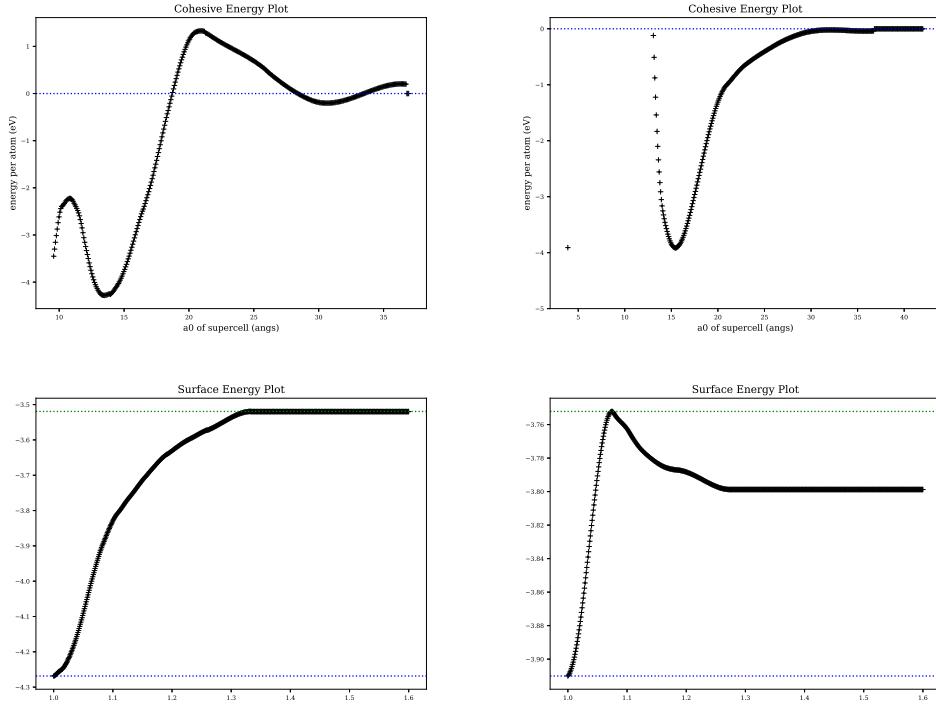


Figure 11.3: Cohesive energy and surface energy plots - detailed plots in the appendix ?? and ??

The non-cubic geometry of antiferromagnetic FCC Fe also suggests that the energy surface should be considered in all three planes, both with DFT and in the fitting process.

11.3.6 Original Contribution

The fitting code is an original contribution as it allows the fitting of metal potentials in the form of the embedded-atom method and two band embedded-atom method to DFT data and bulk properties. Other similar codes do exist, such as FitPot and PotFit, but at the time of developing my code the PotFit did not have the ability to fit analytic potentials (although, this feature is now available) and does not have, in its available form, the ability to fit the potential to the bulk properties of the material (although the Sheng version did have this ability). The code developed here may be modified in the future to add surface energy and point defect energies as data to fit to. It would require the addition of a geometry relaxation subroutine to optimise the atom locations as the basis vectors change.

11.4 Two-Band EAM Contribution to DL-Poly

During the early stages of this work it was clear that the Embedded Atom Method was a good candidate for the type of potential to be derived as it was well suited to modelling metals and had been used previously (along with the similar FS potential) to model radiation damage[25]. Work by Prof. Ackland at the University of Edinburgh considered using a separate density function and embedding functional to represent the S-Band and D-Band of transition elements such as Caesium. Work by P. Olsson and J. Wallenius also used this two band version to model Fe-Cr with a later Fe-Cr potential being developed by Bonny et al to replicate the mixing enthalpy as a function of Cr content.

After meeting with Prof. Todorov at Daresbury Laboratory, the source code for DL-Poly was modified to include new keywords, additional arrays to store the two-band density and embedding data for the second band

and modifications to the energy and force subroutines. This was then released with version 4.05 of DL_POLY in July 2013.

Ultimately, only EAM potentials were fit in this work. In the future, as more elements are added to the potential, in particular Cr, the 2BEAM may be required. The molecular dynamics code and fitting code both have the capability to use this type of potential.

Chapter 12

Future Work

At a certain point, work on various codes developed in this work, as well as the resulting Fe-Pd potential(s) had to stop, but there are a number of improvements that could be made given more time.

12.1 Activity Code

12.1.1 Experimental Activity Readings for Ion Irradiated Targets

The Activity code was developed using data from the TENDL data file. To test the validity of the code, an Iron sample was irradiated and its activity was measured several days after irradiation.

To validate the code further, a range of pure targets and alloys would be irradiated by a cyclotron, with targets of varying thickness and beams of varying fluences and energies. This would generate data to be used to compare to the results predicted by the code.

12.1.2 Code Improvements

Ideally, a built in ion transport code similar to SRIM would be developed. Failing this, an improvement in the reaction rate calculation and a rewrite of the code in Python with F2PY to improve ease of developing in the future.

Given the variation of energy ranges between cross section databases, it would be useful to give the user a selection of a variety of TENDL and similar data files to choose from. In addition to this, reaction cross sections could be computed over a set energy range using the TALYS code specifically for use in the Activity program.

Finally, extending the range of projectiles in the Activity code to include Deuterons, Tritons and Helium would be useful as the cyclotron is capable of accelerating these particles as well as protons. It would require an extension of the database to include the relevant cross section files.

12.2 Potential Fitting and MD Simulations

Two potentials were derived for Fe-Pd that reproduced the bulk properties of each element individually reasonably well. Cohesive energy and surface energy plots could be improved by adding new modules to help fit the potential, and overall more DFT data could be collected.

12.2.1 Larger Supercells

In many examples in the literature, the supercell sizes were at least 4x4x4 containing 256 atoms for FCC crystals. These require much more memory and processing power, especially considering the lack of symmetry and inclusion of magnetism in the calculations. More calculations over a greater range of randomised atom locations, with a varying percentage of palladium (1 atom per 256, 2 atoms per 256, 3 atoms per 256 and so on) would also help towards fitting a better potential.

12.2.2 Properties of Fe-Pd Alloy

The properties of palladium were known and those of pure FCC iron were calculated. Using DFT to calculate the properties of a 1-2% iron-palladium alloy would provide another set of data to fit the potentials to.

12.2.3 Improved Fitting: Surface Energy, Cohesive Energy and Defects

The cohesive energy and surface energy plots for the derived iron and palladium potentials were recognisable, but they were not smooth and did have many small bumps in the plots. Having plotted the same for Aluminium, the DFT results were much smoother.

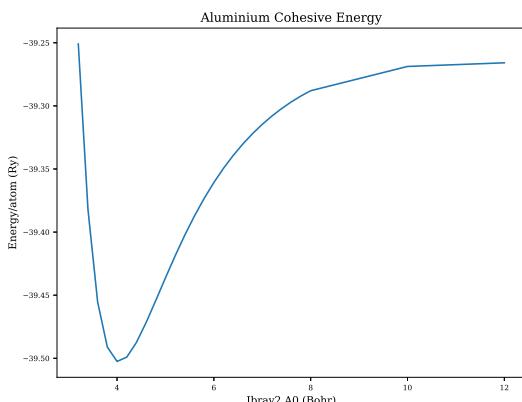


Figure 12.1: DFT Aluminium Cohesive Energy

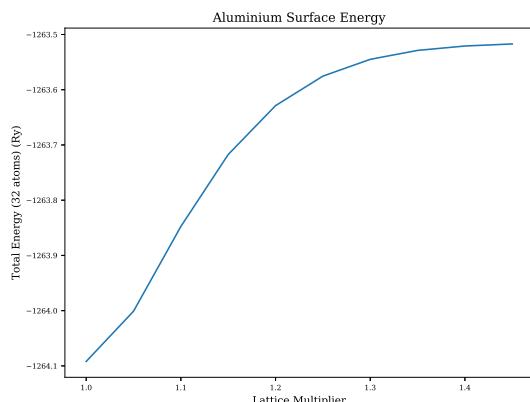


Figure 12.2: DFT Aluminium Surface Energy

Rather than just fit the data to the energy surface, it would be useful to create DFT configurations for the cohesive energy and surface plots for both Iron and Palladium. These configurations, and the energies, forces and stresses computed by DFT would be used to improve the fitting process.

Currently the fitting code has two modules: EFS, to compute energy, forces and stress of a set of atom configurations provided and BP, to calculate the material properties (bulk modulus, elastic constants etc). A new module could be created to calculate the value of the surface energy. It would require computing the relaxed energy of the potential for the bulk material. Next a slab, with enough vacuum along its surface(s) to prevent interference over the periodic boundary condition, would have the atomic positions relaxed and the energy (for the potential) calculated. This would allow computation of the value of the surface energy to be used to improve the potential.

The computation of the cohesive energy could also be implemented into a module. The relaxed bulk energy would be computed as well as that of isolated atoms, and this would give enough data to compute the cohesive energy.

Finally, a defects module could be created. Common defects, and their experimental or DFT computed energies, would be provided by the user. The module would create the configurations based on the user's input. The

volume and positions of the atoms would need to be relaxed with the current iteration of the potential for each defect, and the energy computed. This, with the energy in bulk, would compute the energy of each defect.

12.2.4 Improved Potential: Non-Isotropic Structure

With a collinear spin-polarized DFT calculation, the relaxed structure was slightly longer in the y direction. Investigating the use of an angularly dependent potential might be worthwhile and the MEAM type potential with an angularly dependent electron density would be a starting point (section 5.3.3).

12.2.5 Improved Potential: Fe-Ni-Cr-Pd

Given more time and a larger computer, more complicated DFT calculations could be performed that better represent the material, austenitic stainless steel. 1% palladium doped steel could be represented as a 4x4x4 supercell containing 256 atoms. For 304SS approximately 50 chromium, 25 nickel, 178 iron and 3 palladium atoms would be needed (and small variations on these figures).

Acronyms

2BEAM two band embedded-atom method. 69, 80, 136, 143

ABWR Advanced Boiling Water Reactor. 4, 7

AGR Advanced Gas Reactor. 3–5, 8, 9, 36

ASME American Society of Engineers. 27, 28

BCC body centered cubic. 14, 26–28, 34, 59, 60, 69, 70, 76, 86, 92, 137, 149

BFGS Broyden Fletcher Goldfarb Shanno. 54, 109

BWR Boiling Water Reactor. 3, 11, 29, 36

BZ Brillouin zone. 92, 102, 103

CANDU Canada Deuterium Uranium. 3

CCGT Combined Cycle Gas Turbine. 6, 10, 11

CERN Conseil Europen pour la Recherche Nuclaire. 41

CPH face centered tetragonal. 86

DFT Density Functional Theory. i, 16, 55, 60, 62, 77, 86, 87, 97, 101–103, 105, 107, 109, 136, 137, 140, 145–147, 150

DNA Deoxyribonucleic Acid. 17

DPA displacements per atom. i, 14, 29, 31, 32, 135

EAM embedded-atom method. 15, 16, 54, 68–70, 76, 80, 136

EBR Experimental Breeder Reactor. 31

ECIS Equations Couples Itrations Squentielle. 47

ECP electrochemical corrosion potential. 36, 37

ELSY European Lead Fast Reactor. 9

ENDF Evaluated Nuclear Data File. 49, 51

EPR European Pressurised Reactor. 7

FCC face centered cubic. i, 26–28, 34, 54, 59, 60, 63, 69, 86, 92, 145

FS Finnis-Sinclair. 16, 54, 136

Gen II Generation II. 3, 4, 29, 31

Gen II+ Generation II+. 4

Gen III Generation III. 4

Gen III+ Generation III+. i, 1, 4, 7, 13, 37

Gen IV Generation IV. i, 1, 2, 4, 8, 9, 11, 13, 29, 37, 44, 113, 135

GFR Gas-Cooled Fast Reactor. 11, 37

GGA generalized gradient approximation. 99–102, 146, 147

GPU graphical processing unit. 105

HFIR High Flux Isotope Reactor. 41, 42, 44

HK Hohenberg-Kohn. 93–95, 102

HRB Hardness Rockwell B. 28

IASCC irradiation assisted stress corrosion cracking. 34

IBZ irreducible Brillouin zone. 92, 102

IGSCC inter granular stress corrosion cracking. 1, 26, 28, 32, 34–37

IKE inverse Kirkendall effect. 31

ITER International Thermonuclear Experimental Reactor. 12

JEFF Joint Evaluated Fission and Fusion File. 50, 51

KE Kirkendall effect. 31

KS Kohn-Sham. 98

LAMMPS Large-scale Atomic/Molecular Massively Parallel Simulator. 73, 105, 107

LDA local density approximation. 98–101, 146

LFR Lead-Cooled Fast Reactor. 9, 11

LHC Large Hadron Collider. 41

LINAC linear accelerator. 40, 43

LMA Levenberg-Marquardt Algorithm. 109, 111

LSDA local spin density approximation. 99–102, 146, 147

LWR Light Water Reactor. 36

MD molecular dynamics. i, 55, 76, 105, 107, 136

MEAM modified embedded-atom method. 72

MOX mixed oxide. 106

- MSR** Molten Salt Reactor. 12
- NAA** neutron activation analysis. 54
- NBS** National Bureau of Standards. 42
- NBSR** National Bureau of Standards Reactor. 43
- NDS** Nuclear Data Services. 46
- NEA** Nuclear Energy Agency. 51
- NG** Newton-Gauss. 111, 112
- ORNL** Oak Ridge National Laboratory. 41, 43
- PBE** Perdew-Burke-Ernzerhof. 99–101
- PBESOL** Perdew-Burke-Ernzerhof functional revised for solids. 101
- PBR** Pebble-Bed Reactor. 10
- PGM** platinum group metal. i, 1, 37–39
- PKA** primary knock-on atom. 13–15, 24, 25, 29, 31, 76, 105–107
- PWR** Pressurised Water Reactor. 3–5, 7, 11, 12, 29, 36, 37
- PZ** Perdew-Zunger. 101
- RBMK** Reaktor Bolshoy Moshchnosti Kanalnyy. 3
- RIS** radiation induced segregation. 31, 32, 37
- SCC** stress corrosion cracking. 31, 36, 38
- SCF** self consistent field. 104, 141, 150
- SCWR** Supercritical-Water-Cooled Reactor. 9, 11, 12
- SEM** scanning electron microscope. 54
- SFR** Sodium-Cooled Fast Reactor. 9, 11, 12, 37
- SIMS** secondary-ion mass spectrometry. 54
- SLAC** Stanford Linear Accelerator Center. 40
- SNS** Spallation Neutron Source. 40, 43, 44
- SRIM** Stopping Range In Matter. 40, 46, 51, 53, 67, 113, 119, 120, 135
- TEM** transmission electron microscope. 38, 54
- TENDL** TALYS-based Evaluated Nuclear Data Library. 39, 40, 46, 49, 51, 125, 127, 130, 133
- TGSCC** trans granular stress corrosion cracking. 35
- TISE** Time Independent Schrödinger Equation. 86, 88–90, 93, 97, 101
- TRIM** TRansport In Matter. 51–53

TRISO Tristructural Isotropic. 12

TRIUMF TRI-University Meson Facility. 40

VHTR Very-High-Temperature Reactor. 10, 11

VPI vacancies per ion. 135

VVER Vodo-Vodyanoi Energetichesky Reaktor. 3

XC exchange-correlation. 97–99, 101

ZBL Ziegler-Biersack-Littmark. 15, 67, 68, 75, 76

Glossary

304SS Austenitic stainless steel: 18-20%Cr, 8-11%Ni. 28, 31, 33, 37, 38

316SS Austenitic stainless steel: 16.5-18.5%Cr, 10-13%Ni, 2.0-2.5%Mo. 28, 34, 37

allotrope an element that exists in multiple forms/structures - the common example of allotropes are graphite (hexagonal sheets) and diamond (regular tetrahedral structure) for carbon. 27

antiferromagnetic neighbouring electrons orientate in opposite directions, cancelling out overall magnetism. 72

Bloch theorem electron wave functions may be written as a sum of plane waves $\psi_{\vec{k},n} \vec{r} = \sum c_{\vec{k}+\vec{G},n} \exp(i(\vec{k} + \vec{G})\vec{r})$. 92

Brillouin Zone the Wigner-Seitz cell in reciprocal space. 92

Cauchy pressure $C_{12} - C_{44}$. 72

Co Cobalt $Z = 27$, $A_r = 58.93$. 1

Cr Chromium $Z = 24$, $A_r = 52.00$. 1

Cu Copper $Z = 29$, $A_r = 63.55$. 31

enthalpy sum of internal energy plus pressure times volume $H = E_{internal} + PV$. 71

eutectic A eutectic contains two or more elements, and has a lower melting point than any of it's constituent elements.. 9

Fe Iron $Z = 26$, $A_r = 55.85$. 1

Fermi energy the energy of the highest occupied state. 92

fermion half-integer spin particle - includes electrons, protons, neutrons. 96

fissile Fissile materials can undergo a nuclear chain reaction with slow (thermal) neutrons. 2

Hamiltonian the total energy of a system - an operator in quantum mechanics, $\hat{H} = \hat{T} + \hat{V} = \hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r},t)$. 86

homogeneous electron gas quantum mechanical model of electrons distributed uniformly through space (jellium). 98

invar effect invar is an Fe Ni (64/36) alloy that has a low thermal expansion coefficient for a range of temperatures below its curie temperature, and this low expansion is the invar effect. 100

Ir Iridium $Z = 77$, $A_r = 192.22$. 38

isostructural transformation a change from one structure to another. 70

jellium quantum mechanical model of electrons distributed uniformly through space (homogeneous electron gas). 95, 98

Kohn-Sham equations $\hat{v}_{KS}(\vec{r}_e, \vec{r}_n) = v_{n-e}(\vec{r}_e, \vec{r}_n) + \int d^3r' \frac{\rho(r')}{|\vec{r}_e - \vec{r}'|} + v_{xc}[\rho](\vec{r}_e)$. 95, 96

matrix diagonalization Take matrix A and find a diagonal matrix D such that $S^{-1}AS = D$. 102

Mg Carbon $Z = 6$, $A_r = 12.00$. 26

Mn Maganese $Z = 25$, $A_r = 54.94$. 3

Mo Molybdenum $Z = 42$, $A_r = 95.95$. 2

Neel temperature temperature at which an antiferromagnetic becomes a paramagnet. 72

orthorhombic $a, b, c, \alpha = \beta = \gamma = 90^\circ$. 56

P Phosphorus $Z = 15$, $A_r = 30.97$. 31

Pauli exclusion principle half-integer spin particles cannot have the same quantum numbers. 96

Pd Palladium $Z = 46$, $A_r = 106.42$. 2, 38

plane wave a wave of parallel planes of constant frequency that are normal to the direction of propagation. 92

proof strength stress at which a material undergoes a permanent extension. 28

Pt Platinum $Z = 78$, $A_r = 195.08$. 38

Pu Plutonium $Z = 94$, $A_r = 244$. 3

Ru Ruthenium $Z = 44$, $A_r = 101.07$. 38

sensitization The precipitation of chromium rich carbides at the grain boundaries of stainless steel. 33

supercritical Above $374^\circ C$ (647K) and 22.1MPa water is no longer a distinct liquid or gas and it becomes supercritical. 11

Th Thorium $Z = 90$, $A_r = 232.04$. 2, 12

U Uranium $Z = 92$, $A_r = 238.03$. 2

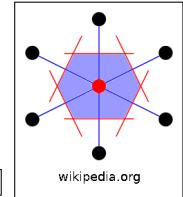
ultimate tensile strength maximum stress the material can withstand. 28

valence electron outer shell electron that can form bonds. 98

W Tungsten $Z = 74$, $A_r = 183.84$. 14

wave function contains all there is to know about a system and it's meaning is given by a probability $|\Psi|^2 = 1$.

Wigner-Seitz cell given a lattice point, the set of all points in space which are closer to that lattice point



that any other lattice point constitute the Wigner-Seitz cell [solidstatebasicswcs]. 92

Young's modulus measure of resistance to elastic deformation $\delta = \frac{\text{stress}}{\text{strain}}$. 28, 64

Zr Zirconium $Z = 40$, $A_r = 91.22$. 7

Bibliography

- [1] D. W. Saxy et al. “Understanding Stress Corrosion Resistant Stainless Steels at the Atomic Scale”. In: (2010).
- [2] World Nuclear. *Advanced Nuclear Power Reactors*. 2020. URL: <https://world-nuclear.org/information-library/nuclear-fuel-cycle/nuclear-power-reactors/advanced-nuclear-power-reactors.aspx>.
- [3] Thomas W. Kerlin and Belle R. Upadhyaya. *Dynamics and Control of Nuclear Reactors*. Vol. 2. 2019.
- [4] H. Khatabil. *Gen-4 Reactors*. 2013. URL: <http://www.gen-4.org/GIF/About/documents/25-Session2-3-Khartabil.pdf>.
- [5] Financial Times. *Cost of new Sizewell C Nuclear Plant*. 2020. URL: <https://www.ft.com/content/77c209f7-6d18-4609-ac3c-77d1b5b82b34>.
- [6] Energy for Generations. *ESB opens 820m Euro Carrington Power Station in Manchester*. 2020. URL: <https://www.esb.ie/tns/press-centre/2017/2017/03/13/esb-opens-820m-carrington-power-station-in-manchester>.
- [7] Our World In Data. *Safest and Cleanest Sources of Energy*. 2020. URL: <https://ourworldindata.org/uploads/2020/10/What-is-the-safest-form-of-energy-2048x1129.png>.
- [8] MIT Open Courseware. *Engineering of Nuclear Systems - Lecture 6A*. 2013. URL: <http://ocw.mit.edu/courses/nuclear-engineering/22-06-engineering-of-nuclear-systems-fall-2010/lectures-and-readings/MIT22\%5F06F10\%5Flec06a.pdf>.
- [9] Yuchen Dou, Hong Luo, and Jing Zhang. “Elastic Properties of FeCr20 Ni8Xn (X = Mo, Nb, Ta, Ti, V, W and Zr) Austenitic Stainless Steels: A First Principles Study”. In: *Metals* 9 (145 2019).
- [10] Philippe F Weck et al. *Mechanical properties of zirconium alloys and zirconium hydrides predicted from density functional perturbation theory*. 2020. URL: <https://www.osti.gov/pages/servlets/purl/1259850>.
- [11] GenIV International Forum. *Introduction to Generation IV Nuclear Energy Systems and the International Forum*. 2013. URL: <https://www.iea.org/topics/world-energy-outlook>.
- [12] Wikipedia. *Carnot's theorem (thermodynamics)*. 2020. URL: [https://en.wikipedia.org/wiki/Carnot\%27s\%5Ftheorem\%5F\(thermodynamics\)](https://en.wikipedia.org/wiki/Carnot\%27s\%5Ftheorem\%5F(thermodynamics)).
- [13] S. E. Jensen and E. Nonbol. *Description of the Magnox Type of Gas Cooled Reactor (MAGNOX)*. 1998. URL: <https://inis.iaea.org/collection/NCLCollectionStore/\%5FPublic/30/052/30052480.pdf>.
- [14] Wikipedia. *Combined cycle power plant*. 2020. URL: <https://en.wikipedia.org/wiki/Combined\%5Fcycle\%5Fpower\%5Fplant>.
- [15] World Nuclear. *Uranium Enrichment*. 2020. URL: <https://www.world-nuclear.org/information-library/nuclear-fuel-cycle/conversion-enrichment-and-fabrication/uranium-enrichment.aspx>.

- [16] Los Alamos National Laboratory. *Los Alamos National Laboratory Tour*. 2013. URL: <http://t2.lanl.gov/nis/tour/sch007.html>.
- [17] GenIV International Forum. *Very-High-Temperature Reactor (VHTR)*. 2020. URL: <https://www.gen-4.org/gif/jcms/c%5F9362/vhtr>.
- [18] Sangeeta Deokattey et al. “Hydrogen production using high temperature reactors: an overview”. In: *Advances in Energy Research* 1 (1 2013), pp. 013–033.
- [19] M Müller and P. U. Foscolo. *Advanced Biomass Gasification - New Concepts for Efficiency Increase and Product Flexibility*. 2016.
- [20] MIT Open Courseware. *Engineering of Nuclear Systems - Lecture 6B*. 2013. URL: <http://ocw.mit.edu/courses/nuclear-engineering/22-06-engineering-of-nuclear-systems-fall-2010/lectures-and-readings/MIT22\%5F06F10\%5Flec06b.pdf>.
- [21] V. Ignatiev and A. Surenkov. *Corrosion phenomena induced by molten salts in Generation IV nuclear reactors*. 2017, p. 166.
- [22] C. Vaglio and J-F. Vidal. *The JEFF-3.1.1 Nuclear Data Library*. 2009.
- [23] M. R. Gilbert, J. Marian, and J-Ch. Sublet. “Energy spectra of primary knock-on atoms under neutron irradiation”. In: *Journal of Nuclear Materials* 467 (2015), pp. 121–134.
- [24] J. Fikar and R. Schäublin Ecole. “Molecular dynamics simulation of radiation damage in bcc tungsten”. In: *Journal of Nuclear Materials* (101 2009), pp. 97–101.
- [25] Andrew F. Calder et al. “Computer simulation of cascade damage in a-iron with carbon in solution”. In: *Journal of Nuclear Materials* (382 2008), pp. 91–95.
- [26] N J Carron. *An Introduction to the Passage of Energetic Particles through Matter*. 2007.
- [27] James Byrne. *Neutrons, Nuclei & Matter*. 2011, p. 161.
- [28] J. Rodenas and G. Verdu. “Analysis of nuclear reactions to determine the radionuclides generated and its activity in various devices”. In: *Radiation Physics and Chemistry* 167 (2020), pp. 1–6.
- [29] H. Sheng. *EAM Potentials*. 2012. URL: <https://sites.google.com/site/eampotentials>.
- [30] G. S. Was. *Fundamentals of Radiation Materials Science*. 2007.
- [31] R. E. Stoller. “Primary Damage Formation in Irradiated Materials”. In: *Journal of Metals* 48 (12 1996), pp. 23–27.
- [32] Jian wei Su and Dong ping Huang. “Phase equilibria of the FeCrNi ternary systems and interfacial-reactions in FeCr alloys with Ni substrate”. In: *Journal of Alloys and Compounds* 457 (2008), pp. 270–278.
- [33] S. Choudhury a et al. “Ab-initio based modelling of diffusion in dilute bcc FeNi and FeCr alloys and implications for radiation induced segregation”. In: *Journal of Materials* 411 (2011), pp. 1–14.
- [34] J. L. Seran and M. Le Flem. *Irradiation-resistant Austenitic Steels as Core Materials for Generation IV Nuclear Reactors*. 2017, pp. 285–328.
- [35] T.R. Allen and J.T. Busby. “Radiation damage concerns for extended light water Reactor service”. In: *Journal of Materials* 61 (2009), pp. 29–34.
- [36] Gary S. Was and Shigeharu Ukai. *Austenitic Stainless Steels*. 2019.
- [37] T.R. Allen et al. “Swelling and radiation-induced segregation in austenitic alloys”. In: *Journal of Nuclear Materials* 342 (2005), pp. 90–100.
- [38] T. R. Allen, J. I. Cole, and E. A. Kenik. *Radiation-Induced Segregation and Void Swelling in 304 Stainless Steel*. 2000. URL: <https://www.osti.gov/servlets/purl/757550>.

- [39] A. Janotti and R.C. Reed. “Diffusion rates of 3d transition metal solutes in nickel by first-principles calculations”. In: *Acta Materialia* 53 (2005), pp. 2369–2376.
- [40] IAEA. 2019. URL: <https://inis.iaea.org/collection/NCLCollectionStore/\%5FPublic/29/010/29010110.pdf>.
- [41] G. S. Was. *Fundamentals of Radiation Materials Science*. 2007, p. 183.
- [42] B. Kombaiah et al. “Mechanisms of radiation-induced segregation around He bubbles in a Fe-Cr-Ni crystal”. In: *Journal of Nuclear Materials* 11 (2018).
- [43] W G Johnston, W G Morris, and A M Turkalo. “Proc. Int. Con. on Radiation Effects in Breeder Reactor Structural Materials”. In: *Metall. Soc. AIME* (1977), p. 421.
- [44] A D Marwick. “Segregation in irradiated alloys: The Inverse Kirkendall Effect and the effect of constitution on void swelling”. In: *J. Phys. F: Met. Phys* 8 (1978), pp. 1849–1862.
- [45] P. R. Okamoto and L. E. Rehn. “RADIATION-INDUCED SEGREGATION IN BINARY AND TERNARY ALLOYS”. In: *Journal of Nuclear Materials* 83 (1979), pp. 2–23.
- [46] Jeremy L. Gilber. *Metals - Basic Principles*.
- [47] S. Hoffmann and H. Hofsass. “Compositional Changes of Passive Films Due To Different Transport Rates and Preferential Dissolution”. In: *Corrosion Science* 31 (1990), pp. 573–578.
- [48] H. Knote and U. Stolz. “The Passivity of Iron-Chromium Alloys”. In: *Corrosion Science* 29 (7 1989), pp. 899–917.
- [49] P. J. Maziasz and J. T. Busby. *Properties of Austenitic Steels for Nuclear Reactor Applications*. Vol. 2. 2012, pp. 267–283.
- [50] Rolled Alloys. *What is alloy sensitization?* 2020. URL: <https://www.rolledalloys.com/dotAsset/50dc766c-08f5-43d9-bbda-5d9a9a2fa6d5.jpg>.
- [51] Speciality Steel Industry of North America. *Intergranular Corrosion*. 2020. URL: <https://www.ssina.com/education/corrosion/intergranular-corrosion/>.
- [52] S. Lyon. *Corrosion of Molybdenum and its Alloys*. 2010. URL: https://www.researchgate.net/publication/287308921_Corrosion_of_Molybdenum_and_its_Alloys.
- [53] V. Maurice et al. “Effects of molybdenum on the composition and nanoscale morphology of passivated austenitic stainless steel surfaces”. In: *Faraday Discussions* 180 (2015), pp. 151–170.
- [54] G. S. Was. *Fundamentals of Radiation Materials Science*. 2007, p. 765.
- [55] Yuchen Dou, Hong Luo, and Jing Zhang. “Elastic Properties of FeCr₂₀ Ni₈X_n (X = Mo, Nb, Ta, Ti, V, W and Zr) Austenitic Stainless Steels: A First Principles Study”. In: *Metals* 9 (145 2019).
- [56] Sophie Le Car. *Water Radiolysis: Influence of Oxide Surfaces on H₂ Production under Ionizing Radiation*. 2011. URL: <https://www.mdpi.com/2073-4441/3/1/235>.
- [57] R.W. Staehle. *Historical views on stress corrosion cracking of nickel-based alloys: the Coriou effect*. 2016.
- [58] United States Nuclear Regulatory Commission. *AP1000 Design Control Document*. 2020. URL: <https://www.nrc.gov/docs/ML1117/ML11171A447.pdf>.
- [59] P. N. Standing. *The Long Term Storage of Advanced Gas-Cooled Reactor Fuel*. URL: <https://inis.iaea.org/collection/NCLCollectionStore/\%5FPublic/30/040/30040095.pdf>.
- [60] Guy O.H. Whillock et al. “Investigation of thermally sensitised stainless steels as analogues for spent AGR fuel cladding to test a corrosion inhibitor for intergranular stress corrosion cracking”. In: *Journal of Nuclear Materials journal* (498 2018), pp. 187–198.
- [61] IAEA. *Status report 81 - Advanced Passive PWR (AP 1000)*. 2020. URL: <https://aris.iaea.org/PDF/AP1000.pdf>.

- [62] Inc Framatome ANP. *EPR Design Description*. 2020. URL: <https://www.nrc.gov/docs/ML0522/ML052280170.pdf>.
- [63] F. Dalle et al. *Conventional austenitic steels as out-of-core materials for Generation IV nuclear reactors*.
- [64] Pascal V. Grundler and Stefan Ritter. “Nobel Metal Chemical Addition for Mitigation of Stress Corrosion Cracking: Theoretical Insights and Applications”. In: *PPChem* (2014), pp. 76–93.
- [65] J. H. Potgieter and A. Van Benekom. “The effect of varying ruthenium content on the corrosion behavior of two cathodically modified superferritic stainless steels”. In: *Canadian Metallurgical Quarterly* 34 (2 1994), pp. 143–146.
- [66] IAEA Department of Nuclear Sciences and Applications. *Directory of Cyclotrons used for Radionuclide Production in Member States*. 2006. URL: <http://www-naweb.iaea.org/napc/iachem/cyclotrons/PDF/DCRP.pdf>.
- [67] Diamond. *About Synchrotrons*. 2020. URL: <https://www.diamond.ac.uk/Home/About/FAQs/About-Synchrotrons.html>.
- [68] World Nuclear. 2018. URL: <http://www.world-nuclear.org/information-library/non-power-nuclear-applications/radioisotopes-research/research-reactors.aspx>.
- [69] IAEA. 2018. URL: <https://nucleus.iaea.org/RRDB/RR/ReactorSearch.aspx>.
- [70] ORNL. 2018. URL: <https://neutrons.ornl.gov/sites/default/files/HighFluxIsotopeReactorUserGuide2.0.pdf>.
- [71] ORNL. 2018. URL: <https://neutrons.ornl.gov/hfir/parameters>.
- [72] ORNL. *High Flux Isotope Reactor (HFIR) USER GUIDE*. 2015. URL: <https://neutrons.ornl.gov/sites/default/files/High\%20Flux\%20Isotope\%20Reactor\%20User\%20Guide\%202.0.pdf>.
- [73] John J. Rush and Ronald L. Cappelletti. *The NIST Center for Neutron Research: Over 40 Years Serving NIST/NBS and the Nation*. 2015. URL: https://www.ncnr.nist.gov/NCNRHistory_Rush_Cappelletti.pdf.
- [74] ISIS. *How ISIS works - In Depth*. 2020. URL: <https://www.isis.stfc.ac.uk/Pages/How-ISIS-works--in-depth.aspx>.
- [75] E. B. Iverson. *Neutron Moderation*. 2016. URL: <https://conference.sns.gov/event/56/attachments/64/99/Lecture\%5F2a\%5F-\%5FNeutron\%5FModeration\%5F-\%5FErik\%5FIverson.pdf>.
- [76] M. N. H. Comsan. *Spallation Neutron Sources for Science and Technology*. 2011. URL: <https://inis.iaea.org/collection/NCLCollectionStore/\%5FPublic/43/099/43099436.pdf>.
- [77] R. B. Leachman. *Peaceful Uses of Atomic Energy*. Vol. 2. 1956, p. 193.
- [78] P E Hodgson. “The Nuclear Optical Model”. In: *Rep. Prog. Phys.* 34 (1971), pp. 765–819.
- [79] A Koning, S Hilaire, and S Goriely. *TALYS 1.95 A Nuclear Reaction Program*. 2019.
- [80] NEA. *Janis Book*. 2020. URL: <https://tendl.web.psi.ch/tendl\%5F2019/other/book-protons.pdf>.
- [81] NEA. 2020. URL: <https://www-nds.iaea.org/exfor/>.
- [82] A Koning, S Hilaire, and S Goriely. *The Stopping and Range of Ions in Matter*. 2015.
- [83] M. D. Ziegler and J. P. Biersack. “SRIM - The stopping and range of ions in matter”. In: *Nuclear Instruments and Methods in Physics Research Section B* 268 (2010), pp. 1818–1823.
- [84] Roger Rousseau, Vassiliki-Alexandra Glezakou, and Annabella Selloni. “Theoretical insights into the surface physics and chemistry of redox-active oxides”. In: *Nature Reviews Materials* (5 2020), pp. 460–475. URL: <https://www.nature.com/articles/s41578-020-0198-9>.
- [85] F. D. Murnaghan. “The Compressibility of Media Under Extreme Pressures”. In: (1944).

- [86] A. Mahmoud ad A. Erba. *Crystal: Equation of State*. 2019. URL: <http://tutorials.crystalsolutions.eu/tutorial.html?td=eos\&tf=eos\%5Ftut>.
- [87] F. Birch. “Finite elastic strain of cubic crystals”. In: (1947).
- [88] gilgamesh. *Fitting Birch-Murnaghan*. 2013. URL: <http://gilgamesh.cheme.cmu.edu/doc/software/jacapo/appendices/appendix-eos.html>.
- [89] Michael J. Mehl. “Pressure depenence of the elastic moduli in aluminium-rich Al-Li compounds”. In: *Physical Review B* 47 (5 1993), pp. 2493–2500.
- [90] P. A. Korzhavyi and B. Johansson. “Density functional theory for calculation of elastic properties of orthorhombic crystals - applications to TiSi₂”. In: *Journal of Applied Physics* 84 (9 1998), pp. 4891–4904.
- [91] B. M. Klein and D. A. Papaconstantopoulos. *First Principles Calculations of Elastic Properties of Metals*. 1994.
- [92] B B Karki, G J Ackland, and J Crain. *Elastic instabilities in crystals from ab initio stressstrain relations*. 2013. URL: <http://www.homepages.ed.ac.uk/gja/karki1.pdf>.
- [93] G. Ghosh. “A first-principles study of cementite (Fe₃C) and its alloyed counterparts: Elastic constants, elastic anisotropies, and isotropic elastic moduli”. In: *AIP Advances* 5 (2015).
- [94] L. D. Brown and H. L. Marcus. “Elastic Constants Versus Melting Temperature in Metals”. In: *Scripta Metallurgica* 18 (1984), pp. 951–956.
- [95] M. W. Finnis and J. E. Sinclair. “A simple empirical N-body potential for transition metals”. In: *Philosophical Magazine A* 50 (1984).
- [96] M. S. Daw and M. I. Baskes. “Embedded-atom method Derivation and application to impurities, surfaces, and other defects in metals”. In: *Physical Review B* 29 (12 1984), pp. 6443–6453.
- [97] Graeme J. Ackland. “Two-band second moment model for transition metals and alloys”. In: *Condensed Materials* (2005).
- [98] K. Nordlund and L. Malerba. “Two-band modelling of alpha prime phase formation in Fe-Cr”. In: *Physical Review B* 72 (2005).
- [99] G. Bonny et al. “Iron chromium potential to model high-chromium ferritic alloys”. In: *Philosophical Magazine* 91 (April 2011), pp. 1724–1746.
- [100] Jae-Hyeok Shim and M. I. Baskes. “Semiempirical atomic potentials for the fcc metals Cu, Ag, Au, Ni, Pd, Pt, Al, and Pb based on first and second nearest-neighbor modified embedded atom method”. In: *Physical Review B* 68 (2003).
- [101] J. Fikar and R. Schublin. “Molecular dynamics simulation of radiation damage in bcc tungsten”. In: *Journal of Nuclear Materials* 386 (2009).
- [102] H. Sheng et al. “Highly Optimized Embedded-Atom-Method Potentials for Fourteen FCC Metals”. In: *Physical Review B* 83 (2011), pp. 1–20.
- [103] P. Brommer and F. Gähler. “Potfit: effective potentials from ab initio data”. In: *Modelling Simul. Mater. Sci. Eng.* 15 (2007), pp. 295–304.
- [104] I.T.Todorov and W. Smith. *The DL_POLY_4 User Manual*. 2016.
- [105] Wikipedia. *Virial Stress*. 2016. URL: <https://en.wikipedia.org/wiki/Virial\%5Fstress>.
- [106] F. Ercolessi and J. B. Adams. “Interatomic Potentials from First-Principles Calculations: the Force-Matching Method”. In: *Europhys. Lett.* 26 (8 1994), pp. 583–588.
- [107] Steven H. Simon. *Solid State Basics*. 2013, p. 213.

- [108] Materials. *Microstructure and Mechanical Properties of Pressure-Quenched SS304 Stainless Steel*. 2019. URL: <https://www.mdpi.com/1996-1944/12/2/290/pdf#:~:text=This%20material%20displays%20atypical,size%20and%20irregular%20grain%20shape..>
- [109] Steven H. Simon. *Solid State Basics*. 2013.
- [110] MIT. *Physics for Solid State Applications*. 2004. URL: <http://web.mit.edu/6.730/www/ST04/Lectures/Lecture7.pdf>.
- [111] Harry Jones. *The Theory of Brillouin Zones and Electronic States in Crystals*. 1960, p. 48.
- [112] P. Hohenberg and W. Kohn. “Inhomogeneous Electron Gas”. In: *Physical Review* 136 (1964).
- [113] Robert van Leeuwen. *Introduction to density-functional theory*. 2020. URL: <http://users.jyu.fi/~roleeuwe/DFTLectureNotes.pdf>.
- [114] W. Kohn and L. J. Sham. “Self-Consistent Equations Including Exchange and Correlation Effects”. In: *Physical Review* 140 (1965).
- [115] SISSA. 2020. URL: <https://people.sissa.it/~degironc/ES/lectures/ABCofDFT.pdf>.
- [116] P. Ziesche, S. Jurth, and J. P. Perdew. “Density functionals from LDA to GGA”. In: *Computational Materials Science* 11 11 (1998), pp. 122–127.
- [117] M. C. Payne et al. “Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients”. In: *REVIEWS OF MODERN PHYSICS* 64 (1992).
- [118] Annabella Selloni. *The plane wave pseudopotential method (basic aspects)*. 2009. URL: https://th.fhi-berlin.mpg.de/th/Meetings/DFT-workshop-Berlin2009/Talks/OnlinePublication/0624-2_20090623-1_Selloni_FINAL_2perA4_-_Selloni-Berlin2.pdf.
- [119] V.P. Gupta. *PRINCIPLES AND APPLICATIONS OF QUANTUM CHEMISTRY - Density Functional Theory (DFT) and Time Dependent DFT (TDDFT)*. 2016, p. 169.
- [120] John P Perdew, Kieron Burke, and Matthias Ernzerhof. “Generalized Gradient Approximation Made Simple”. In: *Physical Review Letters* 77 (18 1996), pp. 3865–3868.
- [121] John P Perdew. “Generalized Gradient Approximations for exchange and correlation: A look backward and forward”. In: *Physica B* 172 (1991), pp. 1–6.
- [122] M R Benam, N Abdoshahi, and M Majidiyan Sarmazdeh. “Ab initio study of the effect of pressure on the structural and electronic properties of cubic LaAlO₃ by density function theory using GGA, LDA and PBESOL exchange correlation potentials”. In: *Physica B* 446 (2014), pp. 32–38.
- [123] periodictable.com. *Sodium: periodictable.com*. 2020. URL: <https://periodictable.com/Elements/011/data.html>.
- [124] periodictable.com. *Aluminium: periodictable.com*. 2020. URL: <https://periodictable.com/Elements/013/data.html>.
- [125] P Giannozzi et al. “Quantum Espresso”. In: *J Phys Condens Matter* 29 (2017).
- [126] University of Illinois. *Density functional theory calculation of elastic constants*. 2020. URL: <https://courses.physics.illinois.edu/mse404ela/sp2018/Project2-DFT.html>.
- [127] Alessandro Genova and Michele Pavanello. “Exploiting the Locality of Periodic Subsystem Density-Functional Theory: Efficient Sampling of the Brillouin Zone”. In: *J. Phys.: Condens. Matter* 27 (2015), pp. 495–501. URL: <https://iopscience.iop.org/article/10.1088/0953-8984/27/49/495501/ampdf>.
- [128] N. Marzari. *Ab-initio Molecular Dynamics for Metallic Systems*. 1996, p. 77.
- [129] *Electron band gap diagram for insulators, semiconductors and conductors, author = Hyperphysics, year = 2020, url = http://hyperphysics.phy-astr.gsu.edu/hbase/Solids/band.html, type = ONLINE, cite = bandgaphyperphysics*.

- [130] University of Cambridge. *Dissemination of IT for the Promotion of Materials Science*. 2020. URL: <https://www.doitpoms.ac.uk/tlplib/semiconductors/fermi.php>.
- [131] M. Methfessel and A. T. Paxton. “High-precision sampling for Brillouin-zone integration in metals”. In: *Physical Review B* 40 (6 1989), pp. 3616–3621.
- [132] N. Marzari. *Ab-initio Molecular Dynamics for Metallic Systems*. 1996, p. 136.
- [133] Hector Balboa et al. “Damage characterization of (U,Pu)O₂ under irradiation by molecular dynamics simulations”. In: *Journal of Nuclear Materials* 512 (2018), pp. 440–449.
- [134] K. A. Nekrasov and. “Sputtering of material from the surface of PuO₂ crystals by collision cascades impact. A molecular dynamics study”. In: *Journal of Nuclear Materials* 512 (2018), pp. 440–449.
- [135] David J. Bacon, Andrew F. Calder, and F Gao. “Defect production due to displacement cascades in metals as revealed by computer simulation”. In: *Journal of Nuclear Materials* 251 (1997), pp. 1–12.
- [136] K. Trachenko et al. “Modeling high-energy radiation damage in nuclear and fusion applications”. In: *Nuclear Instruments and Methods in Physics Research B* 277 (2012), pp. 6–13.
- [137] H. Stehfest. “Algorithm 368 Numerical Inversion of Laplace Transform”. In: *Communications of the ACM* 13 (1970), pp. 47–49.
- [138] J. Ziegler. *The Stopping and Range of Ions in Matter*. 2019. URL: <http://www.srim.org/SRIM/SRIMINTRO.htm>.
- [139] D. Rochman and A. J. Koning. “Modern Nuclear Data Evaluation With The TALYS Code System”. In: *Nuclear Data Sheets* 113 (2012).
- [140] webelements.com. *Webelements: Aluminium*. 2020. URL: <https://www.webelements.com/aluminium>.
- [141] webelements.com. *Webelements: Iron*. 2020. URL: <https://www.webelements.com/iron>.
- [142] webelements.com. *Webelements: Nickel*. 2020. URL: <https://www.webelements.com/nickel>.
- [143] webelements.com. *Webelements: Palladium*. 2020. URL: <https://www.webelements.com/palladium>.
- [144] N. Marzari et al. “Thermal Contraction and Disordering of the Al(110) Surface Nicola”. In: *Condensed Matter* (110 2008), pp. 4–7.
- [145] G. Dimitri Ngantso and A. T. Raji. “Adsorption of Br₂ molecule on the Fe W(110) substrate - Energetics, electronic and magnetic properties”. In: *Computational Condensed Matter* 23 (2020).
- [146] A. E. Rezaee and M. Almasi-Kashi. “The influence of point defects on Na diffusion in black phosphorene - first principles study”. In: *Journal Pre-proof* ().
- [147] N. Acharya and S. P. Sanyal. “Structural phase transition, electronic and superconducting properties of ScBi and YBi”. In: *Solid State Communications* 266 (2017), pp. 39–45.
- [148] L. Malakkal and R. K. Siripurapu. “First principles calculations of hydrogen storage on Cu and Pd-decorated graphene”. In: *International Journal of Hydrogen Energy* 41 (2016), pp. 17652–17656.
- [149] P. O. Adebambo and G. A. Adebayo. “Elastic constants and observed ferromagnetism in inverse Heusler alloy Ti₂CoAs using kjpaw pseudopotentials - A first-principles approach”. In: *Journal of Alloys and Compounds* 722 (2017), pp. 207–211.

Chapter 13

Acknowledgements

Thank you to Brian, Mark and Alessandro for your supervision and support, to my wife Rachel, son Zakk, parents and my family for encouragement. The various forums in the scientific community, in particular useful discussions with Dr. Todorov of Daresbury and members of the Quantum Espresso forum, have been helpful throughout. Two teachers in particular from my school days, Mr. George and Mr. Rockett for their encouragement in the sciences, in particular Physics.

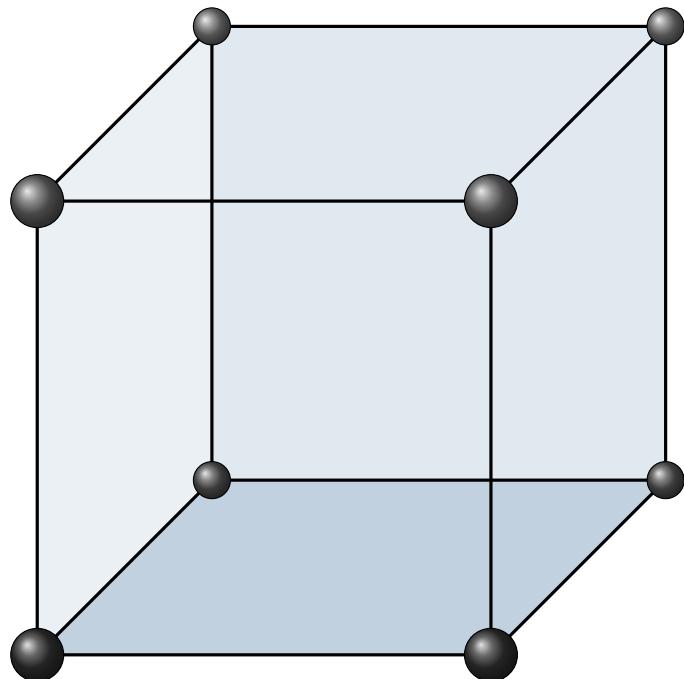
Appendices

Appendix A

Crystal Structures

A.1 Common Cubic Structures

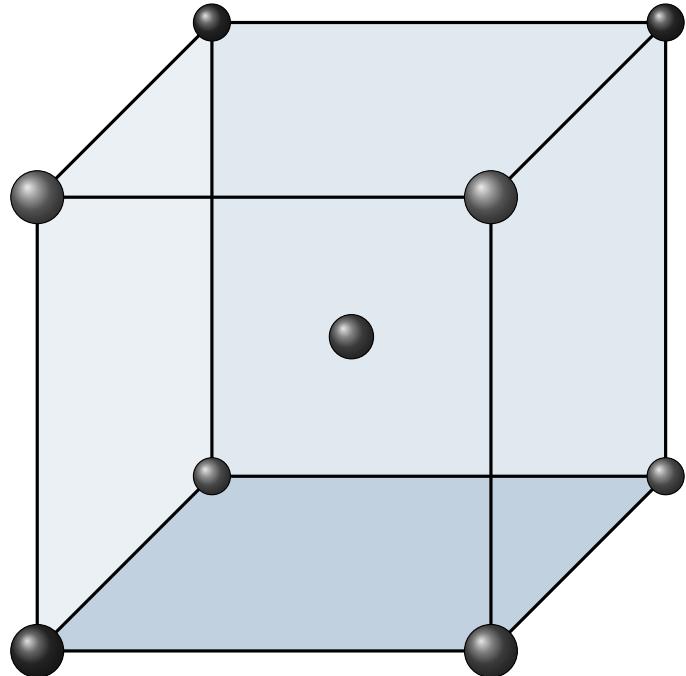
A.1.1 Simple Cubic



Sample Configuration
[0.0 0.0 0.0]

Table A.1

A.1.2 Body Centered Cubic

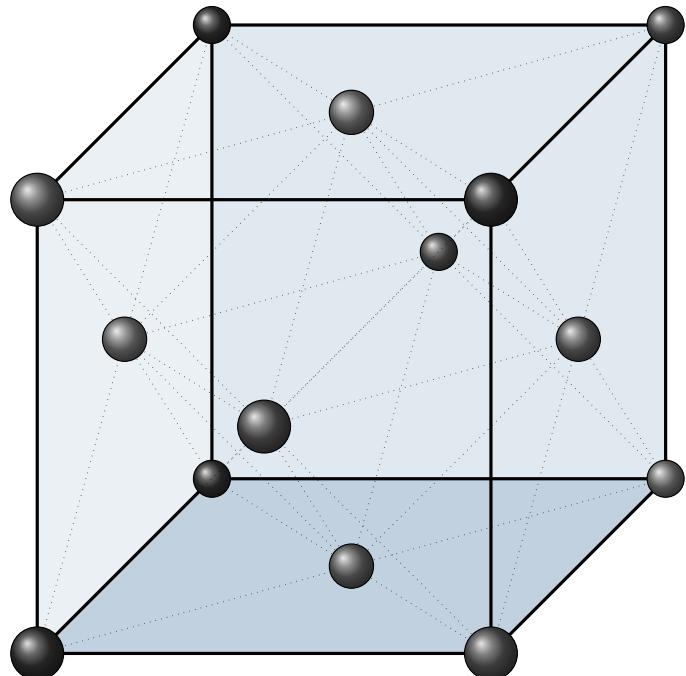


Sample Configuration

$$\begin{array}{ccc} [0.0 & 0.0 & 0.0] \\ [0.5 & 0.5 & 0.5] \end{array}$$

Table A.2

A.1.3 Face Centered Cubic

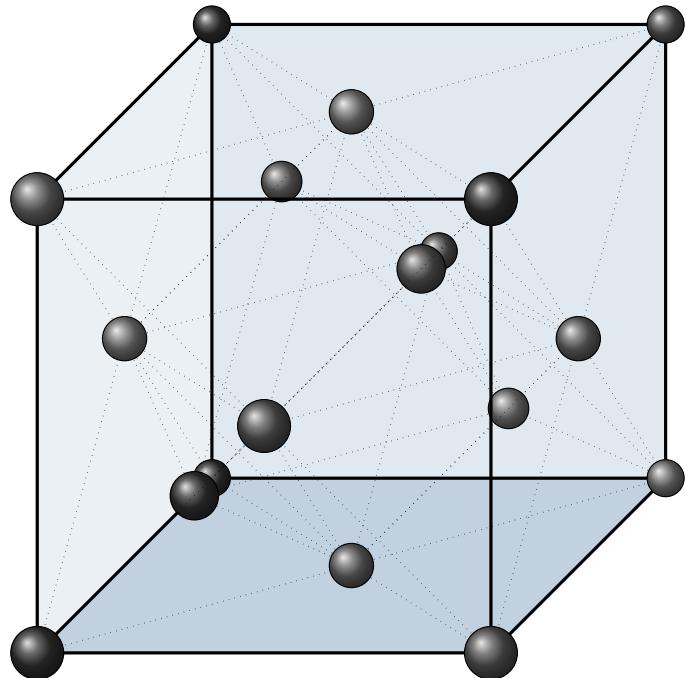


Sample Configuration

[0.0	0.0	0.0]
[0.0	0.5	0.5]
[0.5	0.0	0.5]
[0.5	0.5	0.0]

Table A.3

A.1.4 Zincblende



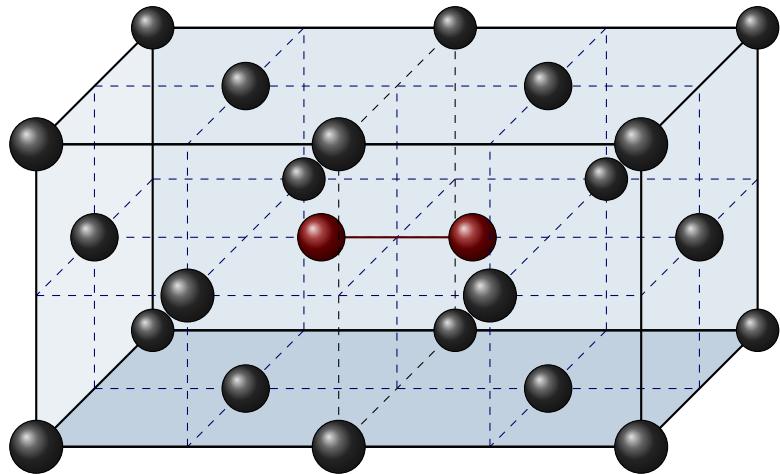
Sample Configuration

[0.0	0.0	0.0]
[0.0	0.5	0.5]
[0.5	0.0	0.5]
[0.5	0.5	0.0]
[0.25	0.25	0.25]
[0.75	0.75	0.25]
[0.25	0.75	0.75]
[0.75	0.25	0.75]

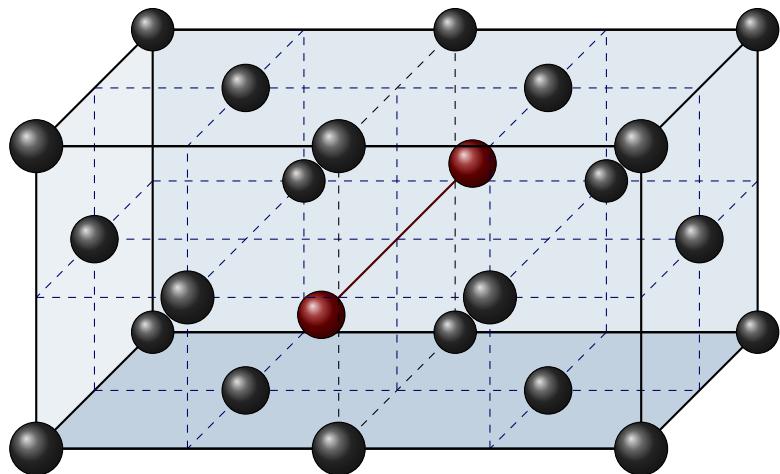
Table A.4

A.2 FCC Defects

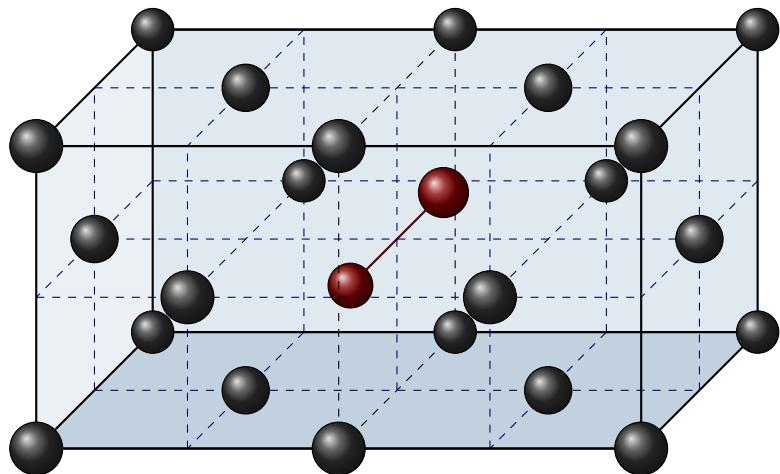
A.2.1 Dumbbell $<100>$



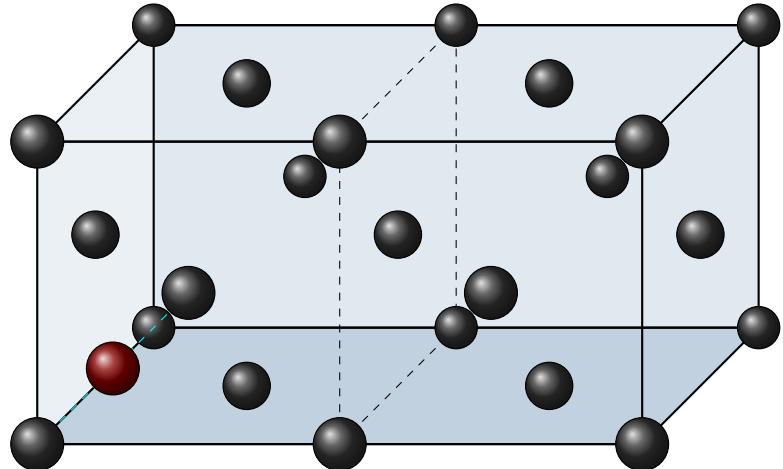
A.2.2 Dumbbell $<110>$



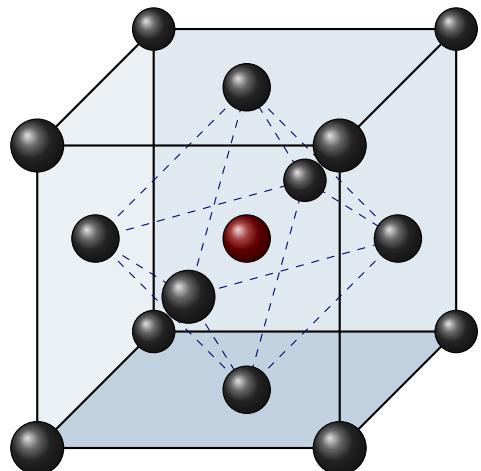
A.2.3 Dumbbell $<111>$



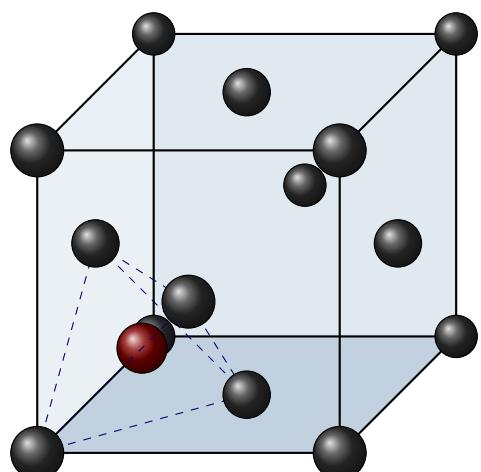
A.2.4 Crowdion



A.2.5 Octahedral Interstitial



A.2.6 Tetrahedral Interstitial



Appendix B

Elastic Constants

Crystal:	Cubic
Sides:	$a = b = c$
Angles:	$\alpha = \beta = \gamma = \frac{\pi}{2}$ radians
Symmetry:	$\frac{\pi}{2}$ radians rotation, 3 mutually orthogonal planes of symmetry
No. Independent Elastic Constants:	3
Elastic Constants:	$\begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{44} \end{bmatrix}$

Table B.1

Crystal:	Orthorhombic/Orthotropic
Sides:	$a \neq b \neq c$
Angles:	$\alpha = \beta = \gamma = \frac{\pi}{2}$ radians
Symmetry:	3 mutually orthogonal planes of symmetry
No. Independent Elastic Constants:	9
Elastic Constants:	$\begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{13} & C_{23} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix}$

Table B.2

Crystal:	Monoclinic
Sides:	$a \neq b \neq c$
Angles:	$\alpha < \frac{\pi}{2}$ radians, $\beta = \gamma = \frac{\pi}{2}$ radians
Symmetry:	symmetry in the xy plane
No. Independent Elastic Constants:	13
Elastic Constants:	$\begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & C_{16} \\ C_{12} & C_{22} & C_{23} & 0 & 0 & C_{26} \\ C_{13} & C_{23} & C_{33} & 0 & 0 & C_{36} \\ 0 & 0 & 0 & C_{44} & C_{45} & 0 \\ 0 & 0 & 0 & C_{45} & C_{55} & 0 \\ C_{16} & C_{26} & C_{36} & 0 & 0 & C_{66} \end{bmatrix}$

Table B.3

Crystal:	Triclinic
Sides:	$a \neq b \neq c$
Angles:	$\alpha < \frac{\pi}{2}$ radians, $\beta < \frac{\pi}{2}$ radians, $\gamma < \frac{\pi}{2}$ radians
Symmetry:	no symmetry
No. Independent Elastic Constants:	21
Elastic Constants:	$\begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & C_{16} \\ C_{12} & C_{22} & C_{23} & 0 & 0 & C_{26} \\ C_{13} & C_{23} & C_{33} & 0 & 0 & C_{36} \\ 0 & 0 & 0 & C_{44} & C_{45} & 0 \\ 0 & 0 & 0 & C_{45} & C_{55} & 0 \\ C_{16} & C_{26} & C_{36} & 0 & 0 & C_{66} \end{bmatrix}$

Table B.4

Appendix C

Activity Equation in Python: decay class

C.1 Full Source Code

The full source code is available to download from github.

https://github.com/BenPalmer1983/atomic_dictionaries <https://github.com/BenPalmer1983/decay>

C.2 Highlighted Code

The code below is taken from the decay class and it is specifically responsible for calculating the amount of isotopes in the decay chain at some time t.

Listing C.1: Decay python3 class for calculating the amount of an isotope in a decay chain at time t

```
1 import os
2 import numpy
3 from pz import pz
4 from isotopes import isotopes
5 import matplotlib.pyplot as plt
6 import copy
7 import hashlib
8
9 class decay:
10
11     path_isotopes = ".../data/isotopes.pz"
12     loaded = False
13
14     @staticmethod
15     def set(path_isotopes):
16         decay.path_isotopes = path_isotopes
17         isotopes.set(path_isotopes)
18         decay.load()
19
20     @staticmethod
21     def load():
22         if(decay.loaded == False):
23             decay.loaded = True
```

```

26     @staticmethod
27     def chain_isotopes(key, out=[]):
28         if(not isotopes.is_valid(key)):
29             return out
30         if(key not in out):
31             out.append(key)
32         else:
33             return out
34         if(isotopes.is_stable(key)):
35             return out
36         else:
37             dm = isotopes.get_decay_modes(key)
38             for k in dm.keys():
39                 decay.chain_isotopes(k, out)
40             return out
41
42
43     @staticmethod
44     def make_chain(key, l=0, out=[], bf=0.0):
45         if(not isotopes.is_valid(key)):
46             return out
47         if(l == 0):
48             decay.chains_store = []
49             out.append([l, key, bf])
50         if(isotopes.is_stable(key)):
51             if(len(out) > 1):
52                 for i in range(len(out)-1,1,-1):
53                     if(out[i-1][0]>=out[i][0]):
54                         out.pop(out[i-1][0])
55             return decay.chains_store.append(copy.deepcopy(out))
56         else:
57             l = l + 1
58             dms = isotopes.get_decay_modes(key)
59             for k in dms.keys():
60                 bf = dms[k]['branching_factor']
61                 decay.make_chain(k, l, out, bf)
62             return out
63
64     @staticmethod
65     def calculate(parent, time, i_data_in, log=None, custom_chain=None):
66
67         if(log != None):
68             log_dir = decay.get_file_dir(log)
69             print(log_dir)
70             decay.make_dir(log_dir)
71
72         decay.results = {
73             'tally': {},
74             'unique': None,
75             'chains': None,
76         }
77         decay.chains_store = None
78
79         if(custom_chain == None):
80             decay.chains_store = []
81             decay.make_chain(parent, 0, [])
82             decay.results['chains'] = []
83             cn = 0
84             for chain in decay.chains_store:
85                 decay.results['chains'].append([])
86                 for iso in chain:

```

```

87         k = iso[1]
88         bf = iso[2]
89         i_data = isotopes.get(k)
90         half_life = None
91         n0 = 0.0
92         w = 0.0
93         if(not i_data['stable']):
94             half_life = i_data['half_life']
95         if(k in i_data_in.keys()):
96             n0 = i_data_in[k]['n0']
97             w = i_data_in[k]['w']
98         d = {
99             'isotope_key': k,
100            'bf': bf,
101            'w': w,
102            'n0': n0,
103            'nend': 0,
104            'half_life': half_life,
105            }
106        decay.results['chains'][cn].append(d)
107        cn = cn + 1
108    else:
109        decay.results['chains'] = custom_chain
110
111
112 # Find unique and make tally
113 decay.results['unique'] = []
114 for chain in decay.results['chains']:
115     for iso in chain:
116         k = iso['isotope_key']
117         if(k not in decay.results['tally'].keys()):
118             decay.results['unique'].append(k)
119             # Use provided data
120             if(iso['half_life'] == None):
121                 stable = True
122                 half_life = None
123                 decay_constant = 0.0
124             else:
125                 stable = False
126                 half_life = iso['half_life']
127                 decay_constant = numpy.log(2) / iso['half_life']
128                 w = iso['w']
129                 n0 = iso['n0']
130                 # Get proton/neutron etc from isotopes database
131                 i_data = isotopes.get(k)
132                 if(i_data is None):
133                     decay.results['tally'][k] = {
134                         'printable': 'Custom',
135                         'element': 'ZZ',
136                         'protons': 999,
137                         'nucleons': 999,
138                         'metastable': 9,
139                         'stable': stable,
140                         'half_life': half_life,
141                         'decay_constant': decay_constant,
142                         'w': w,
143                         'n0': n0,
144                         'nend': 0.0,
145                         }
146                 else:
147                     decay.results['tally'][k] = {

```

```

148         'printable': decay.pad(isotopes.get_printable_name(k), 12),
149         'element': i_data['element'],
150         'protons': i_data['protons'],
151         'nucleons': i_data['nucleons'],
152         'metastable': i_data['metastable'],
153         'stable': stable,
154         'half_life': half_life,
155         'decay_constant': decay_constant,
156         'w': w,
157         'n0': n0,
158         'nend': 0.0,
159     }
160
161
162 decay.results['chains_individual'] = []
163 for cn in range(len(decay.results['chains'])):
164     for n in range(len(decay.results['chains'][cn])):
165         nc = []
166         if(decay.results['chains'][cn][n]['n0']>0.0 or decay.results['chains'][cn][n]['w']>0.0):
167             for j in range(n, len(decay.results['chains'][cn])):
168                 iso = copy.deepcopy(decay.results['chains'][cn][j])
169                 if(j>n):
170                     iso['n0'] = 0.0
171                     iso['w'] = 0.0
172                 nc.append(iso)
173             if(len(nc)>0):
174                 decay.results['chains_individual'].append(nc)
175
176 for cn in range(len(decay.results['chains_individual'])):
177     chain = decay.results['chains_individual'][cn]
178     n0 = numpy.zeros((len(chain),))
179     w = numpy.zeros((len(chain),))
180     l = numpy.zeros((len(chain),))
181     b = numpy.zeros((len(chain)-1,))
182     for n in range(len(decay.results['chains_individual'][cn])):
183         k = decay.results['chains_individual'][cn][n]['isotope_key']
184         n0[n] = decay.results['chains_individual'][cn][n]['n0']
185         w[n] = decay.results['chains_individual'][cn][n]['w']
186         l[n] = decay.results['tally'][k]['decay_constant']
187         if(n>0):
188             b[n-1] = decay.results['chains_individual'][cn][n]['bf']
189     nt = decay.calculate_activity(time, l, b, w, n0)
190     for n in range(len(decay.results['chains_individual'][cn])):
191         decay.results['chains_individual'][cn][n]['nend'] = nt[n]
192
193 set = []
194 for cn in range(len(decay.results['chains_individual'])):
195     ckey = ''
196     for n in range(len(decay.results['chains_individual'][cn])):
197         k = decay.results['chains_individual'][cn][n]['isotope_key']
198         ckey = ckey + str(decay.results['chains_individual'][cn][n]['isotope_key']) + "NO:" + str(decay.results['chains_individual'][cn][n]['n0']) + "W:" + str(decay.results['chains_individual'][cn][n]['w'])
199         ckeyh = hashlib.md5(ckey.encode())
200         ckeyh = ckeyh.hexdigest()
201         if(ckeyh not in set):
202             decay.results['tally'][k]['nend'] = decay.results['tally'][k]['nend'] + decay.results['chains_individual'][cn][n]['nend']
203             set.append(ckeyh)
204
205
206 # Log

```

```

207     if(log != None):
208         width = 140
209         fh = open(log, 'w')
210         fh.write("Unique Isotopes\n")
211         fh.write(decay.hr(width) + "\n")
212         for k in decay.results['unique']:
213             line = decay.results['tally'][k]['printable']
214             fh.write(line + "\n")
215             fh.write("\n")
216             fh.write("\n")
217             fh.write("Decay Chains\n")
218             fh.write(decay.hr(width) + "\n")
219             fh.write("\n")
220             fh.write("\n")
221             for cn in range(len(decay.results['chains'])):
222                 chain = decay.results['chains'][cn]
223
224                 fh.write(decay.pad(cn+1,6))
225                 for n in range(len(chain)):
226                     iso = chain[n]
227                     k = iso['isotope_key']
228                     if(n>0):
229                         bf = "{0:3e}".format(iso['bf'])
230                         fh.write(" --[" + str(bf) + "]--> ")
231                         fh.write(decay.pad(decay.results['tally'][k]['printable'], 12))
232                         fh.write("\n")
233                         fh.write(decay.pad("T1/2",6))
234                         for n in range(len(chain)):
235                             if(n>0):
236                                 fh.write(decay.pad("",17))
237                             if(decay.results['chains'][cn][n]['half_life'] == None):
238                                 fh.write(decay.pad("[Stable]",12))
239                             else:
240                                 fh.write(decay.pad("[" + str("{0:8e}".format(decay.results['chains'][cn][n]['half_life'])) + "]",12))
241                         fh.write("\n")
242                         fh.write("\n")
243                         fh.write("\n")
244
245
246
247             fh.write("Amounts\n")
248             fh.write(decay.hr(width) + "\n")
249             fh.write("\n")
250             fh.write(decay.hr(width) + "\n")
251             line = decay.pad("Isotope", 12)
252             line = line + decay.pad("T(1/2)", 18)
253             line = line + decay.pad("Decay Constant", 18)
254             line = line + decay.pad("W", 18)
255             line = line + decay.pad("N(t=0)", 18)
256             line = line + decay.pad("N(t=" + str(time) + ")", 18)
257             line = line + decay.pad("A(t=0)", 18)
258             line = line + decay.pad("A(t=" + str(time) + ")", 18)
259             fh.write(line + "\n")
260             fh.write(decay.hr(width) + "\n")
261
262
263             for k in decay.results['tally'].keys():
264                 line = decay.pad(decay.results['tally'][k]['printable'], 12)
265                 if(decay.results['tally'][k]['half_life'] is None):
266                     line = line + decay.pad("Stable", 18)
267                     line = line + decay.pad("Stable", 18)

```

```

268     else:
269         line = line + decay.pad(str("{0:16e}").format(decay.results['tally'][k]['half_life'])).strip(), 18)
270         line = line + decay.pad(str("{0:16e}").format(decay.results['tally'][k]['decay_constant'])).strip(), 18)
271         line = line + decay.pad(str("{0:16e}").format(decay.results['tally'][k]['w'])).strip(), 18)
272         line = line + decay.pad(str("{0:16e}").format(decay.results['tally'][k]['n0'])).strip(), 18)
273         line = line + decay.pad(str("{0:16e}").format(decay.results['tally'][k]['nend'])).strip(), 18)
274     if(decay.results['tally'][k]['half_life'] is not None):
275         line = line + decay.pad(str("{0:16e}").format(decay.results['tally'][k]['decay_constant'] * decay.results
276             ['tally'][k]['n0'])).strip(), 18)
277         line = line + decay.pad(str("{0:16e}").format(decay.results['tally'][k]['decay_constant'] * decay.results
278             ['tally'][k]['nend'])).strip(), 18)
279     fh.write(line + "\n")
280     fh.write(decay.hr(width) + "\n")
281     fh.write("\n")
282     fh.write("\n")
283
284
285 ######
286 # DECAY EQUATIONS
287 #####
288
289 @staticmethod
290 def calculate_activity(t, l, b, w, n0):
291     nt = numpy.zeros((len(n0),))
292     for m in range(0,len(n0)):
293         if(l[m] > 0.0):
294             nt[m] = decay.activity_unstable(t, l, b, w, n0, m)
295         elif(l[m] == 0.0):
296             nt[m] = decay.activity_stable(t, l, b, w, n0, m)
297     return nt
298
299 @staticmethod
300 def activity_unstable(t, l, b, w, n0, m):
301     s = 0.0
302     for k in range(0, m+1):
303         s = s + decay.r(k, m, b, 1) * (decay.f_unstable(t,k,m,1) * n0[k] + decay.g_unstable(t,k,m,1) * w[k])
304     return s
305
306 @staticmethod
307 def f_unstable(t,k,m,l):
308     s = 0.0
309     for i in range(k, m+1):
310         p = 1.0
311         for j in range(k, m+1):
312             if(i != j):
313                 p = p * (1 / (l[i] - l[j]))
314             s = s + numpy.exp(-1 * l[i] * t) * p
315     s = (-1)**(m-k) * s
316     return s
317
318 @staticmethod
319 def g_unstable(t,k,m,l):
320     pa = 1.0
321     for i in range(k,m+1):
322         pa = pa * l[i]
323         pa = 1.0 / pa
324     s = 0.0
325     for i in range(k, m+1):
326         pb = 1.0

```

```

327     for j in range(k, m+1):
328         if(i != j):
329             pb = pb * (1 / (l[i]-l[j]))
330         s = s + (1/l[i]) * numpy.exp(-l[i]*t) * pb
331     return pa + s * (-1)**(m-k+1)
332
333 @staticmethod
334 def activity_stable(t, l, b, w, n0, m):
335     s = n0[m] + w[m] * t
336     for k in range(0, m):
337         s = s + decay.r(k, m, b, 1) * (decay.f_stable(t,k,m-1,1) * n0[k] + decay.g_stable(t,k,m-1,1) * w[k])
338     return s
339
340 @staticmethod
341 def f_stable(t,k,m,l):
342     p = 1.0
343     for i in range(k, m+1):
344         p = p * l[i]
345     s = 0.0
346     for i in range(k, m+1):
347         r = l[i]
348         for j in range(k, m+1):
349             if(i != j):
350                 r = r * (l[i] - l[j])
351             s = s + (1/r)*numpy.exp(-l[i]*t)
352     return (1.0/p) + s * (-1.0)**(m-k+1)
353
354 @staticmethod
355 def g_stable(t,k,m,l):
356     nd = 0.0
357     mp = (-1.0)**(m - k)
358
359     r = 1.0
360     for i in range(k, m+1):
361         r = r * l[i]
362     nd = nd + (1.0/r) * t
363
364     p = 0.0
365     for i in range(k, m+1):
366         tm = 1.0
367         for j in range(k, m+1):
368             if(i != j):
369                 tm = tm * l[j]
370             p = p + tm
371
372     q = 1.0
373     for i in range(k, m+1):
374         q = q * l[i]*l[i]
375
376     r = (-1.0) * (p / q)
377     nd = nd + r
378
379     C = 0.0
380     for i in range(k, m+1):
381         r = 1.0 * l[i] * l[i]
382         for j in range(k, m+1):
383             if(i != j):
384                 r = r * (l[i] - l[j])
385             C = C + (1/r) * numpy.exp(-1.0 * l[i] * t)
386     nd = nd + C * mp

```

```

388
389     return nd
390
391
392     @staticmethod
393     def r(k, m, b, l):
394         if(k == m):
395             return 1.0
396         else:
397             p = 1.0
398             for i in range(k, m):
399                 p = p * (b[i] * l[i])
400             return p
401
402
403     @staticmethod
404     def pad(inp, l=16):
405         inp = str(inp)
406         while(len(inp)<l):
407             inp = inp + " "
408         return inp
409
410
411     @staticmethod
412     def hr(l=16):
413         inp = ""
414         while(len(inp)<l):
415             inp = inp + "="
416         return inp
417
418
419     @staticmethod
420     def get_file_dir(file_path):
421         file_path = file_path.strip()
422         if(file_path[0] != "/"):
423             root = os.getcwd()
424             file_path = root + "/" + file_path
425         file_path = file_path.split("/")
426         path = ""
427         for i in range(1,len(file_path) - 1):
428             path = path + "/" + file_path[i]
429         return path
430
431     @staticmethod
432     def make_dir(dir):
433         dirs = dir.split("/")
434         try:
435             dir = ''
436             for i in range(len(dirs)):
437                 dir = dir + dirs[i]
438                 if(not os.path.exists(dir) and dir.strip() != ''):
439                     os.mkdir(dir)
440                 dir = dir + '/'
441             return True
442         except:
443             return False
444
445     @staticmethod
446     def test():
447         print("Decay")
448

```

```
449 # Load isotopes dictionary
450 decay.set("../data/isotopes.pz")
451
452     idata = {}
453     parent = 84216
454     time = 10
455     idata[84216] = {'w': 0.20, 'n0': 100.0}
456     idata[82212] = {'w': 0.0, 'n0': 5.0}
457     idata[83212] = {'w': 0.07, 'n0': 15.0}
458     idata[81208] = {'w': 0.005, 'n0': 0.0}
459     # 84Po212 0 0 (default)
460     idata[82208] = {'w': 0.01, 'n0': 300.0}
461     decay.calculate(parent, time, idata, "testing/log_84216_new.txt")
462
463
464 def main():
465     decay.test()
466
467 if __name__ == "__main__":
468     main()
```

Appendix D

Decay Equations and Numeric Computation for Po-216

This section of the appendix contains the full details of the first version of the Fe-Pd potential.

Listing D.1: Activity V1 Input File

```
1 PROGRAM main_test
2 ! University of Birmingham
3 ! Ben Palmer
4 Use kinds
5 Use MPI
6
7 IMPLICIT NONE
8
9 CALL main()
10
11 CONTAINS
12
13 ! Subroutines
14
15
16 SUBROUTINE main()
17 !#####
18 ! PRIVATE VARIABLES
19 INTEGER(kind=StandardInteger) :: i, j, k
20 INTEGER(kind=StandardInteger) :: steps
21 REAL(kind=DoubleReal) :: w(1:6)
22 REAL(kind=DoubleReal) :: l(1:6)
23 REAL(kind=DoubleReal) :: no(1:6)
24 REAL(kind=DoubleReal) :: n(1:6)
25 REAL(kind=DoubleReal) :: source(1:6)
26 REAL(kind=DoubleReal) :: loss(1:6)
27 REAL(kind=DoubleReal) :: t_end, t_step
28 !#####
29
30 print *, "Decay Test - Po216"
31
32 w(:) = 0.0D0
33 w(1) = 0.2D0
34 w(3) = 0.07D0
35 w(4) = 0.005D0
36 w(6) = 0.01D0
37
38 l(1) = 4.683427D+00 ! 84Po216
```

```

39 l(2) = 1.809595D-05 ! 82Pb212
40 l(3) = 1.908235D-04 ! 83Bi212
41 l(4) = 3.777781D-03 ! 81Tl208
42 l(5) = 2.310491D+06 ! 84Po212
43 ! 82Pb208
44
45 n0(:) = 0.0D0
46 n0(1) = 1.0D2
47 n0(2) = 5.0D0
48 n0(3) = 1.5D1
49 n0(6) = 3.0D2
50
51 n(:) = n0(:)
52
53 t_end = 10.0D0
54 t_step = t_end / 1.0D10
55
56 Do k=1,10
57 DO i=1,1000000000
58   source = 0.0D0
59   loss = 0.0D0
60
61   source(1) = t_step * w(1)
62   loss(1) = n(1) * (1.0D0 - exp(-l(1) * t_step))
63
64   source(2) = t_step * w(2) + loss(1)
65   loss(2) = n(2) * (1.0D0 - exp(-l(2) * t_step))
66
67   source(3) = t_step * w(3) + loss(2)
68   loss(3) = n(3) * (1.0D0 - exp(-l(3) * t_step))
69
70   source(4) = t_step * w(4) + 0.359300D0 * loss(3)
71   loss(4) = n(4) * (1.0D0 - exp(-l(4) * t_step))
72
73   source(5) = t_step * w(5) + 0.640700D0 * loss(3)
74   loss(5) = n(5) * (1.0D0 - exp(-l(5) * t_step))
75
76   source(6) = t_step * w(6) + loss(4) + loss(5)
77
78   n(1:6) = n(1:6) + source(1:6) - loss(1:6)
79
80 End Do
81 End Do
82
83
84 print *, "Po216 ", n(1)
85 print *, "Pb212 ", n(2)
86 print *, "Bi212 ", n(3)
87 print *, "Tl208 ", n(4)
88 print *, "Po212 ", n(5)
89 print *, "Pb208 ", n(6)
90
91
92 End SUBROUTINE main
93
94
95
96 End PROGRAM main_test

```

Appendix E

Activity V1

E.1 Sample Input File

Listing E.1: Activity V1 Input File

```
1 #elements
2 Fe 100
3 #isotopes
4 "/home/ben/activity/data/isotopes.txt"
5 #decaymodes
6 "/home/ben/activity/data/decaymodes.txt"
7 #gammaenergies
8 "/home/ben/activity/data/gammaenergies.txt"
9 #xsfiles
10 "/home/ben/activity/data/xs"
11 #trajfile
12 "Fe36MeV.exyz"
13 #polyfitorder
14 5
15 #integrationgranularity
16 10
17 #beamflux
18 0.5 uA
19 #beamenergy
20 36 MeV
21 #beadmduration
22 300 s
23 #beamarea
24 100 mm2
25 #amtime
26 260000 s
27 #timestep
28 1000 s
29 #projectile
30 1 1
31 #targetthickness
32 0.5 mm
33 #materialdensity
34 8000 kgm3
35 #vpi
36 60.2
37 #individualisotopeactivity
38 yes
39 #verboseterminal
```

```
40 yes
41 #targetdpa
42 0.0
43 #gammachartresolution
44 200
```

E.2 Output Plots

The Activity V1 code and the above input file generates data files and a selection of plots. These plots include the activity of the top five radioactive isotopes over the entire simulation period, from the time the target is irradiated until the simulation end time is reached (fig. ??). A second plot is that of the expected gamma lines at the simulation end time (fig. ??).

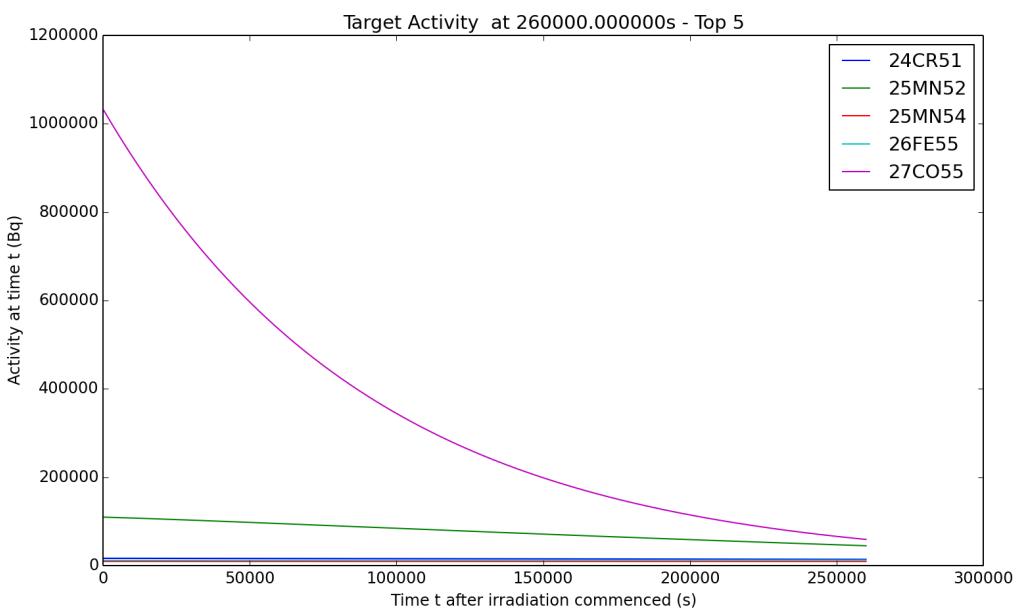


Figure E.1: Top 5 radioactive isotopes over the entire simulation time using the input file in section ??

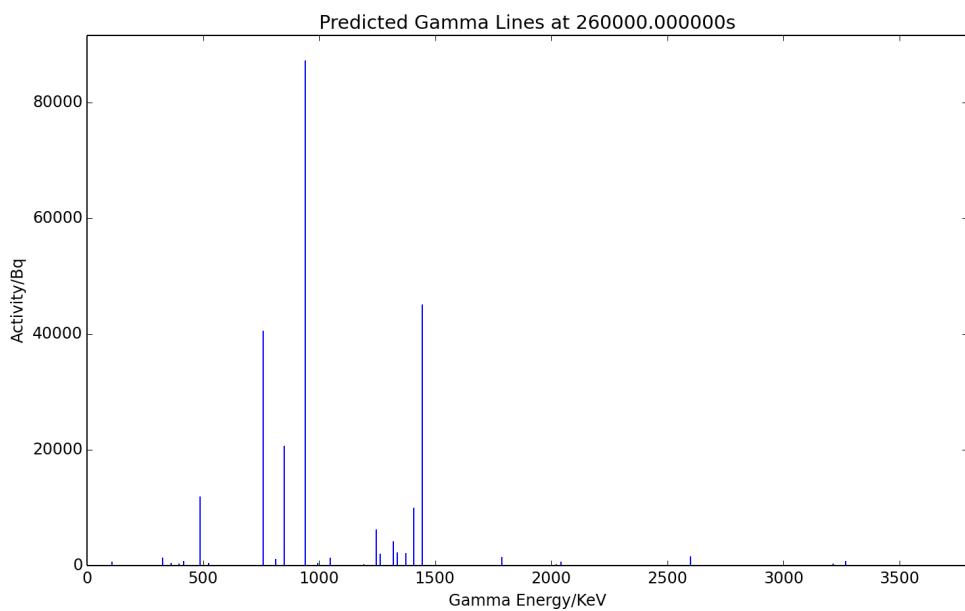
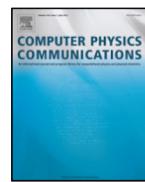


Figure E.2: Expected gamma lines at the end of the simulation ??

E.3 Paper published in Computer Physics Communications



Activity computer program for calculating ion irradiation activation[☆]

Ben Palmer ^{*}, Brian Connolly, Mark Read

University of Birmingham, United Kingdom



ARTICLE INFO

Article history:

Received 1 June 2015

Received in revised form 15 February 2017

Accepted 19 February 2017

Available online 29 March 2017

ABSTRACT

A computer program, Activity, was developed to predict the activity and gamma lines of materials irradiated with an ion beam. It uses the TENDL (Koning and Rochman, 2012) [1] proton reaction cross section database, the Stopping and Range of Ions in Matter (SRIM) (Biersack et al., 2010) code, a Nuclear Data Services (NDS) radioactive decay database (Sonzogni, 2006) [2] and an ENDF gamma decay database (Herman and Chadwick, 2006) [3]. An extended version of Bateman's equation is used to calculate the activity at time t, and this equation is solved analytically, with the option to also solve by numeric inverse Laplace Transform as a failsafe. The program outputs the expected activity and gamma lines of the activated material.

Program summary

Program title: Activity

Catalogue identifier: AFBS_v1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AFBS_v1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: GNU GPL v3

No. of lines in distributed program, including test data, etc.: 688828

No. of bytes in distributed program, including test data, etc.: 71056048

Distribution format: tar.gz

Programming language: Fortran.

Computer: PCs or HPCs.

Operating system: Linux (tested on Debian).

Has the code been vectorized or parallelized?: OpenMPI

RAM: 250MB per process + 200MB overhead

Classification: 2.2, 17.8.

Nature of problem: To calculate the predicted activity of an ion irradiated target. The expected range of ion energies is between 1MeV and 200MeV; this is the range of the available ion cross section data.

Solution method: The program loads cross section data from the TENDL database and trajectory data from a SRIM [1] simulation exyz data file. It uses this data to calculate the production/loss rate of each isotope in the simulated target. Radioactive decay equations are used to calculate the amounts and activity of each radioactive isotope at the set time.

Running time: Typically the Activity program runs each input from seconds to no more than several minutes.

References:

- [1] SRIM – The stopping and range of ions in matter (2010). Ziegler, James F., Ziegler, M.D. and Biersack, J.P. 2010, Nuclear Instruments and Methods in Physics Research Section B, Vol. 268, pp. 1818–1823.

© 2017 Elsevier B.V. All rights reserved.

[☆] This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (<http://www.sciencedirect.com/science/journal/00104655>).

* Corresponding author.

E-mail address: benpalmer1983@gmail.com (B. Palmer).

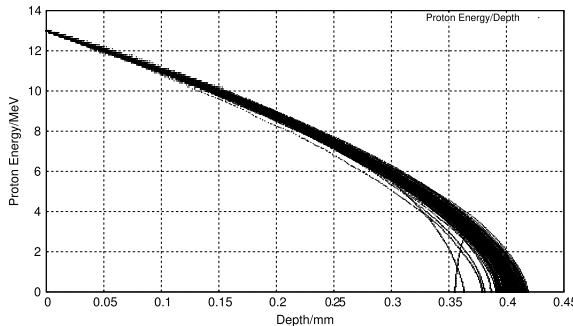


Fig. 1. One hundred simulated 13 MeV proton energy loss curves in Fe simulated with SRIM [4].

1. Background

1.1. Ion irradiation at the University of Birmingham

The Scanditronix MC-40 Cyclotron is used at the University of Birmingham to create a beam of protons or other light ions. The energies of these ions are typically between 10 MeV and 60 MeV with beam currents ranging up to 50 μA (3.1×10^{14} protons per second). Target materials are irradiated by this cyclotron for a number of reasons, including purposely creating radioactive isotopes for the nearby Queen Elizabeth Hospital, investigating ion irradiation damage and emulating neutron irradiation.

The cyclotron is usually used to create radioactive isotopes for medical use, but an additional beam line has been devoted to material science investigations into radiation damage. While the creation of radioactive isotopes is desired in some cases, material being tested for radiation damage should preferably have low levels of radioactivity.

It is expensive to arrange the irradiation of target materials by high energy neutrons sources, whereas it is relatively inexpensive to irradiate using an ion beam on the MC-40 Cyclotron. The energies can be controlled, and a set dose at a single energy, or a range of energies, can be precisely deposited into the target material.

The Activity code discussed here was developed to calculate the activity of a target material irradiated by a proton beam. It has been developed in Fortran and uses data from the TENDL-2013 proton cross section database, SRIM ion transport code and NDS radioactive decay database.

1.2. Simulating ion irradiation with SRIM

A package of ion transport codes, SRIM, is freely available to download and use to investigate the transport of ions through matter. SRIM uses the binary collision approximation (BCA) to simulate the passage of ions in a material. It is an approximate method, and one key restriction is that it does not take into account the structure of the material, and this approximation is therefore also imposed on the Activity code.

One file that SRIM creates is of importance to the Activity code, and that is the trajectory file that contains the energy and x , y , z co-ordinate data points for simulated ions moving through matter. Fig. 1 shows the trajectory of one hundred 13 MeV protons entering and passing through an Iron target, and it is this set of data points (together with the cross section database) that the Activity code uses to calculate the reaction rates for the transmutation of nuclei in the target. At higher energies, the ions slow as they lose energy due to electronic stopping, but as the ion energy drops the mechanism of loss through nuclear collisions becomes important.

The spreading of ion depths at lower energies is a result of the higher momentum transfer during nuclear collisions, as can be seen in Fig. 1.

1.3. Transmutation of nuclei by ion irradiation

Considering a simplified nuclear potential well, energetic protons approaching a nucleus may overcome the Coulomb potential barrier. They are captured by the nucleus and held within the potential well by the strong nuclear force. This process may leave the nucleus in an excited and unstable state, depending on the input energy of the proton and configuration of nucleons. The process is probabilistic, and the average chance of a reaction (the microscopic cross section) may be measured as a function of the projectile, projectile energy and target, either experimentally or by optical model potential calculations. The reaction rate is calculated from the microscopic cross section using the following equation:

$$R = \frac{J}{e} n_t \sigma \cdot 10^{-28} \delta t \quad (1)$$

- R Reaction Rate (reactions per second)
- J Beam current (A)
- n_t Number density of target (atoms per cubic meter)
- σ Microscopic reaction cross section (barns)
- e Elementary charge (1.602177E-19C)
- δt Target thickness (m).

1.4. Radioactive decay

Radioactive decay is the random change in nucleons or energy state of an unstable nucleus. It is impossible to predict when a single nucleus will decay, but the decay of a collection of nuclei is statistical in nature. The radioactivity and number of unstable nuclei at time t can be predicted using the decay constant, λ , for the radioactive isotope. This constant is defined as follows:

$$\lambda = -\frac{N'(t)}{N(t)}. \quad (2)$$

The number of radioactive nuclei $N(t)$ at time t is given by the following equation, where $N(0)$ is the starting number of nuclei:

$$N(t) = N(0) \exp(-t\lambda). \quad (3)$$

The activity $A(t)$ of the radioactive nuclei is predicted at time t by using the following equations, where $N'(t)$ is the change in amount of nuclei with respect to time:

$$A(t) = -N'(t) = \lambda N(t) \quad (4)$$

$$A(t) = \lambda N(0) \exp(-t\lambda). \quad (5)$$

1.5. Bateman equation for radioactive decay

The English mathematician Harry Bateman derived an Eq. (6) to calculate the amount of each isotope in a decay chain, illustrated in Fig. 2, at time t .

$$N_n(t) = \sum_{i=1}^{i=n} \left(\left(\prod_{j=i}^{j=n-1} \lambda_{(ij+1)} \right) \sum_{j=i}^{j=n} \left(\frac{N_{i0} \exp(-\lambda_j t)}{\prod_{p=i, p \neq j}^{p=n} (\lambda_p - \lambda_j)} \right) \right). \quad (6)$$

When a radioactive isotope decays, there may be more than one mode of decay, and this leads to branching factors. Pb-214 only decays via beta decay to Bi-214, giving a branching factor of 1.0, whereas Bi-214 has a 99.979% chance of decaying to Po-214 by beta decay and a 0.021% of emitting an alpha particle and decaying to Tl-210 (branching factors of 0.99979 and 0.00021 respectively) [5].

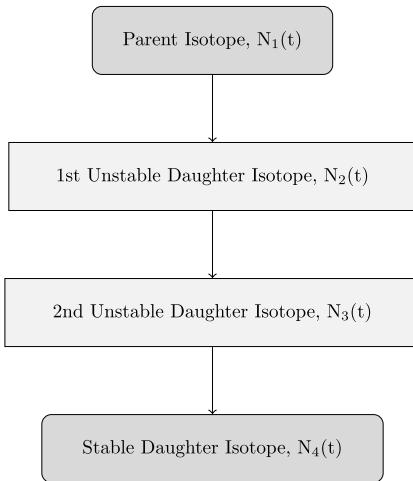


Fig. 2. An example decay chain from an unstable parent isotope, through unstable daughter isotopes ending with a stable daughter isotope.

When a target material is irradiated, there is a source term for transmuted nuclei due to the irradiation. The daughter isotopes of these transmuted isotopes will also be affected by the irradiation and will transmute further, giving a source term for each daughter isotope as a result of the irradiation. Sources for each isotope in the decay chain, and branching factors between a parent isotope and its daughter isotope/s must be accounted for.

The Bateman equation was derived using Laplace transforms, and this same method has been used to develop a modified equation that incorporates branching factors and production rates for each isotope in the decay chain, as illustrated by Fig. 3.

1.6. Laplace transform

Laplace Transforms (7) are a useful mathematical tool, and allow ordinary differential equations to be solved by simple algebraic manipulation in the s domain. Bateman took advantage of Laplace Transforms in deriving his equation, and this is the method that has been taken here as well.

$$F(s) = \int_0^\infty f(t) \exp(-st) dt. \quad (7)$$

1.7. Constructing the differential equations

The first step is to set up differential equations for the parent isotope, unstable daughter isotopes and stable daughter isotope. The parent isotope has a source term, due to production, and a loss term, due to decay. The unstable daughter isotopes have two source terms, from the production by irradiation induced transmutation and the decay of preceding isotopes in the decay chain, and a loss term, due to decay. Finally, the stable daughter that finalizes the decay chain has two source terms (the same as the unstable daughters) but no loss term.

The variables (and vectors) used in these equations are defined as follows:

- $\vec{\lambda}$ vector containing isotope decay constants λ_i
- \vec{b} vector containing isotope to isotope branching factors b_i
- \vec{w} vector containing isotope production rates w_i
- t time at which activity/amount of isotope is measured
- $N_i(0)$ starting amount of the i th isotope

- $N_i(t)$ amount of the i th isotope at time t
- $N'_i(t)$ change in amount of the i th isotope, with respect to time, at time t .

The differential equations for the parent isotope (first isotope), unstable daughter isotopes (i th isotopes) and stable, final, daughter isotope (z th isotope) in the time domain are as follows:

$$N'_1(t) = \omega_1 - \lambda_1 N_1(t) \quad (8)$$

$$N'_i(t) = \omega_i + b_{i-1} \lambda_{i-1} N_{i-1}(t) - \lambda_i N_i(t) \quad (9)$$

$$N'_z(t) = \omega_z + b_{z-1} \lambda_{z-1} N_{z-1}(t). \quad (10)$$

Applying the Laplace Transform to these three differential equations allows them to be manipulated and solved algebraically in the s-domain.

$$N_1(s) = \frac{1}{s + \lambda_1} N_1(0) + \frac{1}{s(s + \lambda_1)} \omega_1 \quad (11)$$

$$N_i(s) = \frac{1}{s(s + \lambda_i)} (\omega_i) + \frac{1}{s + \lambda_i} (b_{i-1} \lambda_{i-1} N_{i-1}(s)) + \frac{1}{s + \lambda_i} N_i(0) \quad (12)$$

$$N_z(s) = \frac{1}{s^2} \omega_z + \frac{1}{s} b_{z-1} \lambda_{z-1} N_{z-1}(s) + \frac{1}{s} N_z(0). \quad (13)$$

1.8. Numerical inversion of the Laplace Transform

The Gaver–Stehfest [6] algorithm was developed in the 1960s and 1970s and is a method of calculating the inverse of a Laplace Transform in the real number domain. It is an easy to implement and reasonably accurate method, although it is an approximation to the real value. A comparison between an analytic and numeric inversion for the unstable isotope Po-218 is discussed at the end of this section (Fig. 4).

$$f(t) \approx f_n(t) = \frac{\ln(2)}{t} \sum_{k=1}^{2n} a_k(n) F(s) \text{ where } n \geq 1, t > 0 \quad (14)$$

$$s = \frac{k \ln(2)}{t} \quad (15)$$

$$a_k(n) = \frac{(-1)^{(n+k)}}{n!} \sum_{j=\text{Floor}(\frac{k+1}{2})} j^{n+1} \binom{n}{j} \binom{2j}{j} \binom{j}{k-j}. \quad (16)$$

The equation for the i th isotope may be calculated by recursively calculating the equations by numeric inversion, starting from the first (parent isotope) and inserting the result into each subsequent recursion until the i th isotope is reached (changing the equations appropriately for the parent, unstable daughter and stable daughter isotopes).

1.9. Analytic solution by partial fraction expansion

The equation for the i th isotope in the s domain can be written in full by substituting the preceding equation until the parent isotope is reached, and this full equation may be rearranged with the production amount of each isotope and starting amount of each isotope in individual terms. Each of these terms is multiplied by a fraction that can be expanded, using partial fractions, and inverted analytically.

This is illustrated with an example unstable isotope, fourth in the decay chain (including the parent isotope):

$$N_4(s) = \frac{1}{(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0)$$

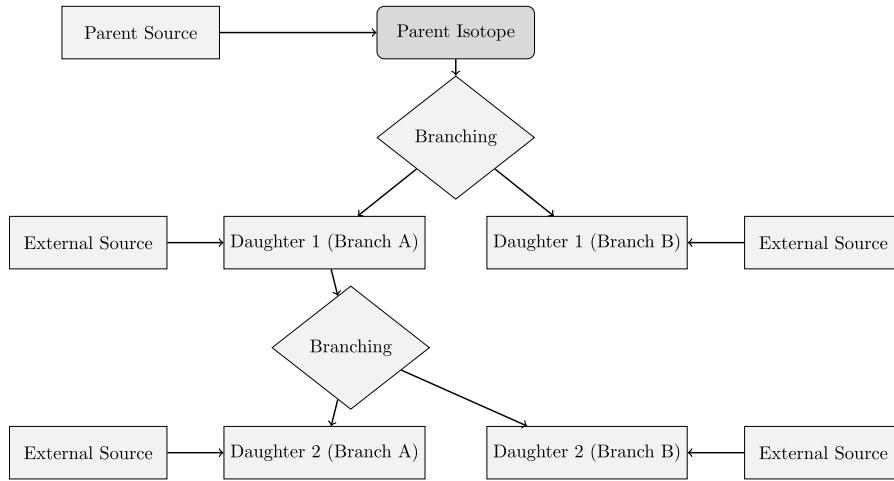


Fig. 3. An example of several decay chains including branching factors and possible external source terms for each isotope on each chain.

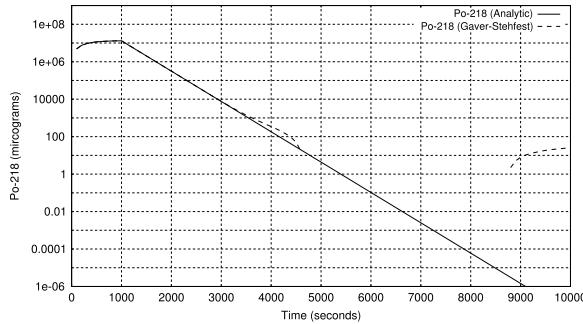


Fig. 4. Decay of Po-218: Analytic and Gaver–Stehfest Calculation [5].

$$\begin{aligned}
 & + \frac{1}{(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\
 & + \frac{1}{(s + \lambda_3)(s + \lambda_4)} b_4 \lambda_3 N_3(0) + \frac{1}{(s + \lambda_4)} N_4(0) \\
 & + \frac{1}{s(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\
 & + \frac{1}{s(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\
 & + \frac{1}{s(s + \lambda_3)(s + \lambda_4)} b_4 \lambda_3 \omega_3 + \frac{1}{s(s + \lambda_4)} \omega_4. \quad (17)
 \end{aligned}$$

An example stable isotope, fourth (last) in the decay chain (including the parent isotope):

$$\begin{aligned}
 N_4(s) = & \frac{1}{s(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\
 & + \frac{1}{s(s + \lambda_2)(s + \lambda_3)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\
 & + \frac{1}{s(s + \lambda_3)} b_4 \lambda_3 N_3(0) + N_4(0) \\
 & + \frac{1}{s^2(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\
 & + \frac{1}{s^2(s + \lambda_2)(s + \lambda_3)} b_3 b_4 \lambda_2 \lambda_3 \omega_2
 \end{aligned}$$

$$+ \frac{1}{s^2(s + \lambda_3)} b_4 \lambda_3 \omega_3 + \frac{1}{s^2} \omega_4. \quad (18)$$

By using partial fraction expansion and standard Laplace Transforms, the set of equations below is used to calculate the amount of the m th isotope in the decay chain, providing the m th isotope is unstable.

$$\begin{aligned}
 N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) &= \sum_{k=1,m} r(k; \vec{\lambda}, \vec{b}) [f(t; k, m, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k] \quad (19)
 \end{aligned}$$

$$r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k, m-1} (b_{i+1} \lambda_i), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \quad (20)$$

$$f(t; k, m, \vec{\lambda}) = (-1)^{m-k} \sum_{i=k, m} \left[\exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (21)$$

$$\begin{aligned}
 g(t; k, m, \vec{\lambda}) = & \frac{1}{\prod_{i=k, m} \lambda_i} + (-1)^{m-k+1} \\
 & \times \sum_{i=k, m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right]. \quad (22)
 \end{aligned}$$

The set of equations below is used to calculate the amount of the m th isotope in the decay chain, where the m th isotope is stable.

$$\begin{aligned}
 N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = & N_m + w_m t + \sum_{k=1, m-1} r(k; \vec{\lambda}, \vec{b}) \\
 & \times [f(t; k, m-1, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k] \quad (23)
 \end{aligned}$$

$$r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k, m-1} (b_{i+1} \lambda_i), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \quad (24)$$

$$\begin{aligned}
 f(t; k, m, \vec{\lambda}) = & \frac{1}{\prod_{i=k, m} \lambda_i} + (-1)^{m-k+1} \\
 & \times \sum_{i=k, m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (25)
 \end{aligned}$$

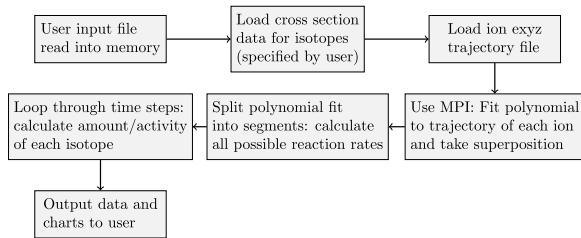


Fig. 5. Flow chart of major processes in the Activity code.

$$g(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k,m} \lambda_i} t + \sum_{i=k,m} \left[\prod_{j=k,m; j \neq i} \lambda_j \right] + (-1)^{m-k+1} \sum_{i=k,m} \left[\frac{1}{\lambda_i^2} \exp(-\lambda_i t) \prod_{j=k,m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right]. \quad (26)$$

1.10. Preference: analytic over numeric

The numeric solution only requires the equation to be solved in the s-domain; the Gaver–Stehfest algorithm performs the inversion. It is worth the extra effort to derive and implement an analytic solution, as the numeric is only an approximation. Examples of the pitfalls of the numeric solution are that it can give negative amounts of an isotope and the difference between the numeric and analytic calculated amounts can become quite large when the isotope decays away to a very small value. Fig. 4 shows the predicted decay of a sample of Po-218 irradiated for 1000 s, and sampled until 10,000 s. In the region between 4000 s and 9000 s the amount from the numeric calculation drops below zero, whereas the analytic calculation remains above zero, as would be expected.

2. Computational methods

The Activity program has been developed in Fortran and takes advantage of MPI (Message Parsing Interface) to speed up calculation times by allowing the use of multiple processes in parallel. It has a self contained maths library, although this could be improved in the future by using optimized maths libraries for certain functions (e.g. matrix operations).

The code was developed on a Debian based distribution of Linux, but it should be supported on other variants of Linux and Unix, and does not require any specialist hardware.

The user is required to prepare an input file that contains the instructions required to perform a calculation. In addition to the input file, the user must provide an EXYZ ion trajectory file output by SRIM. Activity will read in the user input file, and the SRIM and data files listed within, before performing the calculation. Fig. 5 shows a flowchart of the major steps the code performs.

There are various settings in the user input file, but the main ones relating to the simulated experiment are:

- Element composition of target (percentage by mass).
- Beam flux (current), energy, duration and area on target.
- Activity measurement time (end of the “experiment”).
- Material density.
- Target thickness.

Several data files are generated by Activity and, if the user has matplotlib [7], charts will be created too. The most relevant to the user are:

- gammaLines.dat – tally into discrete bins of predicted gamma counts.
- ionTraj.dat – the averaged ion trajectory used in the calculation.
- isotopeActivityFileG.dat – a large data file detailing the activity of every predicted radioactive isotope in the target at user specified times following irradiation.

The charts include:

- activityTop5.png – activity of the top 5 active isotopes as a function of time after irradiation starts.
- gammaLines.png – predicted gamma spectrum expected at the “experiment end time”.

The Activity code uses the equations derived above to calculate the amount and activity of each isotope in the calculation. One problem with the original Bateman equation that also exists in the set of modified Bateman equations is that two different isotopes with the same decay constant will cause a singularity and a halt in the calculation. The activity code loops through all the decay constants in use before it attempts to run the calculation. If any isotope decay constants match they are varied by a small amount relative to the decay constant. It repeats this process until all decay constants are unique before proceeding.

3. Approximations

The accuracy of the Activity code is dependent on the input files provided by the user and the method used to calculate the reaction rates and resulting activity. The TENDL proton database consists of experimental measured cross sections as well as values calculated using the optical model potential. Using the latest database is recommended.

SRIM uses the binary collision approximation to simulate ion transport. It is a well tested code that has been used for many years. One limitation is that the structure of the material is not taken into account. This would have an impact on a user of the Activity program if they were trying to calculate, for example, whether a FCC (face centered cubic) steel would be irradiated differently when compared to a BCC (body centered cubic) steel. The Activity code would determine the activity of the steel as a function of the ion current, ion type and the density, thickness and composition of the steel, not its structure.

This version of the Activity code averages the path of all the SRIM simulated ions, rather than treating each ion differently. This may or may not have an impact on the results. If a new version of the code is developed there would be an option to calculate reaction rates for each individual simulated ion, and a comparison could then be made to the calculations using the averaged path of a set of ions.

The final approximation would be to use the numeric solution to the activity equations, although the analytic solution is forced within the code unless it returns a failed result.

4. Results

A target of high purity Iron was irradiated with 36 MeV protons by the University of Birmingham Scanditronix MC-40 Cyclotron. The target was 0.5 mm thick and was irradiated at a current of 0.5 μ A for 300 s, irradiating approximately 0.25 g of Iron. A high purity Germanium detector was used to measure the gamma peaks three days after irradiation.

The peak that dominated the readings was the 931 keV Cobalt 55 line. After calibrating the detector and adjusting the readings, this peak was measured at 44,300 Bq+/-2000 Bq. The activity of this peak as predicted by the Activity code was 44,565 Bq.

Table 1

Gamma peaks predicted and measured for a 13 MeV ion irradiated sample of Mo.

Gamma energy (keV)	Predicted activity (Bq)	Experimental activity (Bq)
766	4.45E6	5.11E5+/-2.5E4
778	6.14E6	1.36E6+/-6.8E4
812	5.04E6	1.15E6+/-5.8E4
850	6.00E6	1.39E6+/-7.0E4
126	9.33E5	2.10E5+/-1.1E4

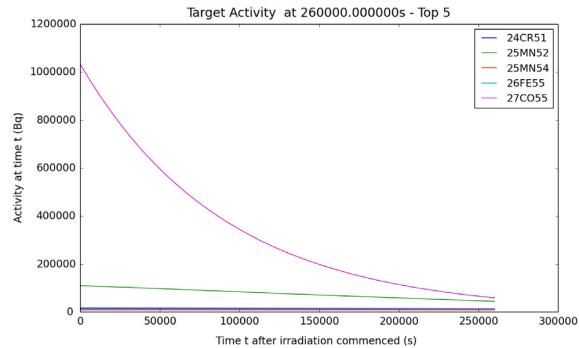


Fig. 6. Sample Activity code output chart for the top five most active isotopes for Iron irradiated by 36 MeV protons.

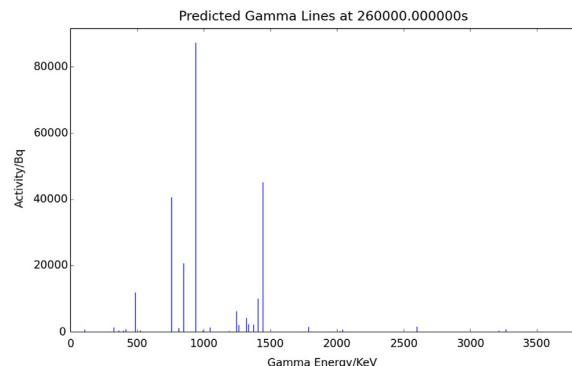


Fig. 7. Sample Activity code output chart for the expected gamma lines to be measured for Iron irradiated by 36 MeV protons.

A target of high purity Molybdenum was irradiated with 13 MeV protons by the University of Birmingham Scanditronix MC-40 Cyclotron. The target was 0.5 mm thick and was irradiated at a current of 5 μ A for 1500 s, irradiating approximately 0.3 g of Molybdenum. A high purity Germanium detector was used to measure the gamma peaks three days after irradiation.

Five peaks were of interest, and these are listed in Table 1.

There was the possibility of the high purity Germanium detector introducing errors due to detector dead time, where by a source that is too active floods the device with gammas. The samples were safe to handle, and did not appear to flood the detector in any way, but there was still the possibility of counts being missed due to dead time. The probabilistic nature of radioactive decay also introduced inherent errors to the experimental activity measurements (see Figs. 6–9).

5. Conclusions

The Activity program is an easy to use Fortran compiled executable that can be built and run, for free, on a Linux computer. It

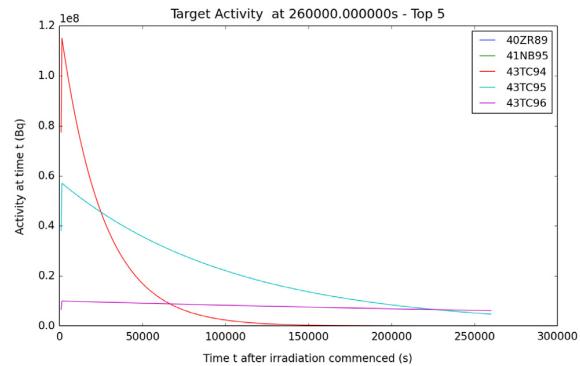


Fig. 8. Sample Activity code output chart for the top five most active isotopes for Molybdenum irradiated by 13 MeV protons.

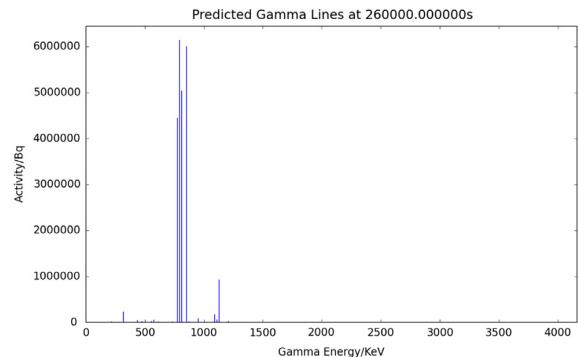


Fig. 9. Sample Activity code output chart for the expected gamma lines to be measured for Molybdenum irradiated by 13 MeV protons.

takes advantage of multi-core processors, and typical calculations take from seconds to minutes.

Using the SRIM code and TENDL database, the code has been used to predict the activity of Iron and Molybdenum targets that have been irradiated with a proton beam. The prediction of the 931 (keV) Cobalt 55 gamma activity of irradiated Iron was very close to the measured value. The five predicted gamma activities for ion irradiated Molybdenum were up to a factor of 5–10 away from the measured activities.

There are a number of improvements that would be considered in a future version of the Activity code. These improvements would include the following:

- A more readable output with less information printed, as too much may confuse the user.
- A file containing the top five radioactive isotopes with their top five gamma lines.
- Individual ion trajectories used to calculate the reaction rates, rather than the average path.
- Experimental ion activation data for a wider range of elements to test the Activity code against.
- Expand to include deuterons (this would require the TENDL deuteron cross section database).

Acknowledgments

We would like to thank and acknowledge the input and advice of the following:

- Dr Chris Cooper and John Hewett for irradiation activation data points.

- The University of Birmingham for providing the funding for this project.

References

- [1] A.J. Koning, D. Rochman, Nucl. Data Sheets 113 (2012).
- [2] A.A. Sonzogni, Nuclear Data Services. www-nds.iaea.org/ndspub/download-endf/ENDF-B-VII.0/decay-index.htm (visited on 08/14/2015).
- [3] M. Herman, M.B. Chadwick, Nucl. Data Sheets 107 (2006) 2931.
- [4] J.P. Biersack, J.F. Ziegler, M.D. Ziegler, Nuclear Instrum. Methods Phys. Res. B 268 (2010) 1818–1823.
- [5] P. Blaise, M. Coste, A. Courcelle, T.D. Huynh, C. Jouanne, P. Leconte, O. Litaize, S. Mengelle, G. Nogure, J.-M. Ruggiri, O. Srot, J. Tommasi, C. Vaglio, J.-F. Vidal, A. Santamarina, D. Bernard, The JEFF-3.1.1 Nuclear Data Library. 2009. ISBN: 978-92-64-99074-6.
- [6] H. Stehfest, Commun. ACM 13 (1970) 47–49.
- [7] J.D. Hunter, Comput. Sci. Eng. 9 (2007) 90–95.

E.4 Accompanying Manual

UNIVERSITY OF BIRMINGHAM



Department of Metallurgy & Materials

Activity Manual

Contents

1	Background	4
1.1	Ion Irradiation	4
1.1.1	Ion Irradiation at the University of Birmingham	4
1.1.2	Simulating Ion Irradiation with SRIM	4
1.1.3	Transmutation of Nuclei by Ion Irradiation	5
1.2	Decay and Activity Equations	6
1.2.1	Radioactive Decay	6
1.2.2	Bateman Equation for Radioactive Decay	6
1.2.3	Laplace Transform	7
1.2.4	Constructing the Differential Equations	7
1.2.5	Numerical Inversion of the Laplace Transform	8
1.2.6	Analytic Solution by Partial Fraction Expansion	9
1.2.7	Preference: Analytic over Numeric	11
1.3	Computational Methods	12
1.3.1	Activity Code	12
1.3.2	Approximations	13
1.3.3	Results	13
2	Installation and Using the Activity Code	17
2.1	Getting Started	17
2.1.1	Prerequisites	17
2.2	Installing Activity	17
2.2.1	Download Source Code	17
2.2.2	Compile Source Code	18
2.3	Input File	18
2.4	Acknowledgements	21
Appendices		22

A Example Input File	23
A.1 Iron 36MeV Proton Beam	23
B Fortran 90 Code	25
B.1 Fortran 90 Implementation of Analytic Method	25

List of Figures

1.1	Proton energy loss in Fe simulated with SRIM [2]	5
1.2	An example decay chain from an unstable parent isotope, through unstable daughter isotopes ending with a stable daughter isotope.	6
1.3	An example of several decay chains including branching factors and possible external source terms for each isotope on each chain.	7
1.4	Decay of Po-218: Analytice and Gaver-Stehfest Calculations [5]	11
1.5	Flow chart of major processes in the Activity code	12
1.6	Sample Activity code output chart for the top five most active isotopes for Iron irradiated by 36MeV protons.	14
1.7	Sample Activity code output chart for the expected gamma lines to be measured for Iron irradiated by 36MeV protons.	15
1.8	Sample Activity code output chart for the top five most active isotopes for Molybdenum irradiated by 13MeV protons.	15
1.9	Sample Activity code output chart for the expected gamma lines to be measured for Molybdenum irradiated by 13MeV protons.	16

Chapter 1

Background

1.1 Ion Irradiation

A computer program, Activity, was developed to predict the activity and gamma lines of materials irradiated with an ion beam. It uses the TENDL[1] proton reaction cross section database, the Stopping and Range of Ions in Matter (SRIM)[2] code, a Nuclear Data Services (NDS) radioactive decay database [3] and an ENDF gamma decay database [4]. An extended version of Bateman's equation is used to calculate the activity at time t, and this equation is solved analytically, with the option to also solve by numeric inverse Laplace transform as a failsafe. The program outputs the expected activity and gamma lines of the activated material.

1.1.1 Ion Irradiation at the University of Birmingham

The Scanditronix MC-40 Cyclotron is used at the University of Birmingham to create a beam of protons or other light ions. The energies of these ions are typically between 10 MeV and 60 MeV with beam currents ranging up to 50 microamps (3.1×10^{14} protons per second). Target materials are irradiated by this cyclotron for a number of reasons, including purposely creating radioactive isotopes for the nearby Queen Elizabeth Hospital, investigating ion irradiation damage and emulating neutron irradiation.

The Cyclotron is usually used to create radioactive isotopes for medical use, but an additional beam line has been devoted to material science investigations into radiation damage. While the creation of radioactive isotopes is desired in some cases, material being tested for radiation damage should preferably have low levels of radioactivity.

It is expensive to arrange the irradiation of target materials by high energy neutrons sources, whereas it is relatively inexpensive to irradiate using an ion beam on the MC-40 Cyclotron. The energies can be controlled, and a set dose at a single energy, or a range of energies, can be precisely deposited in the target material.

The Activity code discussed here was developed to calculate the activity of a target material irradiated by a proton beam. It has been developed in Fortran and uses data from the TENDL-2013 proton cross section database, SRIM ion transport code and NDS radioactive decay database.

1.1.2 Simulating Ion Irradiation with SRIM

A package of ion transport codes, SRIM, is freely available to download and use to investigate the transport of ions through matter. SRIM uses the binary collision approximation (BCA) to simulate the passage of ions in a material. It is an approximate method, and one key restriction is that it does not take into account the structure of the material, and this approximation is therefore also imposed on the Activity code.

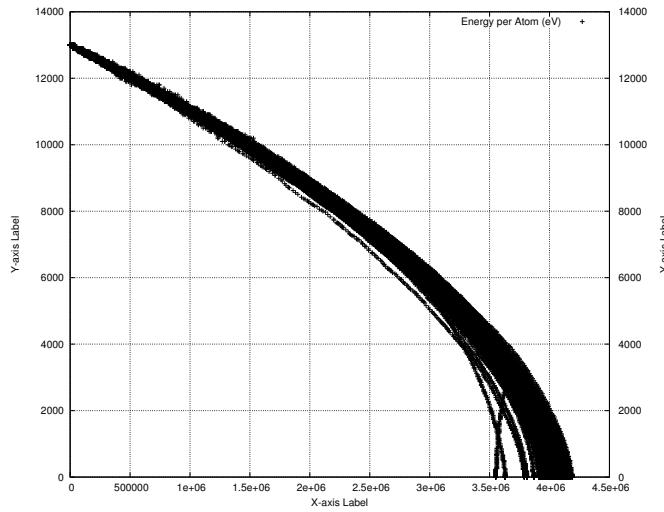


Figure 1.1: Proton energy loss in Fe simulated with SRIM [2]

One file that SRIM creates is of importance to the Activity code, and that is the trajectory file that contains the energy and x,y,z co-ordinate data points for simulated ions moving through matter. Figure 1.1 shows the trajectory of protons passing through an Iron target, and it is this set of data points (together with the cross section database) that the Activity code uses to calculate the reaction rates for the transmutation of nuclei in the target. At higher energies, the ions slow as they lose energy due to electronic stopping, but as the ion energy drops the mechanism of loss through nuclear collisions becomes important. The spreading of ion depths at lower energies is a result of the higher momentum transfer during nuclear collisions, as can be seen in Figure 1.1.

1.1.3 Transmutation of Nuclei by Ion Irradiation

Considering a simplified nuclear potential well, energetic protons approaching a nucleus may overcome the coulomb potential barrier. They are captured by the nucleus and held within the potential well by the strong nuclear force. This process may leave the nucleus in an excited and unstable state, depending on the input energy of the proton and configuration of nucleons. The process is probabilistic, and the average chance of a reaction (the microscopic cross section) may be measured as a function of the projectile, projectile energy and target, either experimentally or by optical model potential calculations. The reaction rate is calculated from the microscopic cross section using the following equation:

$$R = \frac{J}{e} n_t \sigma \cdot 10^{28} \delta t \quad (1.1)$$

- R Reaction Rate (reactions per second)
- J Beam current (A)
- n_t Number density of target (atoms per cubic metre)
- σ Microscopic reaction cross section (barns)
- e Elementary charge (1.602177E-19C)
- δT Target thickness (m)

1.2 Decay and Activity Equations

1.2.1 Radioactive Decay

Radioactive decay is the random change in nucleons or energy state of an unstable nucleus. It is impossible to predict when a nucleus will decay, but the decay of a collection of nuclei is statistical in nature. The radioactivity and number of unstable nuclei at time t can be predicted using the decay constant λ for the radioactive isotope. This constant is defined as follows:

$$\lambda = -\frac{N'(t)}{N(t)} \quad (1.2)$$

The number of radioactive nuclei $N(t)$ at time t is given by the following equation, where $N(0)$ is the starting number of nuclei:

$$N(t) = N(0) \exp(-t\lambda) \quad (1.3)$$

The activity $A(t)$ of the radioactive nuclei is predicted at time t by using the following equations, where $N'(t)$ is the change in amount of nuclei with respect to time:

$$A(t) = -N'(t) = \lambda N(t) \quad (1.4)$$

$$A(t) = \lambda N(0) \exp(-t\lambda) \quad (1.5)$$

1.2.2 Bateman Equation for Radioactive Decay

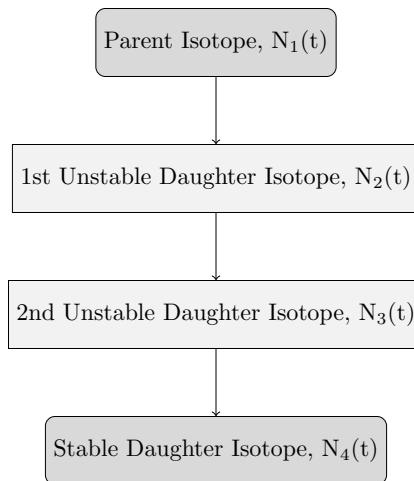


Figure 1.2: An example decay chain from an unstable parent isotope, through unstable daughter isotopes ending with a stable daughter isotope.

The English mathematician Harry Bateman derived an equation to calculate the amount of each isotope in a decay chain, illustrated in Figure 1.2, at time t .

$$N_n(t) = \sum_{i=1}^{i=n} \left(\left(\prod_{j=i}^{j=n-1} \lambda_{(ij+1)} \right) \sum_{j=i}^{j=n} \left(\frac{N_{i0} \exp(-\lambda_j t)}{\prod_{p=i, p \neq j}^p (\lambda_p - \lambda_j)} \right) \right) \quad (1.6)$$

When a radioactive isotope decays, there may be more than one mode of decay, and this leads to branching factors. Pb-214 only decays via beta decay to Bi-214, giving a branching factor of 1.0, whereas Bi-214 has a 99.979% chance of decaying to Po-214 by beta decay and a 0.021% of emitting an alpha particle and decaying to Tl-210 (branching factors of 0.99979 and 0.00021 respectively) [5].

When a target material is irradiated, there is a source term for transmuted nuclei due to the irradiation. The daughter isotopes of these transmuted isotopes will also be affected by the irradiation and will transmute further, giving a source term for each daughter isotope as a result of the irradiation. Sources for each isotope in the decay chain, and branching factors between a parent isotope and its daughter isotope/s must be accounted for.

The Bateman equation was derived using Laplace transforms, and this same method has been used to develop a modified equation that incorporates branching factors and production rates for each isotope in the decay chain, as illustrated by Figure 1.3.

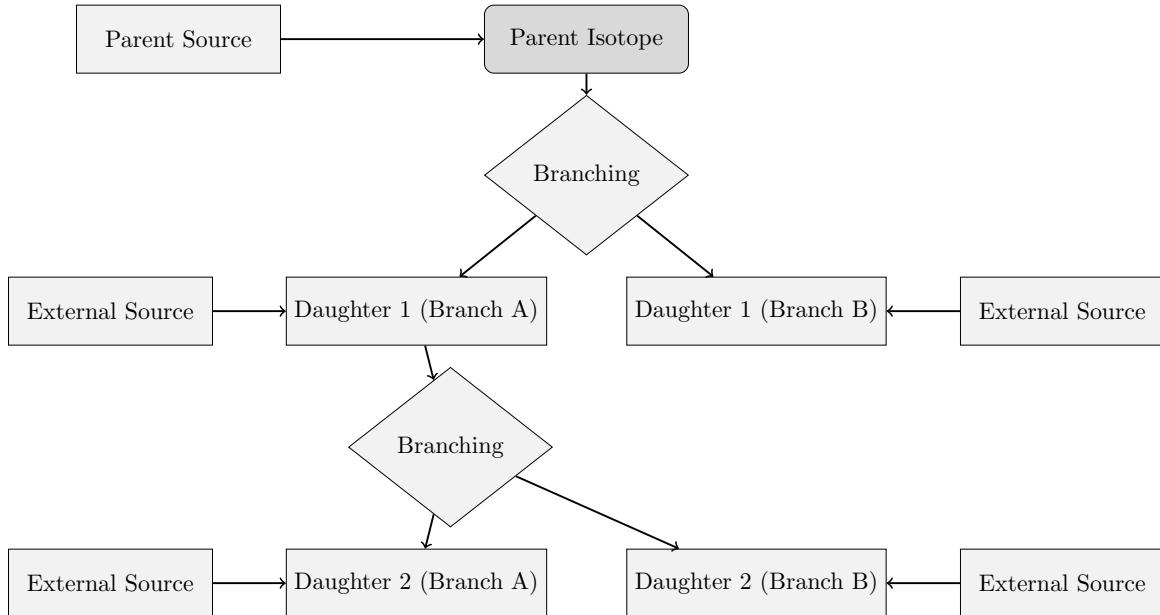


Figure 1.3: An example of several decay chains including branching factors and possible external source terms for each isotope on each chain.

1.2.3 Laplace Transform

Laplace Transforms are a useful mathematical tool, and allow ordinary differential equations to be solved by simple algebraic manipulation in the s domain. Bateman took advantage of Laplace Transforms in deriving his equation, and this is the method that has been taken here as well.

$$F(s) = \int_0^{\infty} f(t) \exp(-st) dt \quad (1.7)$$

1.2.4 Constructing the Differential Equations

The first step is to set up differential equations for the parent isotope, unstable daughter isotopes and stable daughter isotope. The parent isotope has a source term, due to production, and a loss term, due to decay. The unstable daughter isotopes have two source terms, from the production by irradiation induced transmutation

and the decay of preceding isotopes in the decay chain, and a loss term, due to decay. Finally, the stable daughter that finalizes the decay chain has two source terms (the same as the unstable daughters) but no loss term.

The variables (and vectors) used in these equations are defined as follows:

- $\vec{\lambda}$ vector containing isotope decay constants λ_i
- \vec{b} vector containing isotope to isotope branching factors b_i
- \vec{w} vector containing isotope production rates w_i
- t time at which activity/amount of isotope is measured
- $N_i(0)$ starting amount of the i th isotope
- $N_i(t)$ amount of the i th isotope at time t
- $N'_i(t)$ change in amount of the i th isotope, with respect to time, at time t

The differential equations for the parent isotope (first isotope), unstable daughter isotopes (i th isotopes) and stable, final, daughter isotope (z th isotope) in the time domain are as follows:

$$N'_1(t) = \omega_1 - \lambda_1 N_1(t) \quad (1.8)$$

$$N'_i(t) = \omega_i + b_{i-1} \lambda_{i-1} N_{i-1}(t) - \lambda_i N_i(t) \quad (1.9)$$

$$N'_z(t) = \omega_z + b_{z-1} \lambda_{z-1} N_{z-1}(t) \quad (1.10)$$

Applying the Laplace Transform to these three differential equations allows them to be manipulated and solved algebraically in the s-domain.

$$N_1(s) = \frac{1}{s + \lambda_1} N_1(0) + \frac{1}{s(s + \lambda_1)} \omega_1 \quad (1.11)$$

$$N_i(s) = \frac{1}{s(s + \lambda_i)} (\omega_i) + \frac{1}{s + \lambda_i} (b_{i-1} \lambda_{i-1} N_{i-1}(s)) + \frac{1}{s + \lambda_i} N_i(0) \quad (1.12)$$

$$N_z(s) = \frac{1}{s^2} \omega_z + \frac{1}{s} b_{z-1} \lambda_{z-1} N_{z-1}(s) + \frac{1}{s} N_z(0) \quad (1.13)$$

1.2.5 Numerical Inversion of the Laplace Transform

The Gaver-Stehfest[6] algorithm was developed in the 1960s and 1970s and is a method of calculating the inverse of a Laplace Transform in the real number domain. It is an easy to implement and reasonably accurate method, although it is an approximation to the real value. A comparison between an analytic and numeric inversion for the unstable isotope Po-218 is discussed at the end of this section.

$$f(t) \approx f_n(t) = \frac{\ln(2)}{t} \sum_{k=1}^{2n} a_k(n) F(s) \text{ where } n \geq 1, t > 0 \quad (1.14)$$

$$s = \frac{k \ln(2)}{t} \quad (1.15)$$

$$a_k(n) = \frac{(-1)^{(n+k)}}{n!} \sum_{j=\text{Floor}(\frac{k+1}{2})} j^{n+1} \binom{n}{j} \binom{2j}{j} \binom{j}{k-j} \quad (1.16)$$

The equation for the i th isotope may be calculated by recursively calculating the equations by numeric inversion, starting from the first (parent isotope) and inserting the result into each subsequent recursion until the i th isotope is reached (changing the equations appropriately for the parent, unstable daughter and stable daughter isotopes).

1.2.6 Analytic Solution by Partial Fraction Expansion

The equation for the i th isotope in the s domain can be written in full by substituting the preceding equation until the parent isotope is reached, and this full equation may be rearranged with the production amount of each isotope and starting amount of each isotope in individual terms. Each of these terms is multiplied by a fraction that can be expanded, using partial fractions, and inverted analytically.

This is illustrated with an example unstable isotope, fourth in the decay chain (including the parent isotope):

$$\begin{aligned} N_4(s) = & \frac{1}{(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\ & + \frac{1}{(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\ & + \frac{1}{(s + \lambda_3)(s + \lambda_4)} b_4 \lambda_3 N_3(0) \\ & + \frac{1}{(s + \lambda_4)} N_4(0) \\ & + \frac{1}{s(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\ & + \frac{1}{s(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\ & + \frac{1}{s(s + \lambda_3)(s + \lambda_4)} b_4 \lambda_3 \omega_3 \\ & + \frac{1}{s(s + \lambda_4)} \omega_4 \end{aligned} \quad (1.17)$$

An example stable isotope, fourth (last) in the decay chain (including the parent isotope):

$$\begin{aligned}
N_4(s) = & \frac{1}{s(s+\lambda_1)(s+\lambda_2)(s+\lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\
& + \frac{1}{s(s+\lambda_2)(s+\lambda_3)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\
& + \frac{1}{s(s+\lambda_3)} b_4 \lambda_3 N_3(0) \\
& + N_4(0) \\
& + \frac{1}{s^2(s+\lambda_1)(s+\lambda_2)(s+\lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\
& + \frac{1}{s^2(s+\lambda_2)(s+\lambda_3)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\
& + \frac{1}{s^2(s+\lambda_3)} b_4 \lambda_3 \omega_3 \\
& + \frac{1}{s^2} \omega_4
\end{aligned} \tag{1.18}$$

By using partial fraction expansion and standard Laplace Transforms, the set of equations below is used to calculate the amount of the m th isotope in the decay chain, providing the m th isotope is unstable.

$$N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = \sum_{k=1, m} r(k; \vec{\lambda}, \vec{b}) [f(t; k, m, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k] \tag{1.19}$$

$$r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k, m-1} (b_{i+1} \lambda_i), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \tag{1.20}$$

$$f(t; k, m, \vec{\lambda}) = (-1)^{m-k} \sum_{i=k, m} \left[\exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \tag{1.21}$$

$$g(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k, m} \lambda_i} + (-1)^{m-k+1} \sum_{i=k, m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \tag{1.22}$$

The set of equations below is used to calculate the amount of the m th isotope in the decay chain, where the m th isotope is stable.

$$N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = N_m + w_m t + \sum_{k=1, m-1} r(k; \vec{\lambda}, \vec{b}) [f(t; k, m-1, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k] \tag{1.23}$$

$$r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k, m-1} (b_{i+1} \lambda_i), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \tag{1.24}$$

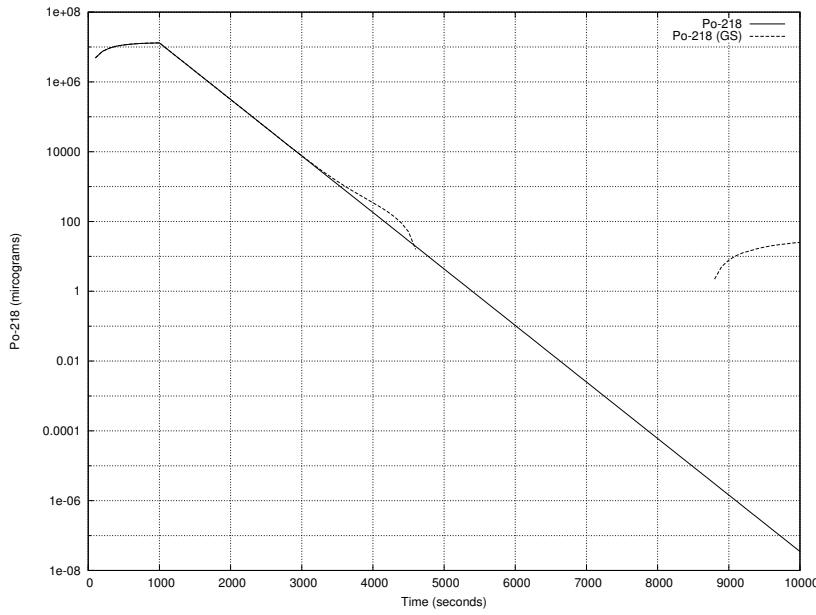


Figure 1.4: Decay of Po-218: Analytic and Gaver-Stehfest Calculations [5]

$$f(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k,m} \lambda_i} + (-1)^{m-k+1} \sum_{i=k,m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k,m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (1.25)$$

$$g(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k,m} \lambda_i} t + \frac{\sum_{i=k,m} \left[\prod_{j=k,m; j \neq i} \lambda_j \right]}{\prod_{i=k,m} \lambda_i^2} + (-1)^{m-k+1} \sum_{i=k,m} \left[\frac{1}{\lambda_i^2} \exp(-\lambda_i t) \prod_{j=k,m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (1.26)$$

1.2.7 Preference: Analytic over Numeric

The numeric solution only requires the equation to be solved in the s-domain; the Gaver-Stehfest algorithm performs the inversion. It is worth the extra effort to derive and implement an analytic solution, as the numeric is only an approximation. Examples of the pitfalls of the numeric solution are that it can give negative amounts of an isotope and the difference between the numeric and analytic calculated amounts can become quite large when the isotope decays away to a very small value. Figure 1.4 shows the predicted decay of a sample of Po-218 irradiated for 1,000s, and sampled until 10,000s. In the region between 4,000s and 9,000s the amount from the numeric calculation drops below zero, whereas the analytic calculation remains above zero, as would be expected.

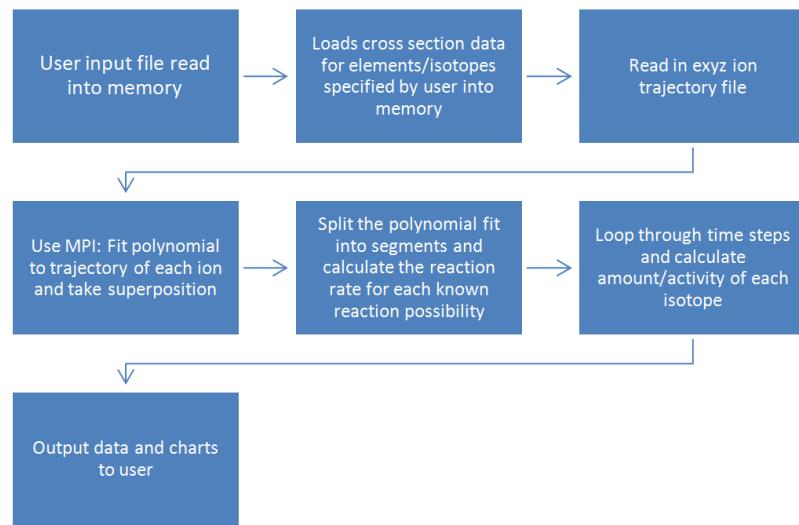


Figure 1.5: Flow chart of major processes in the Activity code

1.3 Computational Methods

1.3.1 Activity Code

The Activity program has been developed in Fortran and takes advantage of MPI (Message Parsing Interface) to speed up calculation times by allowing the use of multiple processes in parallel. It has a self contained maths library, although this could be improved in the future by using optimised maths libraries for certain functions (e.g. matrix operations).

The code was developed on a Debian version of Linux, but it should be supported on other variants of Linux and Unix, and does not require any specialist hardware.

The user is required to prepare an input file that contains the instructions required to perform a calculation. In addition to the input file, the user must provide an EXYZ ion trajectory file output by SRIM. Activity will read in the user input file, and the SRIM and data files listed within, before performing the calculation. Figure 1.5 shows a flowchart of the major steps the code performs.

There are various settings in the user input file, but the main ones relating to the simulated experiment are:

- element composition of target (percentage by mass)
- beam flux (current), energy, duration and area on target
- activity measurement time (end of the “experiment”)
- material density
- target thickness

Several data files are generated by Activity and, if the user has matplotlib [7], charts will be created too. The most relevant to the user are:

- gammaLines.dat - tally into discrete bins of predicted gamma counts

- ionTraj.dat - the averaged ion trajectory used in the calculation
- isotopeActivityFileG.dat - a large data file detailing the activity of every predicted radioactive isotope in the target at user specified times following irradiation

The charts include:

- activityTop5.png - activity of the top 5 active isotopes as a function of time after irradiation starts
- gammaLines.png - predicted gamma spectrum expected at the “experiment end time”

The Activity code uses the equations derived above to calculate the amount and activity of each isotope in the calculation. One problem with the original Bateman equation that also exists in the set of modified Bateman equations is that two different isotopes with the same decay constant will cause a singularity and a halt in the calculation. The activity code loops through all the decay constants in use before it attempts to run the calculation. If any isotope decay constants match they are varied by a small amount relative to the decay constant. It repeats this process until all decay constants are unique before proceeding.

1.3.2 Approximations

The accuracy of the Activity code is dependant on the input files provided by the user and the method used to calculate the reaction rates and resulting activity. The TENDL proton database consists of experimental measured cross sections as well as values calculated using the optical model potential. Using the latest database is recommended.

SRIM uses the binary collision approximation to simulate ion transport. It is a well tested code that has been used for many years. One limitation is that the structure of the material is not taken into account. This would have an impact on a user of the Activity program if they were trying to calculate, for example, whether a FCC (face centered cubic) steel would be irradiated differently when compared to a BCC (body centered cubic) steel. The Activity code would determine the activity of the steel as a function of the ion current, ion type and the density, thickness and composition of the steel, not its structure.

This version of the Activity code averages the path of all the SRIM simulated ions, rather than treating each ion differently. This may or may not have an impact on the results. If a new version of the code is developed there would be an option to calculate reaction rates for each individual simulated ion, and a comparison could then be made to the calculations using the averaged path of a set of ions.

The final approximation would be to use the numeric solution to the activity equations, although the analytic solution is forced within the code unless it returns a failed result.

1.3.3 Results

A target of high purity Iron was irradiated with 36 MeV protons by the University of Birmingham Scanditronix MC-40 Cyclotron. The target was 0.5mm thick and was irradiated at a current of 0.5 micro Amps for 300 seconds, irradiating approximately 0.25g of Iron. A high purity Germanium detector was used to measure the gamma peaks three days after irradiation.

The peak that dominated the readings was the 931 keV Cobalt 55 line. After calibrating the detector and adjusting the readings, this peak was measured at 44,300Bq+/-2,000Bq. The activity of this peak as predicted by the Activity code was 44,565Bq.

A target of high purity Molybdenum was irradiated with 13 MeV protons by the University of Birmingham Scanditronix MC-40 Cyclotron. The target was 0.5mm thick and was irradiated at a current of 5 micro Amps

Table 1.1: Gamma peaks predicted and measured for a 13 MeV ion irradiated sample of Mo

Gamma Energy (keV)	Predicted Activity (Bq)	Experimental Activity (Bq)
766	4.45E6	5.11E5 +/- 2.5E4
778	6.14E6	1.36E6 +/- 6.8E4
812	5.04E6	1.15E6 +/- 5.8E4
850	6.00E6	1.39E6 +/- 7.0E4
126	9.33E5	2.10E5 +/- 1.1E4

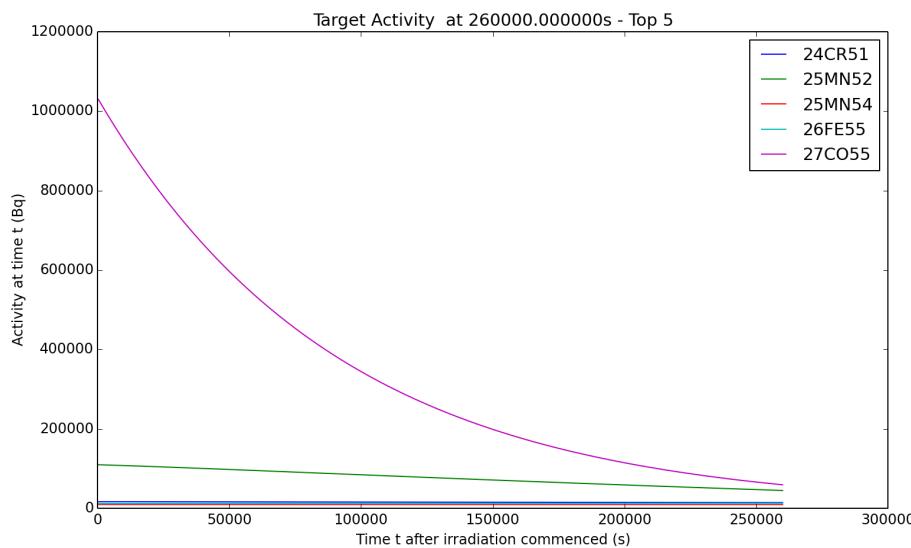


Figure 1.6: Sample Activity code output chart for the top five most active isotopes for Iron irradiated by 36MeV protons.

for 1500 seconds, irradiating approximately 0.3g of Molybdenum. A high purity Germanium detector was used to measure the gamma peaks three days after irradiation.

Five peaks were of interest, and these are listed in Table 1.1.

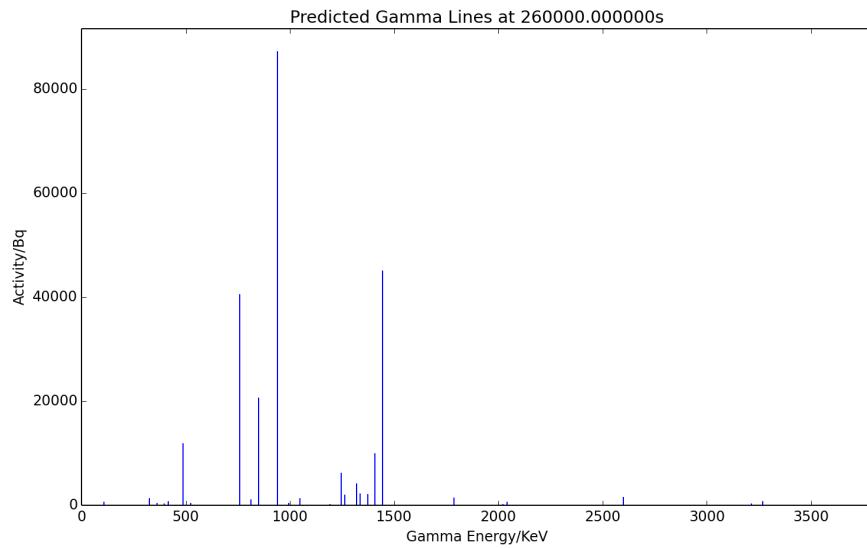


Figure 1.7: Sample Activity code output chart for the expected gamma lines to be measured for Iron irradiated by 36MeV protons.

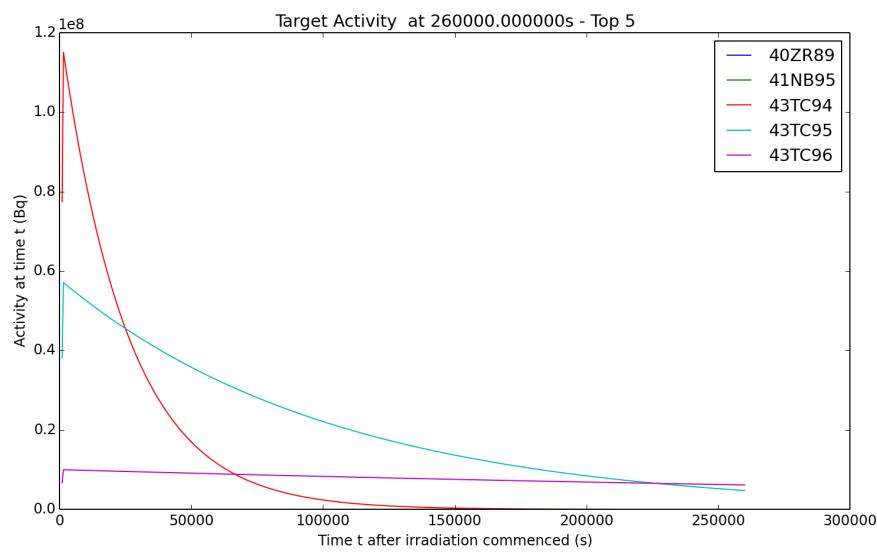


Figure 1.8: Sample Activity code output chart for the top five most active isotopes for Molybdenum irradiated by 13MeV protons.

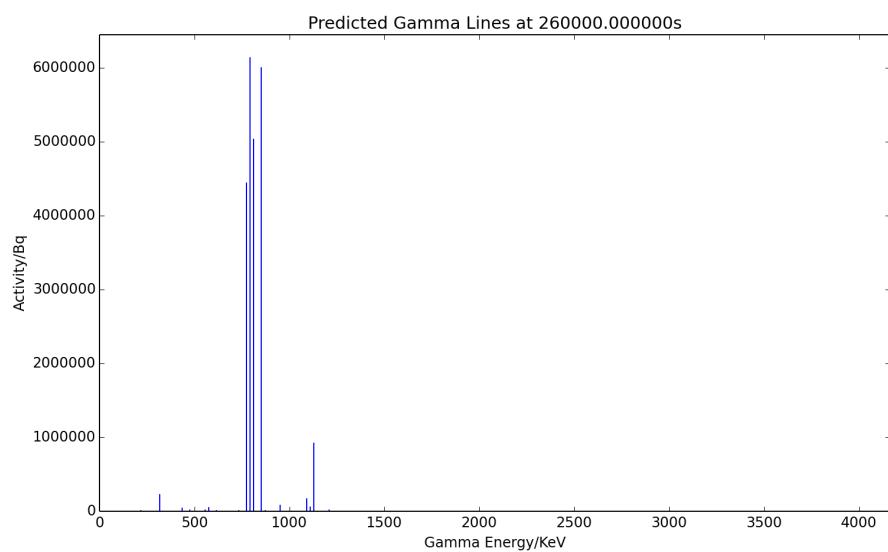


Figure 1.9: Sample Activity code output chart for the expected gamma lines to be measured for Molybdenum irradiated by 13MeV protons.

Chapter 2

Installation and Using the Activity Code

2.1 Getting Started

2.1.1 Prerequisites

This code was designed to run on Linux and has been developed and tested on the Debian based distribution Mint. It has not been tested on any other versions of Linux or Unix, nor has it been tested on cygwin.

The minimal requirements of the code are:

- 250MB per process + 200MB overhead
- 1GB Hard Drive space
- Ideally a modern multicore processor
- Fortran90
- OpenMPI

Optional:

- GNUPlot
- Python & MatPlotLib

2.2 Installing Activity

2.2.1 Download Source Code

The most recent source code is available for download from github. It may be downloaded and extracted using the terminal and the commands in listing 1.

Listing 2.1: bash

```
1 cd ~
2 mkdir -p ~/activity/tar
3 cd ~/activity/tar
4 wget https://github.com/BenPalmer1983/activity/raw/master/activity.tar.gz
5 tar -xzvf activity.tar.gz
```

2.2.2 Compile Source Code

Once the source files have been downloaded and extracted, run the install script then remove the downloaded files (if you need to free up space). The terminal commands are given in listing 2. Providing the prerequisites are available, the process will only take a few minutes (with a reasonable Internet connection).

Listing 2.2: bash

```
1 cd ~/activity/tar
2 ./install.sh #choose installation directory when prompted; default is ~/activity
3 rm -R ~/activity/tar
```

The activity code is now installed and the activity.x binary may be executed on the terminal. If the terminal fails to find the binary, the .bashrc or .bash_profile file must be edited, or a symlink created from the bin file to a valid and accessible bin directory.

2.3 Input File

The input file sets the target starting composition, the beam and simulated experiment settings as well as the paths to the required data files.

#elements

The elements keyword instructs the code to read in the composition of the target material. The elements and their percentage by weight are listed under the keyword.

Listing 2.3: Input File

```
1 #elements
2 Fe 72
3 Cr 18
4 Ni 8
5 Mg 2
```

#isotopes #decaymodes #gammaenergies #xsfiles

Four sets of data files are distributed with the code, and the paths to these files are defined underneath each keyword. N.B. the path for the xsfiles is to the directory that holds all the cross section data files, rather than each individual file.

Listing 2.4: Input File

```
1 #isotopes
2 "/home/ben/activity/data/isotopes.txt"
3 #decaymodes
4 "/home/ben/activity/data/decaymodes.txt"
```

```
5 #gammaenergies
6 "/home/ben/activity/data/gammaenergies.txt"
7 #xsfiles
8 "/home/ben/activity/data/xs"
```

```
#trajfile
```

The SRIM exyz file used for the calculation is pointed to under this keyword.

Listing 2.5: Input File

```
1 #trajfile
2 "/home/ben/activity/examples/Fe36MeV.exyz"
```

```
#polyfitorder #integrationgranularity
```

The code calculates the reaction rate of each projectile and target nucleus by first fitting a polynomial to the energy/depth data file. This polynomial gives an average path for the projectiles travelling through the target, allowing E(x) to be calculated quickly. The order of the polynomial is selected here, and a 5th order shold give a reasonable fit. The integration granularity keyword determines how many sections the polynomial is split up into. The energy, cross section for each section and the beam settings are used to calculate the reaction rate for each section, and these are all summed up to give the overall reaction rate.

Listing 2.6: Input File

```
1 #polyfitorder
2 5
3 #integrationgranularity
4 10
```

```
#beamflux #beamenergy #beamduration #beamarea
```

These keywords are self explanatory and control the beam settings used in the calculation.

Listing 2.7: Input File

```
1 #beamflux
2 0.5 uA
3 #beamenergy
4 36 MeV
5 #beamduration
6 300 s
7 #beamarea
8 100 mm2
```

```
#amtime #timestep
```

The time at which the activity of the sample is measured is set by the #amtime keyword. The Activity code calculates the activity of each radioactive isotope in the calculation from the time the beam starts until the activity measured time; and the intervals at which these calculations are made is determined by the #timestep keyword.

Listing 2.8: Input File

```
1 #amtime
2 260000 s
```

```
3 #timestep  
4 1000 s
```

#projectile

The current version only supports protons as the projectile, therefore the only possible atomic and mass number combination for a projectile is 1 1.

Listing 2.9: Input File

```
1 #projectile  
2 1 1
```

#targetthickness #materialdensity

Both the target thickness and density of the material it is made from are input with these keywords.

Listing 2.10: Input File

```
1 #targetthickness  
2 0.5 mm  
3 #materialdensity  
4 8000 kgm3
```

#vpi

This keyword is only being used for a feature currently under testing, so the keyword can be ommited from the input file.

Listing 2.11: Input File

```
1 #vpi  
2 60.2
```

#individualisotopeactivity #verboseterminal

If the #individualisotopeactivity keyword is followed by yes; additional data files are output containing the activity of each individual isotope in the calulcation. #verboseterminal followed by yes will output more verbosely to the terminal.

Listing 2.12: Input File

```
1 #individualisotopeactivity  
2 yes  
3 #verboseterminal  
4 yes
```

#targetdpa

This keyword is only being used for a feature currently under testing, so the keyword can be ommited from the input file.

Listing 2.13: Input File

```
1 #targetdpa  
2 0.0
```

```
#gammachartresolution
```

The gamma chart is created by tallying the gamma values into bins, and this figure specifies the resolution(number of bins) used to create the output chart.

Listing 2.14: Input File

```
1 #gammachartresolution  
2 200
```

2.4 Acknowledgements

We would like to thank and acknowledge the input and advice of the following:

- Dr Chris Cooper and John Hewett for irradiation activation data points.
- The University of Birmingham for providing the funding for this project.

Appendices

Appendix A

Example Input File

A.1 Iron 36MeV Proton Beam

Listing A.1: Fe36MeV.in

```
1 #elements
2 Fe 100
3 #isotopes
4 "/home/ben/activity/data/isotopes.txt"
5 #decaymodes
6 "/home/ben/activity/data/decaymodes.txt"
7 #gammaenergies
8 "/home/ben/activity/data/gammaenergies.txt"
9 #xsfiles
10 "/home/ben/activity/data/xs"
11 #trajfile
12 "/home/ben/activity/examples/Fe36MeV.exyz"
13 #polyfitorder
14 5
15 #integrationgranularity
16 10
17 #beamflux
18 0.5 uA
19 #beamenergy
20 36 MeV
21 #beadmduration
22 300 s
23 #beamarea
24 100 mm2
25 #amtime
26 260000 s
27 #timestep
28 1000 s
29 #projectile
30 1 1
31 #targetthickness
32 0.5 mm
33 #materialdensity
34 8000 kgm3
```

```
35 #vpi
36 60.2
37 #individualisotopeactivity
38 yes
39 #verboseterminal
40 yes
41 #targetdpa
42 0.0
43 #gammachartresolution
44 200
```

Appendix B

Fortran 90 Code

B.1 Fortran 90 Implementation of Analytic Method

Listing B.1: Fortran 90

```
1 Type :: decayChainObj
2   Real(kind=DoubleReal) :: time = 0.0D0
3   !Real(kind=DoubleReal) :: productionRate = 0.0D0
4   Integer(kind=StandardInteger) :: isotopes
5   Character(Len=16), Dimension(1:100) :: label
6   Real(kind=DoubleReal), Dimension(1:100) :: productionRate = 0.0D0
7   Real(kind=DoubleReal), Dimension(1:100) :: branchFactor = 1.0D0 ! from isotope parent
8   Real(kind=DoubleReal), Dimension(1:100) :: decayConstant = -1.0D0 ! negative for stable
9   Real(kind=DoubleReal), Dimension(1:100) :: halfLife = -1.0D0 ! negative for stable
10  Real(kind=DoubleReal), Dimension(1:100) :: amountStart = 0.0D0
11  Real(kind=DoubleReal), Dimension(1:100) :: amountEnd = 0.0D0
12 End Type
13
14 Subroutine CalcIsotopeChain(decayChain)
15 ! Uses inverse laplace transform to calculate isotope amounts at time t (after time = 0)
16 ! t time in seconds after t=0
17 ! w production rate of parent isotope
18 ! isotope chain data
19 Implicit None ! Force declaration of all variables
20 ! Vars In/Out
21 Type(decayChainObj) :: decayChain
22 ! Vars Private
23   Integer(kind=StandardInteger) :: i
24   Real(kind=DoubleReal) :: t, nEnd
25   Real(kind=DoubleReal), Dimension(1:100) :: W ! Production Rate
26   Real(kind=DoubleReal), Dimension(1:100) :: L ! Lambda
27   Real(kind=DoubleReal), Dimension(1:100) :: N ! Starting number of atoms
28   Real(kind=DoubleReal), Dimension(1:100) :: B ! Exp
29 ! Complete decay chain data
30   Call decayChainComplete(decayChain)
31 ! store input in shortned name arrays to make equations clearer
32   t = decayChain%time
33   Do i=1,decayChain%isotopes
34     W(i) = decayChain%productionRate(i)
```

```

35      L(i) = decayChain%decayConstant(i)
36      B(i) = decayChain%branchFactor(i)
37      N(i) = decayChain%amountStart(i)
38      End Do
39 ! Break infinities
40      Call DecayBreakInfinities(L,decayChain%isotopes)
41 ! Run analytic calculations
42 ! Loop through isotopes
43 ! Using L-1(1/(q+ps) = (1/p)*exp(-1*(q/p)*t) and partial fractions
44      Do i=1,decayChain%isotopes
45          nEnd = CalcIsotopeChainCalc(t,i,W,L,N,B)
46          decayChain%amountEnd(i) = nEnd
47      End Do
48 End Subroutine CalcIsotopeChain
49
50 Subroutine decayChainComplete(decayChain)
51 ! Completes the decay chain object
52 Implicit None ! Force declaration of all variables
53 ! Vars In/Out
54 Type(decayChainObj) :: decayChain
55 ! Vars Private
56 Integer(kind=StandardInteger) :: i, n
57 n = 0
58 Do i=1,100
59     n = n + 1
60     If(decayChain%decayConstant(i).eq.-1.0D0.and.decayChain%halfLife(i).gt.0.0D0)Then
61 ! complete decay constant from half life
62         decayChain%decayConstant(i) = lnTwo/decayChain%halfLife(i)
63     End If
64     If(decayChain%halfLife(i).le.0.0D0.and.decayChain%decayConstant(i).gt.0.0D0)Then
65 ! complete decay constant from half life
66         decayChain%halfLife(i) = lnTwo/decayChain%decayConstant(i)
67     End If
68 ! Adjust for stable isotope
69     If(decayChain%decayConstant(i).lt.0.0D0)Then
70         decayChain%halfLife(i) = -1.0D0
71         decayChain%decayConstant(i) = 0.0D0
72     End If
73     If(decayChain%halfLife(i).lt.0.0D0)Then
74         decayChain%halfLife(i) = -1.0D0
75         decayChain%decayConstant(i) = 0.0D0
76     End If
77 ! Break out if stable
78     If(decayChain%decayConstant(i).eq.0.0D0)Then
79         Exit
80     End If
81 End Do
82 decayChain%isotopes = n
83 End Subroutine decayChainComplete
84
85 Subroutine DecayBreakInfinities(L,n)
86 !
87 Implicit None ! Force declaration of all variables

```

```

88 ! Vars In/Out
89   Real(kind=DoubleReal), Dimension(:) :: L ! Lambda
90   Integer(kind=StandardInteger) :: n
91 ! Vars Private
92   Integer(kind=StandardInteger) :: i,j
93   Logical :: breaking
94 ! Loop and alter matching decay constants slightly
95   breaking = .true.
96   Do While(breaking)
97     breaking = .false.
98     Do i=1,n-1
99       Do j=i+1,n
100         If(L(i).eq.L(j))Then
101           breaking = .true.
102           L(i) = L(i)*1.0000001D0 ! Vary by 0.00001%
103         End If
104       End Do
105     End Do
106   End Do
107 End Subroutine DecayBreakInfinities
108
109 Function CalcIsotopeChainCalc(t,m,W,L,N,B) Result (nEnd)
110   Implicit None ! Force declaration of all variables
111 ! Vars In
112   Real(kind=DoubleReal) :: t
113   Integer(kind=StandardInteger) :: m
114   Real(kind=DoubleReal), Dimension(1:100) :: W ! Production Rate
115   Real(kind=DoubleReal), Dimension(1:100) :: L ! Lambda
116   Real(kind=DoubleReal), Dimension(1:100) :: N ! Starting number of atoms
117   Real(kind=DoubleReal), Dimension(1:100) :: B ! Branch factor
118 ! Vars Out
119   Real(kind=DoubleReal) :: nEnd
120 ! Vars Private
121   Integer(kind=StandardInteger) :: i, j, k, z
122   Real(kind=DoubleReal) :: mult, multR
123   Real(kind=DoubleReal) :: nChange
124 ! Init
125   nEnd = 0.0D0
126 ! -----
127 ! UNSTABLE Isotopes
128 ! -----
129   If(L(m).gt.0.0D0)Then
130 ! Loop through terms
131   Do k=1,m
132     multR = CalcIsotopeChainMultR(k,m,L,B)
133     mult = multR * N(k)
134 ! decay of starting matter
135     nChange = CalcIsotopeChainF_Unstable(k,m,t,mult,L)
136     nEnd = nEnd + nChange
137 ! production term
138     mult = multR * W(k)
139     nChange = CalcIsotopeChainG_Unstable(k,m,t,mult,L)
140     nEnd = nEnd + nChange

```

```

141      print *,k,nEnd
142      End Do
143      Else
144 ! -----
145 ! STABLE Isotopes
146 ! -----
147 ! Loop through terms
148      nEnd = N(m)+t*W(m)
149      Do k=1,m-1
150          multR = CalcIsotopeChainMultR(k,m,L,B)
151          mult = multR * N(k)
152 ! decay of starting matter
153          nChange = CalcIsotopeChainF_Stable(k,m,t,mult,L)
154          nEnd = nEnd + nChange
155 ! production term
156          mult = multR * W(k)
157          nChange = CalcIsotopeChainG_Stable(k,m,t,mult,L)
158          nEnd = nEnd + nChange
159          print *,k,nEnd
160      End Do
161      End If
162 End Function CalcIsotopeChainCalc

163
164 Function CalcIsotopeChainMultR(k,m,L,B) Result (multR)
165 ! Vars In
166     Integer(kind=StandardInteger) :: k, m
167     Real(kind=DoubleReal), Dimension(1:100) :: L ! Lambda
168     Real(kind=DoubleReal), Dimension(1:100) :: B ! Branching Factor
169 ! Vars Out
170     Real(kind=DoubleReal) :: multR
171 ! Private
172     Integer(kind=StandardInteger) :: i
173 ! Result
174     multR = 1.0D0
175     If(k.lt.m)Then
176         Do i=k,m-1
177             multR = multR * B(i+1) * L(i)
178         End Do
179     End If
180 End Function CalcIsotopeChainMultR

181 !
182 ! -----
183 ! Unstable Isotope Functions
184 ! -----
185
186 Function CalcIsotopeChainF_Unstable(k,m,t,mult,L) Result (nChange)
187 ! Vars In
188     Integer(kind=StandardInteger) :: k, m
189     Real(kind=DoubleReal) :: t, mult
190     Real(kind=DoubleReal), Dimension(1:100) :: L ! Lambda
191 ! Vars Out
192     Real(kind=DoubleReal) :: nChange
193 ! Private

```

```

194  Real(kind=DoubleReal) :: multP
195  Real(kind=DoubleReal) :: p, q, r, s
196  Integer(kind=StandardInteger) :: i, j
197 ! Calculate isotope amount change
198  nChange = 0.0D0
199  multP = (-1.0D0)**(m-k)
200 ! Loop through pfrac
201  Do i=k,m
202    r = 1.0D0
203    Do j=k,m
204      If(j.ne.i)Then
205        r = r * (L(i)-L(j))
206      End If
207    End Do
208    nChange = nChange + (1.0D0/r)*exp(-1.0D0*L(i)*t)*multP*mult
209  End Do
210 End Function CalcIsotopeChainF_Unstable
211
212 Function CalcIsotopeChainG_Unstable(k,m,t,mult,L) Result (nChange)
213 ! Vars In
214  Integer(kind=StandardInteger) :: k, m
215  Real(kind=DoubleReal) :: t, mult
216  Real(kind=DoubleReal), Dimension(1:100) :: L ! Lambda
217 ! Vars Out
218  Real(kind=DoubleReal) :: nChange
219 ! Private
220  Real(kind=DoubleReal) :: multP
221  Real(kind=DoubleReal) :: p, q, r, s
222  Integer(kind=StandardInteger) :: i, j
223 ! Calculate isotope amount change
224  nChange = 0.0D0
225  multP = (-1.0D0)**(m-k+1)
226 ! term A
227  r = 1.0D0
228  Do i=k,m
229    r = r * L(i)
230  End Do
231  nChange = nChange + (1.0D0/r)*mult
232 ! Loop through pfrac
233  Do i=k,m
234    r = 1.0D0*L(i)
235    Do j=k,m
236      If(j.ne.i)Then
237        r = r * (L(i)-L(j))
238      End If
239    End Do
240    nChange = nChange + (1.0D0/r)*exp(-1.0D0*L(i)*t)*multP*mult
241  End Do
242 End Function CalcIsotopeChainG_Unstable
243
244 ! -----
245 ! Stable Isotope Functions
246 ! -----

```

```

247
248     Function CalcIsotopeChainF_Stable(k,mIn,t,mult,L) Result (nChange)
249 ! Vars In
250     Integer(kind=StandardInteger) :: k, mIn
251     Real(kind=DoubleReal) :: t, mult
252     Real(kind=DoubleReal), Dimension(1:100) :: L ! Lambda
253 ! Vars Out
254     Real(kind=DoubleReal) :: nChange
255 ! Private
256     Integer(kind=StandardInteger) :: m
257     Real(kind=DoubleReal) :: multP
258     Real(kind=DoubleReal) :: p, q, r, s
259     Integer(kind=StandardInteger) :: i, j
260 ! In
261     m = mIn-1
262 ! Calculate isotope amount change
263     nChange = 0.0D0
264     multP = (-1.0D0)**(m-k+1)
265 ! term A
266     r = 1.0D0
267     Do i=k,m
268         r = r * L(i)
269     End Do
270     nChange = nChange + (1.0D0/r)*mult
271 ! Loop through pfrac
272     Do i=k,m
273         r = 1.0D0*L(i)
274         Do j=k,m
275             If(j.ne.i)Then
276                 r = r * (L(i)-L(j))
277             End If
278         End Do
279         nChange = nChange + (1.0D0/r)*exp(-1.0D0*L(i)*t)*multP*mult
280     End Do
281 End Function CalcIsotopeChainF_Stable
282
283
284 Function CalcIsotopeChainG_Stable(k,mIn,t,mult,L) Result (nChange)
285 ! Vars In
286     Integer(kind=StandardInteger) :: k, mIn
287     Real(kind=DoubleReal) :: t, mult
288     Real(kind=DoubleReal), Dimension(1:100) :: L ! Lambda
289 ! Vars Out
290     Real(kind=DoubleReal) :: nChange
291 ! Private
292     Integer(kind=StandardInteger) :: m
293     Real(kind=DoubleReal) :: multP
294     Real(kind=DoubleReal) :: p, q, r, s
295     Integer(kind=StandardInteger) :: i, j
296 ! In
297     m = mIn-1
298 ! Calculate isotope amount change
299     nChange = 0.0D0

```

```

300      multP = (-1.0D0)**(m-k)
301 ! term A
302      r = 1.0D0
303      Do i=k,m
304          r = r * L(i)
305      End Do
306      nChange = nChange + (1.0D0/r)*t*mult
307 ! term B
308      p = CalcIsotopeChainC(L,k,m)
309      q = 1.0D0
310      Do i=k,m
311          q = q*L(i)*L(i)
312      End Do
313      r = (-1.0D0)*(p/q)
314      nChange = nChange + r*mult
315 ! Loop through pfrac
316      Do i=k,m
317          r = 1.0D0*L(i)*L(i)
318          Do j=k,m
319              If(j.ne.i)Then
320                  r = r * (L(i)-L(j))
321              End If
322          End Do
323          nChange = nChange + (1.0D0/r)*exp(-1.0D0*L(i)*t)*multP*mult
324      End Do
325 End Function CalcIsotopeChainG_Stable
326
327 Function CalcIsotopeChainC(L,k,m) Result (numerator)
328 ! Calculates numerator in isotope activity function
329 Implicit None ! Force declaration of all variables
330 ! Vars In
331     Real(kind=DoubleReal), Dimension(:) :: L
332     Integer(kind=StandardInteger) :: k, m
333 ! Vars Out
334     Real(kind=DoubleReal) :: numerator
335 ! Vars Private
336     Integer(kind=StandardInteger) :: i, j
337     Real(kind=DoubleReal) :: tempMult
338     numerator = 0.0D0
339     Do i=k,m
340         tempMult = 1.0D0
341         Do j=k,m
342             If(j.ne.i)Then
343                 tempMult = tempMult * L(j)
344             End If
345         End Do
346         numerator = numerator + tempMult
347     End Do
348 End Function CalcIsotopeChainC

```

Bibliography

- [1] A. J. Koning D. Rochman. “Modern Nuclear Data Evaluation With The TALYS Code System”. In: *Nuclear Data Sheets* 113 (2012).
- [2] J. P. Biersack James F. Ziegler M. D. Ziegler. “SRIM - The stopping and range of ions in matter”. In: *Nuclear Instruments and Methods in Physics Research Section B* 268 (2010), pp. 1818–1823.
- [3] A. A. Sonzogni. *Nuclear Data Services*. www-nds.iaea.org. 2006. URL: <http://www-nds.iaea.org/ndspub/download-endf/ENDF-B-VII.0/decay-index.htm> (visited on 08/14/2015).
- [4] M. Herman M. B. Chadwick. “ENDF/B-VII.0: Next generation evaluated nuclear data library for nuclear science and technology.” In: *Nucl. Data Sheets* 107 (2006), p. 2931.
- [5] P. Blaise M. Coste A. Courcelle T.D. Huynh C. Jouanne P. Leconte O. Litaize S. Mengelle G. Nogure J-M. Ruggiri O. Srot J. Tommasi C. Vaglio J-F. Vidal A. Santamarina D. Bernard. *The JEFF-3.1.1 Nuclear Data Library*. 2009. ISBN: 978-92-64-99074-6.
- [6] H. Stehfest. “Algorithm 368: Numerical Inversion of Laplace Transform”. In: *Communications of the ACM* 13 (1970), pp. 47–49.
- [7] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing In Science & Engineering* 9 (2007), pp. 90–95.

Appendix F

Activity V2

F.1 Sample Input File

Alex Dickinson-Lomas from the University of Birmingham requested a calculation to predict the radioactivity of a thin 0.1mm piece of steel in a 5MeV proton beam. A SRIM exyz file had been prepared and the following input file was set up.

Listing F.1: Activity V2 Input File

```
1 # Data files
2 data_isotopes="/cloud/Code/python/activity/data/isotopes" xs="/cloud/Code/python/activity/data/xs"
3
4 # Sim
5 sim1 exyz="EXYZ.txt" target_composition=Fe,96.375,C,0.772,Cu,0.024,Mn,1.36,Ni,0.698,Si,0.381,Cr,0.092,V,0.008,P
   ,0.009,Si,0.003,Mo,0.278 target_depth=0.1,mm target_density=7808,kgm3 beam_projectile='proton' beam_energy=5,
   MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,uA end_time=260000,s
```

F.2 Sample Output

The activity code processed the input file, and the calculation duration was approximately 15-20 minutes.

Listing F.2: Activity V2 Steel Terminal Output

```
1 Beam Details
2 =====
3 Projectile: Proton
4 Energy: None
5 Duration: 300.0
6 Area: 300.0
7 Current: 5e-07
8 Flux: 3120754564499.9995
9
10 Target
11
12 Depth: 0.0001 m
13 Density: 7808.0 kgm3
14 Mass: 55.107863928784624 amu
15
16 Composition:
17
18 Isotope Mass PBM PBN
19 =====
```

20	Fe54	5.3940E+01	5.6335E+00	5.7297E+00	4.3986E+02	4.9109E+27
21	Fe56	5.5935E+01	8.8428E+01	8.6731E+01	6.9045E+03	7.4336E+28
22	Fe57	5.6935E+01	2.0423E+00	1.9679E+00	1.5946E+02	1.6866E+27
23	Fe58	5.7933E+01	2.7216E-01	2.5773E-01	2.1250E+01	2.2090E+26
24	C12	1.2000E+01	7.6380E-01	3.4919E+00	5.9638E+01	2.9929E+27
25	C13	1.3003E+01	8.3222E-03	3.5111E-02	6.4979E-01	3.0094E+25
26	Cu63	6.2930E+01	1.6602E-02	1.4473E-02	1.2962E+00	1.2405E+25
27	Cu65	6.4928E+01	7.3999E-03	6.2526E-03	5.7779E-01	5.3590E+24
28	Mn55	5.4938E+01	1.3600E+00	1.3581E+00	1.0619E+02	1.1640E+27
29	Ni60	5.9931E+01	1.8304E-01	1.6755E-01	1.4292E+01	1.4361E+26
30	Ni61	6.0931E+01	7.9569E-03	7.1643E-03	6.2128E-01	6.1404E+24
31	Ni62	6.1928E+01	2.5369E-02	2.2474E-02	1.9808E+00	1.9262E+25
32	Ni64	6.3928E+01	6.4613E-03	5.5449E-03	5.0450E-01	4.7525E+24
33	Si28	2.7977E+01	2.7669E-03	5.4257E-03	2.1604E-01	4.6503E+24
34	Si29	2.8977E+01	1.4050E-04	2.6600E-04	1.0970E-02	2.2799E+23
35	Si30	2.9974E+01	9.2618E-05	1.6952E-04	7.2316E-03	1.4529E+23
36	Cr52	5.1941E+01	7.7086E-02	8.1421E-02	6.0189E+00	6.9785E+25
37	Cr53	5.2941E+01	8.7411E-03	9.0582E-03	6.8250E-01	7.7636E+24
38	Cr54	5.3939E+01	2.1764E-03	2.2137E-03	1.6994E-01	1.8973E+24
39	V51	5.0944E+01	7.9800E-03	8.5936E-03	6.2308E-01	7.3655E+24
40	P31	3.0974E+01	9.0000E-03	1.5941E-02	7.0272E-01	1.3663E+25
41	Mo94	9.3905E+01	2.5718E-02	1.5025E-02	2.0081E+00	1.2878E+25
42	Mo95	9.4906E+01	4.4261E-02	2.5586E-02	3.4559E+00	2.1929E+25
43	Mo96	9.5905E+01	4.6376E-02	2.6529E-02	3.6210E+00	2.2738E+25
44	Mo97	9.6906E+01	2.6571E-02	1.5043E-02	2.0747E+00	1.2893E+25

45

46 Starting Tally

47 =====

48	26054	Fe54	3.1429536676960092e+19
49	26056	Fe56	4.7574989041735316e+20
50	26057	Fe57	1.079453575560004e+19
51	26058	Fe58	1.4137464501727972e+18
52	6012	C12	1.9154473653970047e+19
53	6013	C13	1.9259847573081418e+17
54	29063	Cu63	7.938956724877902e+16
55	29065	Cu65	3.429786778008423e+16
56	25055	Mn55	7.449656284946808e+18
57	28060	Ni60	9.190961022927041e+17
58	28061	Ni61	3.929859828235443e+16
59	28062	Ni62	1.2327721842728802e+17
60	28064	Ni64	3.0415894209574764e+16
61	14028	Si28	2.9762044600901344e+16
62	14029	Si29	1459126206750858.0
63	14030	Si30	929869335659313.8
64	24052	Cr52	4.4662210592794496e+17
65	24053	Cr53	4.968735444487695e+16
66	24054	Cr54	1.214272751722711e+16
67	23051	V51	4.713925180516024e+16
68	15031	P31	8.744174716558614e+16
69	42094	Mo94	8.241841437276395e+16
70	42095	Mo95	1.4034621025229758e+17
71	42096	Mo96	1.4552033702553504e+17
72	42097	Mo97	8.251470736805168e+16
73	27053	Co53	0.0
74	25050	Mn50	0.0
75	26053	Fe53	0.0
76	26052	Fe52	0.0
77	25051	Mn51	0.0
78	25052	Mn52	0.0
79	24049	Cr49	0.0
80	27055	Co55	0.0

81	23047	V47	0.0
82	25053	Mn53	0.0
83	24050	Cr50	0.0
84	26055	Fe55	0.0
85	25054	Mn54	0.0
86	24051	Cr51	0.0
87	27057	Co57	0.0
88	23049	V49	0.0
89	27056	Co56	0.0
90	27058	Co58	0.0
91	23050	V50	0.0
92	25056	Mn56	0.0
93	27059	Co59	0.0
94	25057	Mn57	0.0
95	7014	N14	0.0
96	30062	Zn62	0.0
97	30061	Zn61	0.0
98	28059	Ni59	0.0
99	28058	Ni58	0.0
100	29062	Cu62	0.0
101	29061	Cu61	0.0
102	29060	Cu60	0.0
103	30064	Zn64	0.0
104	30063	Zn63	0.0
105	29064	Cu64	0.0
106	28063	Ni63	0.0
107	27060	Co60	0.0
108	30066	Zn66	0.0
109	27061	Co61	0.0
110	22048	Ti48	0.0
111	29059	Cu59	0.0
112	28057	Ni57	0.0
113	27062	Co62	0.0
114	26059	Fe59	0.0
115	27063	Co63	0.0
116	26060	Fe60	0.0
117	13024	Al24	0.0
118	14027	Si27	0.0
119	13026	Al26	0.0
120	15029	P29	0.0
121	14026	Si26	0.0
122	13025	Al25	0.0
123	11021	Na21	0.0
124	13027	Al27	0.0
125	12024	Mg24	0.0
126	12023	Mg23	0.0
127	15028	P28	0.0
128	15030	P30	0.0
129	11022	Na22	0.0
130	13028	Al28	0.0
131	12025	Mg25	0.0
132	11023	Na23	0.0
133	13029	Al29	0.0
134	12026	Mg26	0.0
135	23048	V48	0.0
136	22047	Ti47	0.0
137	21045	Sc45	0.0
138	21046	Sc46	0.0
139	23052	V52	0.0
140	22049	Ti49	0.0
141	21047	Sc47	0.0

```

142 23053      V53      0.0
143 22050      Ti50     0.0
144 22046      Ti46     0.0
145 20044      Ca44     0.0
146 16030      S30     0.0
147 16032      S32     0.0
148 10020      Ne20     0.0
149 43093      Tc93    0.0
150 43092      Tc92    0.0
151 41090      Nb90    0.0
152 41089      Nb89    0.0
153 42093      Mo93    0.0
154 42092      Mo92    0.0
155 42091      Mo91    0.0
156 41091      Nb91    0.0
157 41092      Nb92    0.0
158 40089      Zr89    0.0
159 43095      Tc95    0.0
160 39087      Y87     0.0
161 41093      Nb93    0.0
162 40090      Zr90    0.0
163 43094      Tc94    0.0
164 43096      Tc96    0.0
165 39088      Y88     0.0
166 41094      Nb94    0.0
167 40091      Zr91    0.0
168 43097      Tc97    0.0
169 39089      Y89     0.0
170 41095      Nb95    0.0
171 40092      Zr92    0.0
172 43098      Tc98    0.0
173 39090      Y90     0.0
174 41096      Nb96    0.0
175 40093      Zr93    0.0
176 1          Nn1     0.0
177 2004       He4     0.0
178 1001       H1      0.0
179 1002       H2      0.0
180 2003       He3     0.0
181 1003       H3      0.0
182 1025052    Mn52-M  0.0
183 1027060    Co60-M  0.0
184 1013026    Al126-M 0.0
185 10021      Ne21    0.0
186 10022      Ne22    0.0
187 1041093    Nb93-M  0.0
188 1040090    Zr90-M  0.0
189 1039089    Y89-M   0.0
190 1040089    Zr89-M  0.0
191 1041091    Nb91-M  0.0
192 1038087    Sr87-M  0.0
193 37087      Rb87    0.0
194 38087      Sr87    0.0
195 38088      Sr88    0.0
196 44098      Ru98    0.0
197
198
199 Run Sim
200 =====
201 Plot Ion Energy Lost
202

```

```

203 Ion Min/Max energies
204 =====
205 Max ion energy: 4950000.0
206 Min ion energy: 37528.5
207
208
209 Number Density
210 =====
211 Fe54      4.9108651057750145e+27
212 Fe56      7.4335920377711426e+28
213 Fe57      1.6866462118125062e+27
214 Fe58      2.2089788283949954e+26
215 C12       2.99288650843282e+27
216 C13       3.0093511832939716e+25
217 Cu63      1.2404619882621723e+25
218 Cu65      5.359041840638161e+24
219 Mn55      1.1640087945229387e+27
220 Ni60      1.43608765983235e+26
221 Ni61      6.140405981617879e+24
222 Ni62      1.926206537926375e+25
223 Ni64      4.752483470246057e+24
224 Si28      4.650319468890835e+24
225 Si29      2.2798846980482155e+23
226 Si30      1.4529208369676776e+23
227 Cr52      6.97847040512414e+25
228 Cr53      7.763649132012024e+24
229 Cr54      1.8973011745667357e+24
230 V51       7.365508094556287e+24
231 P31       1.3662772994622835e+25
232 Mo94      1.2877877245744368e+25
233 Mo95      2.1929095351921498e+25
234 Mo96      2.273755266023985e+25
235 Mo97      1.2892923026258074e+25
236
237 Calculate reaction rates
238
239 Reaction Rates
240 =====
241 26054 -6077.33918634
242 26056 -483804.853992
243 26057 -8178.39195382
244 26058 -1909.94156575
245 6013 -6.52012152744
246 29063 -1652.38584892
247 29065 -198.453009044
248 25055 -12210.915484
249 28060 1004.98674045
250 28061 -58.1671144121
251 28062 -9.28061752454
252 28064 -24.2493668631
253 14028 21113.5234259
254 14029 -0.0461122607693
255 14030 -0.645300628062
256 24052 7290.04169226
257 24053 -28.9023619139
258 24054 -10.2857142806
259 23051 -50.6697439819
260 15031 -21120.2628724
261 42094 -352.232677673
262 42095 -25.4917606732
263 42096 -155.434866026

```

```

264 42097 -3.89613150928
265 25051 1.54062703256e-08
266 27055 6077.33918633
267 25053 1493.67422813
268 25054 318.212949211
269 27057 487357.891417
270 23049 2.64460811847e-05
271 27058 7891.04027139
272 23050 0.00286551559081
273 27059 1854.07086162
274 7014 6.52012152744
275 29062 56.2053440309
276 29061 705.207008156
277 30064 124.84393152
278 30066 28.0983150696
279 27061 0.138917914494
280 22048 12.0822038765
281 15029 0.0853179405141
282 13027 0.383149852247
283 15030 0.0461122607692
284 16032 6.91627938818
285 41091 10.0503816208
286 41092 0.599431762138
287 43095 342.182296081
288 39087 6.58933417483e-16
289 41093 0.415508798647
290 40090 9.2262969886e-13
291 43096 24.8923288821
292 39088 8.97947641477e-14
293 41094 0.0754109801463
294 40091 9.70141425186e-15
295 43097 155.019357228
296 39089 5.86184699163e-15
297 40092 3.41253896966e-13
298 43098 3.82072052817
299 39090 3.32385561384e-15
300 40093 3.49317591935e-15
301 1 2.98443186628e-08
302 2004 32176.4777246
303 1001 61663841.0917
304 1002 7.0398304228e-12
305
306
307 Saturation Times
308 =====
309 Mn51      11980.384679
310 Co55      272748.238212
311 Mn53      5.01914153515e+14
312 Mn54      116576230.889
313 Co57      101493998.416
314 V49       123226813.841
315 Co58      26456423.5172
316 V50       1.90941486654e+25
317 Cu62      2528.32793551
318 Cu61      52371.3958826
319 Zn64      3.13696345947e+26
320 Co61      25672.2528836
321 P29       17.901426169
322 P30       647.770582862
323 Nb91      92745119373.8
324 Nb92      4.77365601936e+15

```

325	Tc95		311178.822832	
326	Y87		1241603.5031	
327	Tc96		1598214.43406	
328	Y88		39817188.8104	
329	Nb94		2.72592648801e+12	
330	Tc97		3.546133358e+14	
331	Tc98		5.86464032836e+14	
332	Y90		995772.233062	
333	Zr93		2.19582904139e+14	
334	Nn1		2656.25700712	
335				
336				
337	End of Beam			
338	=====			
339	26054	Fe54	3.1429536677e+19	0.0
340	26056	Fe56	4.75749890417e+20	0.0
341	26057	Fe57	1.07945357559e+19	0.0
342	26058	Fe58	1.41374645018e+18	0.0
343	6012	C12	1.9154473653970047e+19	0.0
344	6013	C13	1.92598475731e+17	0.0
345	29063	Cu63	7.93895672483e+16	0.0
346	29065	Cu65	3.429786778e+16	0.0
347	25055	Mn55	7.44965628495e+18	0.0
348	28060	Ni60	9.19096102293e+17	0.0
349	28061	Ni61	3.92985982828e+16	0.0
350	28062	Ni62	1.23277218427e+17	0.0
351	28064	Ni64	3.04158942096e+16	0.0
352	14028	Si28	2.97620446072e+16	0.0
353	14029	Si29	1.45912620675e+15	0.0
354	14030	Si30	9.29869335659e+14	0.0
355	24052	Cr52	4.4662210593e+17	0.0
356	24053	Cr53	4.96873544458e+16	0.0
357	24054	Cr54	1.21427275174e+16	0.0
358	23051	V51	4.71392518051e+16	0.0
359	15031	P31	8.74417471593e+16	0.0
360	42094	Mo94	8.24184143727e+16	0.0
361	42095	Mo95	1.40346210252e+17	0.0
362	42096	Mo96	1.45520337026e+17	0.0
363	42097	Mo97	8.25147073681e+16	0.0
364	25051	Mn51	4.45277850945e-06	1.1134310496e-09
365	24051	Cr51	1.69097660998e-07	4.89672869074e-14
366	27055	Co55	1820201.2799	19.9921940992
367	26055	Fe55	3000.47354256	2.39917424244e-05
368	25053	Mn53	448102.268439	2.67454985681e-09
369	25054	Mn54	95463.5167853	0.00245318566229
370	27057	Co57	146206720.102	4.31548856931
371	23049	V49	0.00793379542383	1.92876260142e-10
372	27058	Co58	2367271.87331	0.268052586418
373	27059	Co59	556221.258486	0.0
374	7014	N14	1956.03645823	0.0
375	29062	Cu62	14190.4346381	16.8137773681
376	29061	Cu61	209757.185203	11.9984651685
377	30066	Zn66	8429.49452087	0.0
378	27061	Co61	40.9543387597	0.00477902094999
379	22048	Ti48	3624.66116296	0.0
380	15029	P29	0.509829542009	0.0853179405141
381	13027	A127	114.944955674	0.0
382	15030	P30	7.48094498488	0.0345969837483
383	16032	S32	2074.88381646	0.0
384	41091	Nb91	3015.11448623	9.73903083617e-08
385	40091	Zr91	2.91042427556e-12	0.0

386	41092	Nb92	179.829528641	1.12852941331e-13
387	40092	Zr92	1.0237616909e-10	0.0
388	43095	Tc95	102506.592248	0.986835491746
389	39087	Y87	1.97608498356e-13	4.76788406747e-19
390	1038087	Sr87-M	7.10403001573e-17	4.85730190185e-21
391	38087	Sr87	3.9568373186e-13	0.0
392	41093	Nb93	124.652639594	0.0
393	40090	Zr90	2.78782773177e-10	0.0
394	43096	Tc96	7465.59941256	0.0139937023624
395	39088	Y88	2.6938125231e-11	2.0267480843e-18
396	38088	Sr88	5.38765544753e-11	0.0
397	41094	Nb94	22.6232940439	2.48624944215e-11
398	43097	Tc97	46505.8071684	3.92875657448e-10
399	39089	Y89	1.75855409749e-12	0.0
400	43098	Tc98	1146.21615845	5.8550167548e-12
401	44098	Ru98	2292.4932886	0.0
402	39090	Y90	9.96706834899e-13	2.99854397767e-18
403	40093	Zr93	1.0479527758e-12	1.42970417663e-26
404	1041093	Nb93-M	2.91835212081e-22	3.97651955467e-31
405	1	Nn1	7.59596051337e-06	8.56674034086e-09
406	1001	H1	18499152327.5	0.0
407	2004	He4	9652943.31739	0.0
408	1002	H2	2.11194912684e-09	0.0
409				
410		End of Sim		
411		=====		
412	26054	Fe54	3.1429536677e+19	0.0
413	26056	Fe56	4.75749890417e+20	0.0
414	26057	Fe57	1.07945357559e+19	0.0
415	26058	Fe58	1.41374645018e+18	0.0
416	6012	C12	1.9154473653970047e+19	0.0
417	6013	C13	1.92598475731e+17	0.0
418	29063	Cu63	7.93895672483e+16	0.0
419	29065	Cu65	3.429786778e+16	0.0
420	25055	Mn55	7.44965628495e+18	0.0
421	28060	Ni60	9.19096102293e+17	0.0
422	28061	Ni61	3.9298598283e+16	0.0
423	28062	Ni62	1.23277218427e+17	0.0
424	28064	Ni64	3.04158942096e+16	0.0
425	14028	Si28	2.97620446072e+16	0.0
426	14029	Si29	1.45912620675e+15	0.0
427	14030	Si30	9.29869335659e+14	0.0
428	24052	Cr52	4.4662210593e+17	0.0
429	24053	Cr53	4.96873544458e+16	0.0
430	24054	Cr54	1.21427275174e+16	0.0
431	23051	V51	4.71392518051e+16	0.0
432	15031	P31	8.74417471593e+16	0.0
433	42094	Mo94	8.24184143727e+16	0.0
434	42095	Mo95	1.40346210253e+17	0.0
435	42096	Mo96	1.45520337026e+17	0.0
436	42097	Mo97	8.25147073681e+16	0.0
437	25051	Mn51	2.79301596638e-34	6.98402288008e-38
438	24051	Cr51	4.29182988878e-06	1.24282774984e-12
439	27055	Co55	105034.147734	1.1536433315
440	26055	Fe55	1715632.35011	0.0137181711003
441	25053	Mn53	448102.267745	2.67454985267e-09
442	25054	Mn54	94828.5456205	0.00243686841137
443	27057	Co57	145090272.195	4.2825351033
444	23049	V49	0.00788386324816	1.91662373119e-10
445	27058	Co58	2298672.19924	0.260284860087
446	27059	Co59	556221.258486	0.0

```

447 7014      N14        1956.03645823    0.0
448 29062     Cu62       3.27537379744e-130   3.88088224441e-133
449 29061     Cu61       0.0741568656357   4.24189792818e-06
450 30066     Zn66       8429.49452087    0.0
451 27061     Co61       2.8255717303e-12   3.29720046856e-16
452 22048     Ti48       3624.66116296    0.0
453 13027     Al27       114.944955674   0.0
454 16032     S32        2074.88381646   0.0
455 41091     Nb91       3015.08919407   9.73894914074e-08
456 40091     Zr91       2.91042427556e-12  0.0
457 41092     Nb92       179.829528612   1.12852941313e-13
458 40092     Zr92       1.0237616909e-10  0.0
459 43095     Tc95       8413.0472009   0.0809927770448
460 39087     Y87        1.05602858852e-13  2.54797841382e-19
461 1038087   Sr87-M    3.86284994055e-15  2.64118089609e-19
462 37087     Rb87       2.64641489592e-16  1.20849009065e-34
463 38087     Sr87       3.9568373186e-13  0.0
464 41093     Nb93       124.652639594   0.0
465 40090     Zr90       2.79323174816e-10  0.0
466 43096     Tc96       4588.33716227   0.00860049153979
467 39088     Y88        2.64168875912e-11  1.9875316437e-18
468 38088     Sr88       5.38765544753e-11  0.0
469 41094     Nb94       22.6232875871   2.48624873256e-11
470 43097     Tc97       46505.8070664   3.92875656586e-10
471 39089     Y89        1.75855409749e-12  0.0
472 43098     Tc98       1146.21615693   5.85501674703e-12
473 44098     Ru98       2292.4932886   0.0
474 39090     Y90        4.56305196266e-13  1.37277196296e-18
475 40093     Zr93       1.04795277209e-12  1.42970417157e-26
476 1041093   Nb93-M    3.001699921e-21  4.09008849481e-30
477 1         Nn1        4.78609027493e-133  5.39776273996e-136
478 1001      H1         18499152327.5   0.0
479 2004      He4        9652943.31739   0.0
480 1002      H2         2.11194912684e-09  0.0
481
482 Gamma Dose - Beam End
483 =====
484
485 Activity/Bq          54.4306601689
486 Power eV/s           28342871818.2
487 Power J/s            4.54103823697e-09
488 Dose Gy/s            4.51575003676e-12
489 Dose Gy/hr           1.62567001323e-08
490 Percentage of annual dose/hr 0.0142506237521
491
492 Gamma Dose - Sim End
493 =====
494
495 Activity/Bq          5.80221584491
496 Energy eV/s           2227911995.02
497 Energy J/s            3.56951604018e-10
498 Dose Gy               8.52319971389e-13
499 Dose Gy/hr            1.27786970412e-09
500 Percentage of annual dose/hr 0.00112018061534
501
502
503 Absorbed Dose Calculations
504 =====
505 Absorbed dose assumptions:
506 1. radiation from point, emitted isotropically
507 2. 80Kg human

```

```
508 3. 1m from point source
509 4. 1m squared surface area
510 5. all energy absorbed
511
512
513 Dose Limits
514 =====
515 employees 18+      20 millisieverts/year
516 trainees 18+       6 millisieverts/year
517 public and under 18s 1 millisievert/year
518 public and under 18s 1.140771128E-04 millisieverts/hour
519
520 Dose averaged over area of skin not exceeding 1cm2
521
522 Source: http://www.hse.gov.uk/radiation/ionising/doses/
523
524
525
526 Run Time: 1007.034s
```

F.3 Output Plots

The program produces many different plots, including those of the cross sections for each target isotope. Four of the most important plots for the 5MeV proton irradiated steel are included here.

First is the total activity (fig. ??), from the time the beam starts until the beam is turned off. It shows a steady rise in activity, and from this graph it is hard to see any curve towards a saturation of radioactive isotopes as the beam duration is so short.

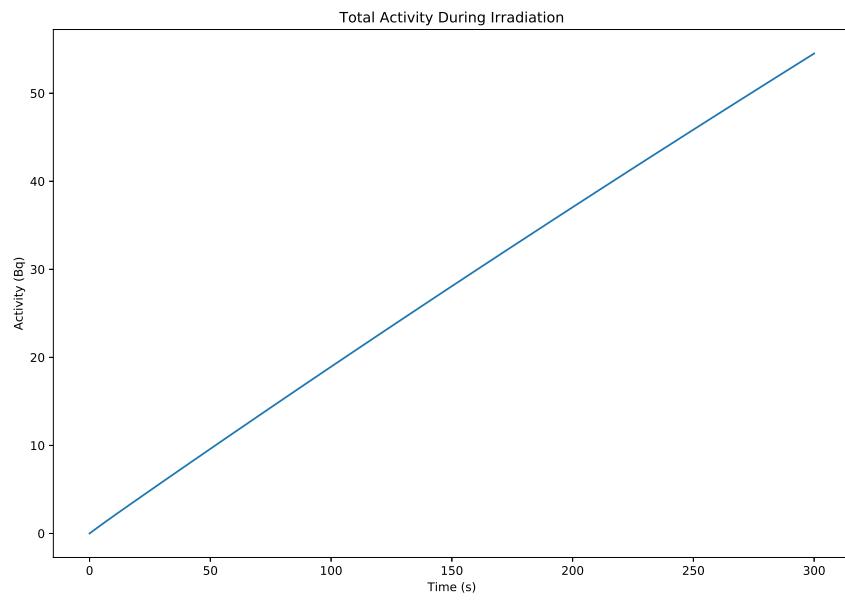


Figure F.1: Total activity from all isotopes from the start to the end of the beam (during irradiation)

The cooling period is much longer than the time spent within the beam, so a second graph shows the overall activity from when irradiation begins, to when it ceases up until the end of the simulation time (fig. ??).

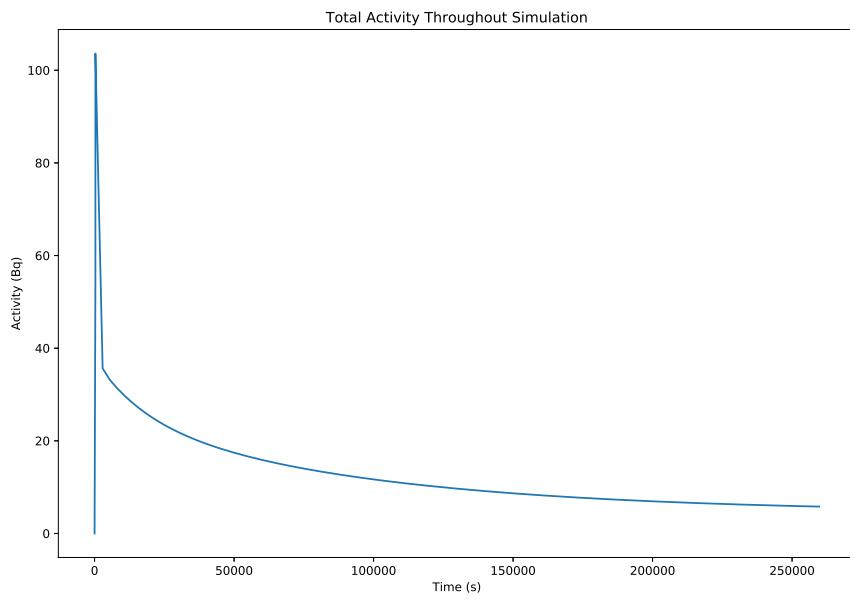


Figure F.2: Total activity from all isotopes from the start to the end of the simulation

The predicted gamma lines at the end of irradiation are predicted in the end of beam gamma line plot (fig. ??).

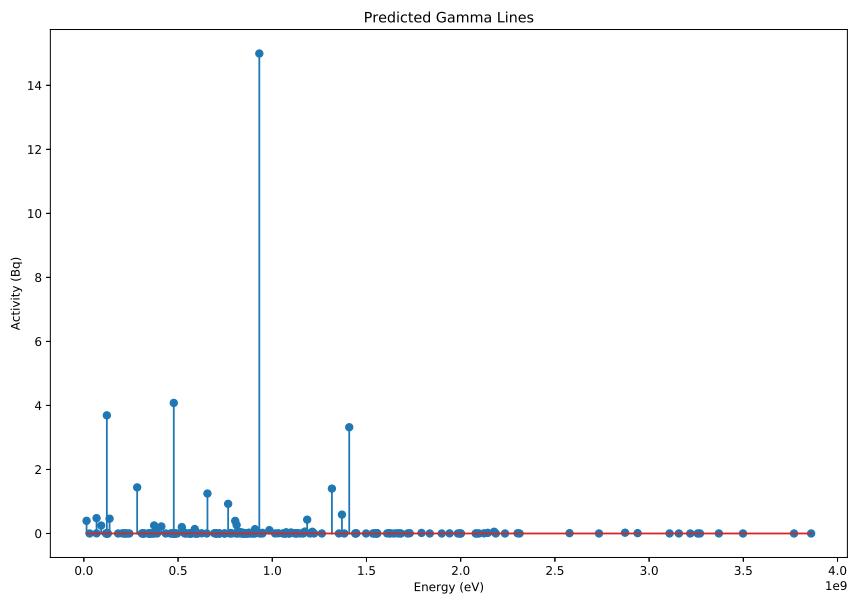


Figure F.3: Expected gamma lines and activity at the end of irradiation

The predicted gamma lines at the end of simulation are predicted in the end of sim gamma line plot (fig. ??).

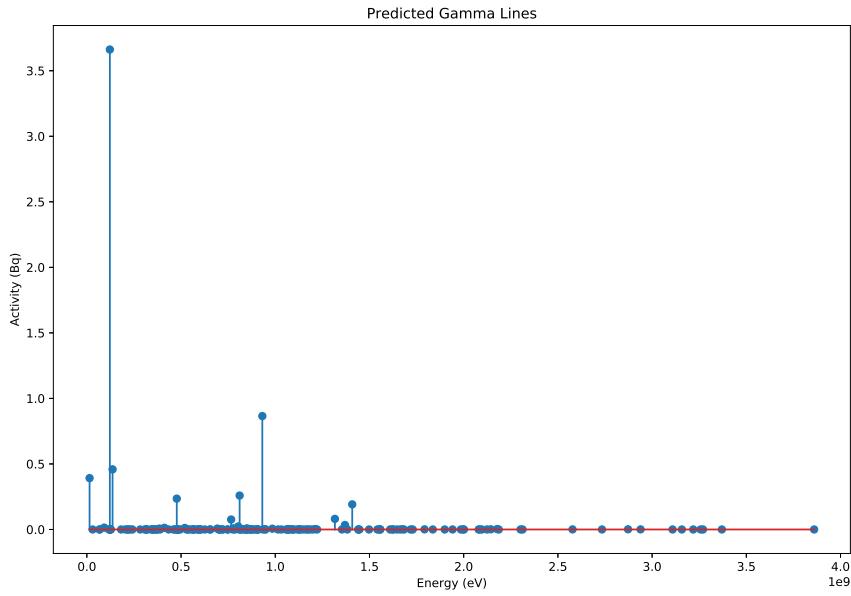


Figure F.4: Expected gamma lines and activity at the end of the simulation

F.4 An Example to Show Saturation In Beam

With the first steel example, the beam duration was too short to show a saturation of radioactive isotopes. This shows the gradual levelling off of activity during irradiation over a day as the source rate begins to equalize with the decay rate (fig. ??).

Listing F.3: Activity V2 input file: steel irradiated for 1 day

```

1 # Data files
2 data_isotopes="/cloud/Code/python/activity/data/isotopes" xs="/cloud/Code/python/activity/data/xs"
3
4 # Sim
5 sim1 exyz="EXYZ.txt" target_composition=Fe,96.375,C,0.772,Cu,0.024,Mn,1.36,Ni,0.698,Si,0.381,Cr,0.092,V,0.008,P
   ,0.009,Si,0.003,Mo,0.278 target_depth=0.1,mm target_density=7808,kgm3 beam_projectile='proton' beam_energy=5,
   MeV beam_area=64,mm2 beam_duration=86400,s beam_current=0.5,uA end_time=864000,s

```

F.5 Accompanying Manual

UNIVERSITY OF BIRMINGHAM



Department of Metallurgy & Materials

Activity V2 Manual

Contents

1 Overview	2
1.1 Reason for the Program	2
1.2 Program Requirements	2
2 Installation	3
3 How To Use	4
3.1 SRIM	4
3.2 Activity	5
3.3 Input Keywords	11
4 Examples	12
4.1 Iron - 28MeV Protons	12
4.2 Iron Fe56 (only) - 28MeV Protons	12
4.3 Steel - 5MeV Protons	12
5 Decay Equation	14
5.1 Bateman Equation	14
5.1.1 Laplace Transform	14
5.1.2 Constructing the Differential Equations	15
5.1.3 Numerical Inversion of the Laplace Transform	15
5.1.4 Analytic Solution by Partial Fraction Expansion	16
5.1.5 Preference: Analytic over Numeric	18
5.2 Python Isotopes Class	19
6 Future Plans	22
6.1 Selection of Cross Section Databases	22

Chapter 1

Overview

1.1 Reason for the Program

The Activity program calculates how radioactive a target becomes after being irradiated by high energy ions. It uses the TENDL-2019[1] database that contains cross-section data for protons and deuterons, and the JEFF-3.3 Radioactive Decay Data File[2].

1.2 Program Requirements

The user must provide a exyz file from the SRIM[3] ion transport program and a breakdown of the composition of the target, as well as other irradiation parameters (beam projectile, beam duration, beam area, target thickness, target composition, simulation end time etc).

Chapter 2

Installation

The program needs Python 3 installed in order to run. At the time of writing, it has only been developed to run on a Linux operating system, but it shouldn't require much adjusting to run on a Windows computer too.

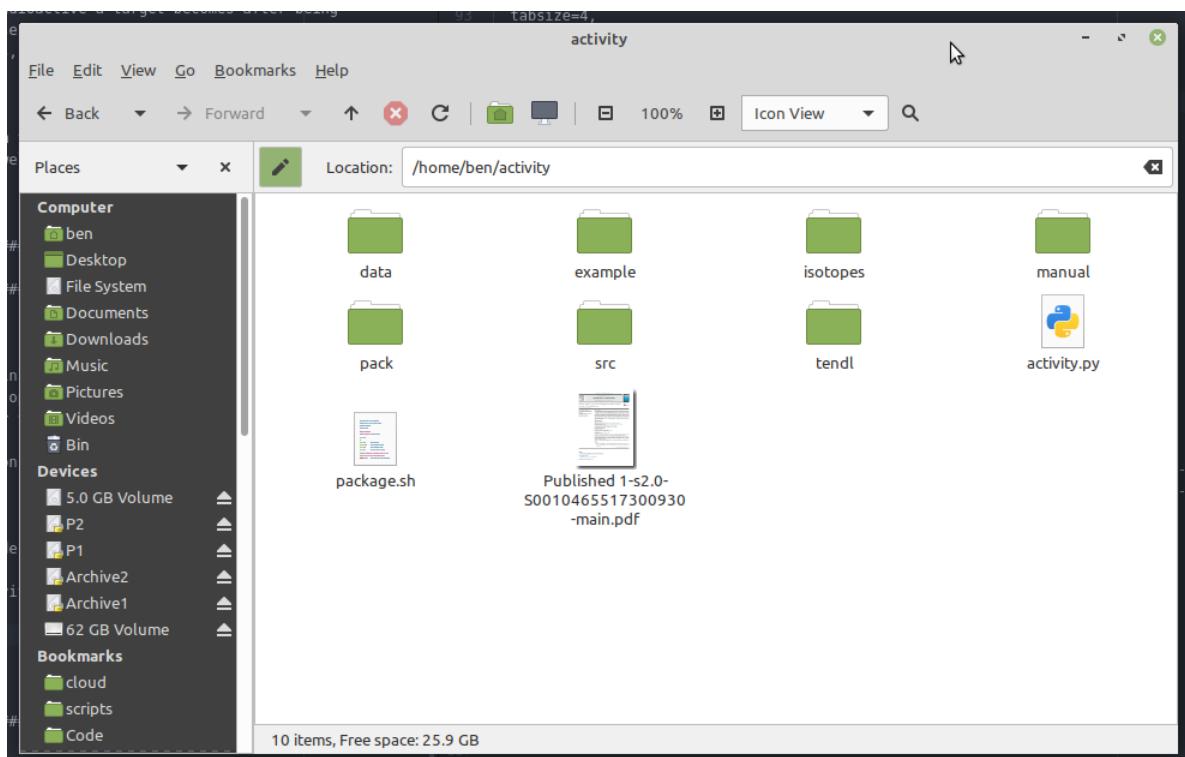
Download and install the latest version of Python 3:

<https://www.python.org/downloads>

The latest version of the activity code with data files must also be downloaded:

https://github.com/BenPalmer1983/activity_v2

For example, I have installed to /home/ben/activity

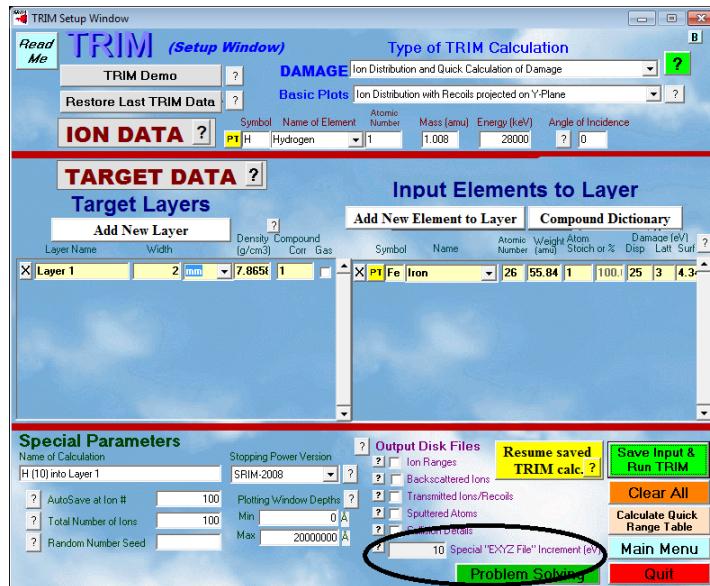


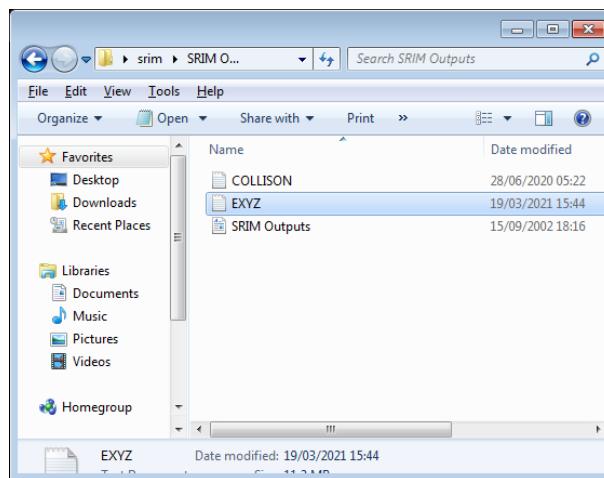
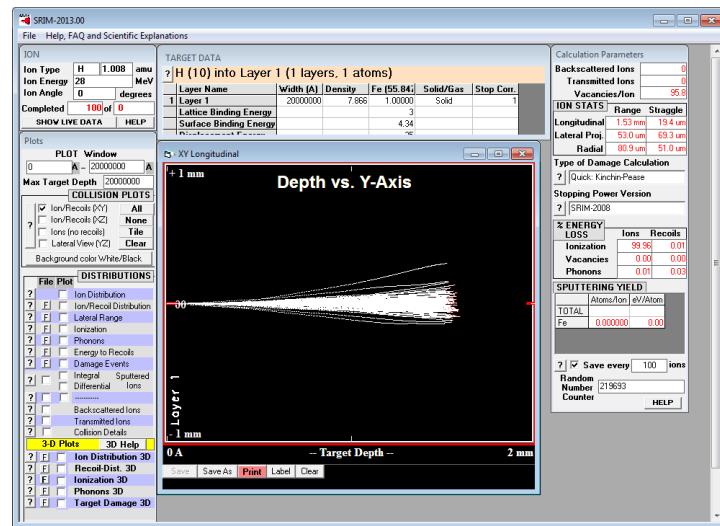
Chapter 3

How To Use

3.1 SRIM

If you haven't done so already, run the required simulation in SRIM. When setting up the calculation, be sure to set an increment for the EXYZ file. Sane values that have been used in examples range from 10eV to 10,000KeV, but this will depend on the resolution of the data in the TENDL database, the thickness of your target and the energy of the projectiles.

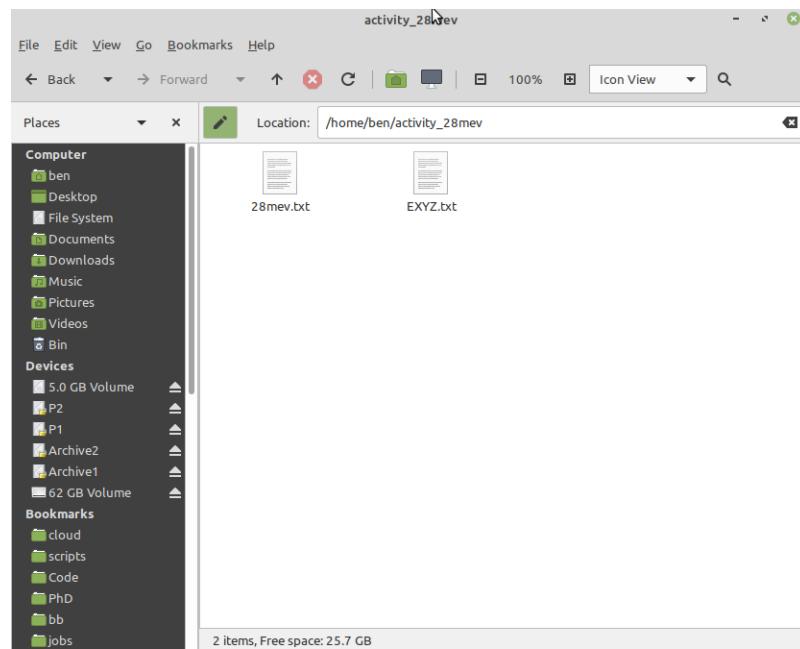




Take a copy of the EXYZ.txt file as it will be required by the activity program.

3.2 Activity

Create a directory to run the calculation. Copy in the EXYZ.txt file and create an input file.



The input file is just a text file, and will specify the calculation details. The data file paths must be specified, and then the simulations can be detailed, each with it's own unique name. The example below is for two simulations - the first with a 300s beam, and the second with a 3000s beam.

```

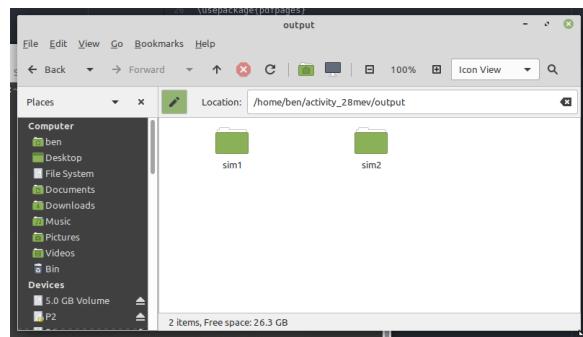
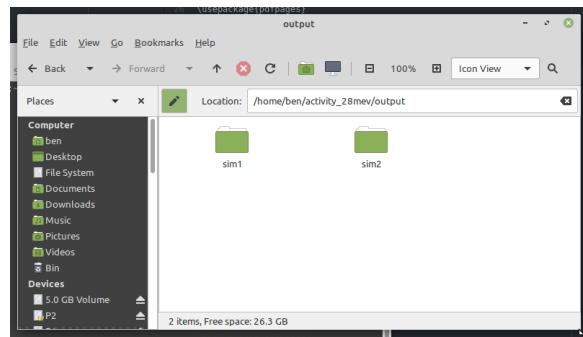
1 # Data files
2 data_isotopes="/home/ben/activity/data/isotopes" xs="/home/ben/activity/data/xs"
3
4 # Sim 1
5 sim1 exyz="EXYZ.txt" target_composition=Fe,100.0 target_depth=0.5,mm target_density=7808,kgm3
   beam_projectile='proton' beam_energy=28,MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,
   uA end_time=260000,s
6
7 # Sim 2
8 sim2 exyz="EXYZ.txt" target_composition=Fe,100.0 target_depth=0.5,mm target_density=7808,kgm3
   beam_projectile='proton' beam_energy=28,MeV beam_area=64,mm2 beam_duration=3000,s beam_current
   =5.0,uA end_time=260000,s

```

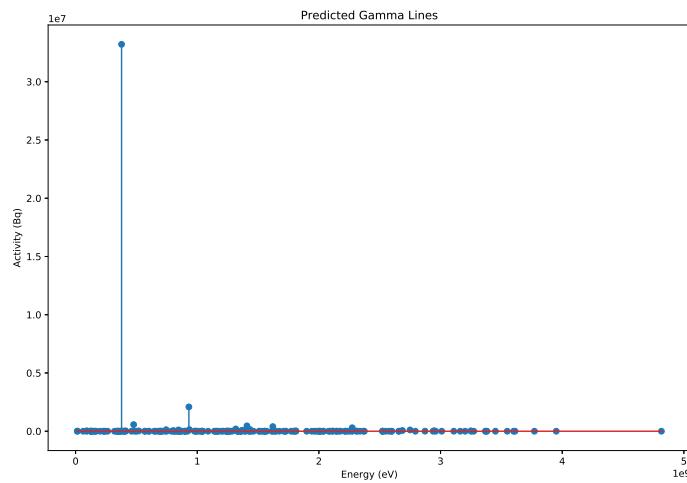
The program is run from the terminal/command line. Depending on your computer system, python3 might run through the command python3 or python, and on my computer the command is python3.

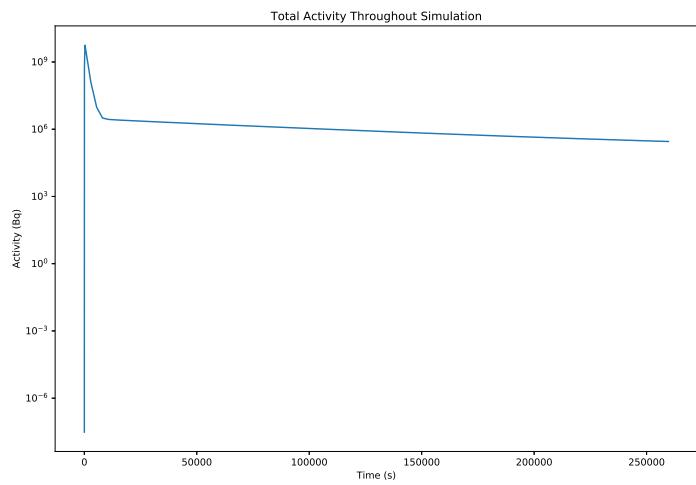
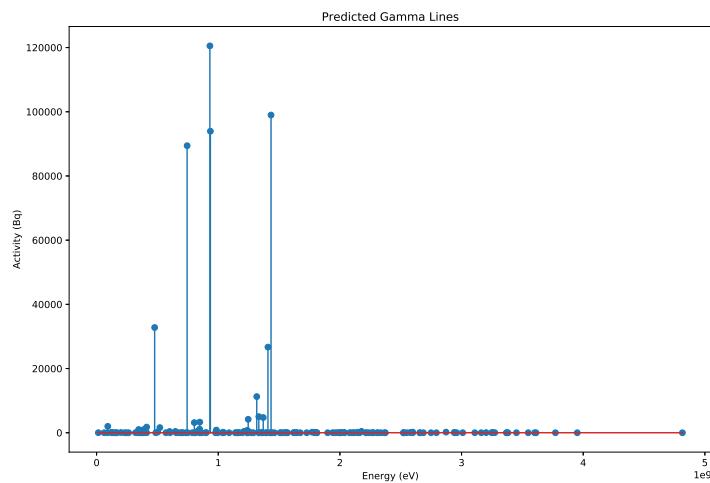
```
1 python3 /home/ben/activity/activity.py /home/ben/activity_28mev/28mev.txt
```

The program outputs into a directory for each simulation.



Within each directory are various plots and data files, including the activity over time and predicted gamma lines.





The program analyses the exyz file and output a chart showing the distribution of energy lost by projectiles as they pass through the target.

As a target is irradiated, the residual radioactive isotopes begin to decay. To begin with, the production outweighs the decay, but at a certain point there's enough of the radioactive isotope within the target that the decay balances with the production. At this point the radioactive isotope in the target is saturated, and will not increase above this amount unless the beam parameters are adjusted.

The saturation times depend on the isotope and the various reaction cross sections with the projectile. The saturation plots for each radioactive isotope are created and saved by the program.

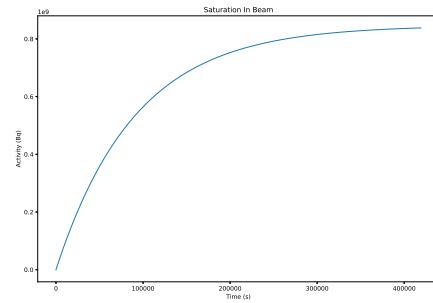
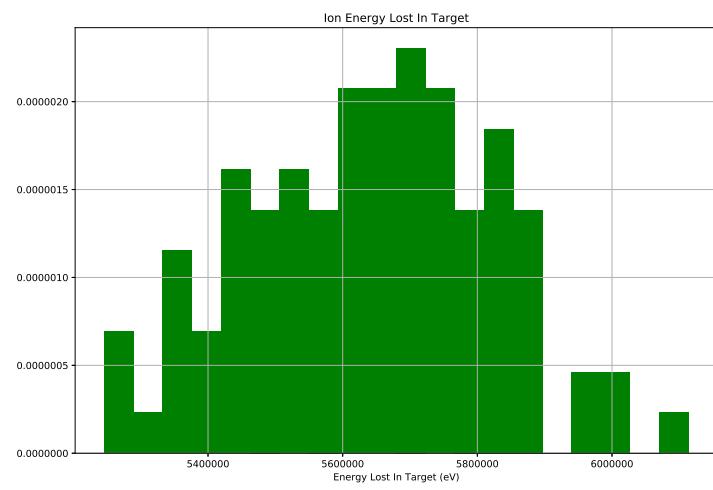


Figure 3.1: Saturation of Cobalt-55

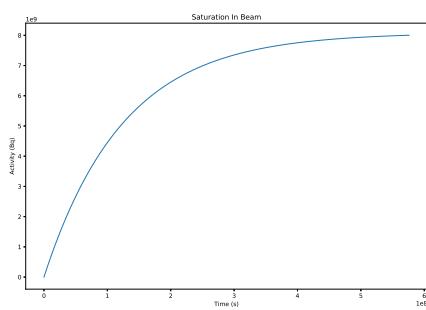


Figure 3.2: Saturation of Iron-55

The program calculates the dose a human may receive from the target at the time it is removed from the beam, and at the time the simulation stops.

Listing 3.1: Predicted Dose

```
1 Gamma Dose - Beam End
2 =====
3
4 Activity/Bq          54.4306601689
5 Power eV/s           28342871818.2
6 Power J/s            4.54103823697e-09
7 Dose Gy/s            4.51575003676e-12
8 Dose Gy/hr           1.62567001323e-08
9 Percentage of annual dose/hr 0.0142506237521
10
11 Gamma Dose - Sim End
12 =====
13
14 Activity/Bq          5.80221584491
15 Energy eV/s           2227911995.02
16 Energy J/s            3.56951604018e-10
17 Dose Gy               8.52319971389e-13
18 Dose Gy/hr            1.27786970412e-09
19 Percentage of annual dose/hr 0.00112018061534
20
21
22 Absorbed Dose Calculations
23 =====
24 Absorbed dose assumptions:
25 1. radiation from point, emitted isotropically
26 2. 80Kg human
27 3. 1m from point source
28 4. 1m squared surface area
29 5. all energy absorbed
30
31
32 Dose Limits
33 =====
34 employees 18+          20 millisieverts/year
35 trainees 18+           6 millisieverts/year
36 public and under 18s    1 millisievert/year
37 public and under 18s    1.140771128E-04 millisieverts/hour
38
39 Dose averaged over area of skin not exceeding 1cm2
40 Source: http://www.hse.gov.uk/radiation/ionising/doses/
```

3.3 Input Keywords

Command	<i>Sub Command</i>	Description
data		data fields are input here
data	isotopes	path to isotopes directory
data	xs	path to cross section directory
sim1		each simulation is numbered sequentially from 1 i.e. sim1 sim2 sim3...
sim1	exyz	path to exyz SRIM file
sim1	target_composition	isotopes that make up the beam target
sim1	target_depth	the depth of material through which the beam passes
sim1	target_density	how dense (specify kgm3 or GCM3)
sim1	beam_projectile	currently, only "proton", but this could be extended in the future
sim1	beam_energy	energy of the protons/projectiles
sim1	beam_area	how much area the beam covers
sim1	beam_duration	how long the beam will be on for (specify s,minute,hour) t=0 to beam_duration is the first part of the calculation
sim1	end_time	the time the calculation and final activity is measured after t=0 (specify s,minute,hour) end_time to beam_duration is the second part of the calculation

Chapter 4

Examples

4.1 Iron - 28MeV Protons

In this example, natural iron makes up the target .

```
1 # Data files
2 data_isotopes="/home/ben/activity/data/isotopes" xs="/home/ben/activity/data/xs"
3
4 # Sim
5 sim1 exyz="XYZ.txt" target_composition=Fe,100.0 target_depth=0.5,mm target_density=7808,kgm3
  beam_projectile='proton' beam_energy=28,MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,
  uA end_time=260000,s
```

The simulation section defines what the target material is made of as well as the beam parameters.

4.2 Iron Fe56 (only) - 28MeV Protons

In this example, the target is made of Fe56 only, and contains none of the other naturally occurring stable isotopes.

```
1 # Data files
2 data_isotopes="/home/ben/activity/data/isotopes" xs="/home/ben/activity/data/xs"
3
4 # Sim
5 sim1 exyz="XYZ.txt" target_composition=Fe56,100.0 target_depth=0.5,mm target_density=7808,kgm3
  beam_projectile='proton' beam_energy=28,MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,
  uA end_time=260000,s
```

4.3 Steel - 5MeV Protons

This example was provided by Alex Dickinson-Lomas, with a steel containing a wider range of elements.

```
1 # Data files
2 data_isotopes="/home/ben/activity/data/isotopes" xs="/home/ben/activity/data/xs"
3
```

```
4 # Sim
5 sim1 exyz="EXYZ.txt" target_composition=Fe,96.375,C,0.772,Cu,0.024,Mn,1.36,Ni,0.698,Si,0.381,Cr,0.092,
V,0.008,P,0.009,Si,0.003,Mo,0.278 target_depth=0.1,mm target_density=7808,kgm3 beam_projectile='
proton' beam_energy=5,MeV beam_area=64,mm2 beam_duration=300,s beam_current=0.5,uA end_time
=260000,s
```

Chapter 5

Decay Equation

5.1 Bateman Equation

The Bateman equation was derived using Laplace transforms, and this same method has been used to develop a modified equation that incorporates branching factors and production rates for each isotope in the decay chain, as illustrated by Figure 5.1.

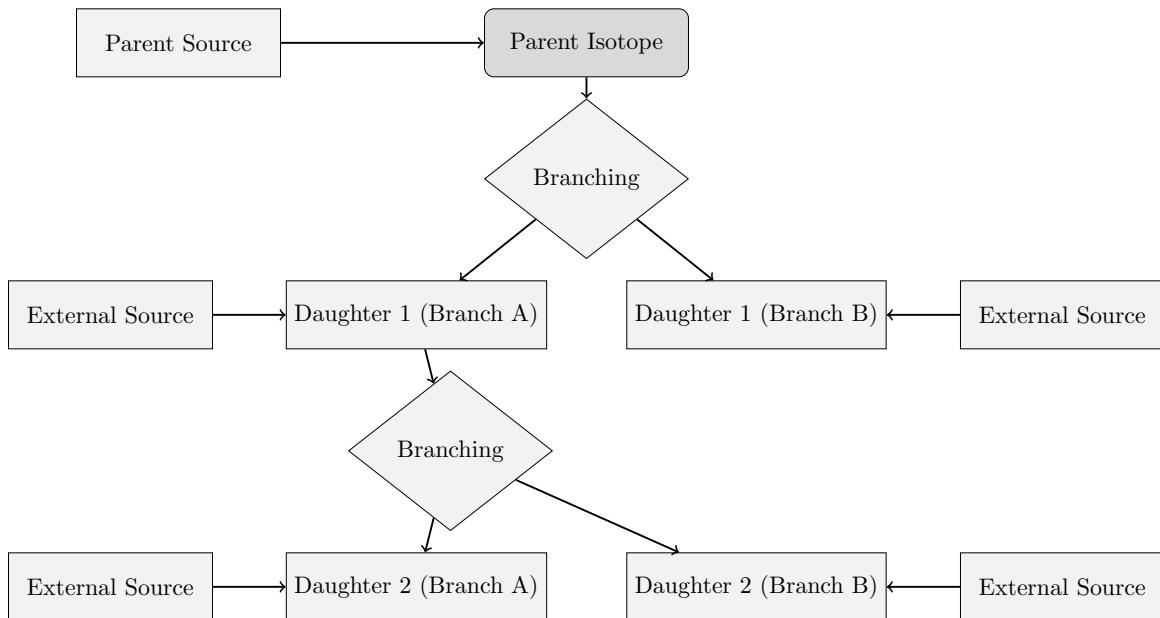


Figure 5.1: An example of several decay chains including branching factors and possible external source terms for each isotope on each chain.

5.1.1 Laplace Transform

Laplace Transforms (5.1) are a useful mathematical tool, and allow ordinary differential equations to be solved by simple algebraic manipulation in the s domain. Bateman took advantage of Laplace Transforms in deriving his equation, and this is the method that has been taken here as well.

$$F(s) = \int_0^{\infty} f(t) \exp(-st) dt \quad (5.1)$$

5.1.2 Constructing the Differential Equations

The first step is to set up differential equations for the parent isotope, unstable daughter isotopes and stable daughter isotope. The parent isotope has a source term, due to production, and a loss term, due to decay. The unstable daughter isotopes have two source terms, from the production by irradiation induced transmutation and the decay of preceding isotopes in the decay chain, and a loss term, due to decay. Finally, the stable daughter that finalizes the decay chain has two source terms (the same as the unstable daughters) but no loss term.

The variables (and vectors) used in these equations are defined as follows:

- $\vec{\lambda}$ vector containing isotope decay constants λ_i
- \vec{b} vector containing isotope to isotope branching factors b_i
- \vec{w} vector containing isotope production rates w_i
- t time at which activity/amount of isotope is measured
- $N_i(0)$ starting amount of the i^{th} isotope
- $N_i(t)$ amount of the i^{th} isotope at time t
- $N'_i(t)$ change in amount of the i^{th} isotope, with respect to time, at time t

The differential equations for the parent isotope (first isotope), unstable daughter isotopes (i^{th} isotopes) and stable, final, daughter isotope (zth isotope) in the time domain are as follows:

$$N'_1(t) = \omega_1 - \lambda_1 N_1(t) \quad (5.2)$$

$$N'_i(t) = \omega_i + b_{i-1} \lambda_{i-1} N_{i-1}(t) - \lambda_i N_i(t) \quad (5.3)$$

$$N'_z(t) = \omega_z + b_{z-1} \lambda_{z-1} N_{z-1}(t) \quad (5.4)$$

Applying the Laplace Transform to these three differential equations allows them to be manipulated and solved algebraically in the s-domain.

$$N_1(s) = \frac{1}{s + \lambda_1} N_1(0) + \frac{1}{s(s + \lambda_1)} \omega_1 \quad (5.5)$$

$$N_i(s) = \frac{1}{s(s + \lambda_i)} (\omega_i) + \frac{1}{s + \lambda_i} (b_{i-1} \lambda_{i-1} N_{i-1}(s)) + \frac{1}{s + \lambda_i} N_i(0) \quad (5.6)$$

$$N_z(s) = \frac{1}{s^2} \omega_z + \frac{1}{s} b_{z-1} \lambda_{z-1} N_{z-1}(s) + \frac{1}{s} N_z(0) \quad (5.7)$$

5.1.3 Numerical Inversion of the Laplace Transform

The Gaver-Stehfest[4] algorithm was developed in the 1960s and 1970s and is a method of calculating the inverse of a Laplace Transform in the real number domain. It is an easy to implement and reasonably accurate method,

although it is an approximation to the real value. A comparison between an analytic and numeric inversion for the unstable isotope Po-218 is discussed at the end of this section (figure 5.2).

$$f(t) \approx f_n(t) = \frac{\ln(2)}{t} \sum_{k=1}^{2n} a_k(n) F(s) \text{ where } n \geq 1, t > 0 \quad (5.8)$$

$$s = \frac{k \ln(2)}{t} \quad (5.9)$$

$$a_k(n) = \frac{(-1)^{(n+k)}}{n!} \sum_{j=\text{Floor}(\frac{k+1}{2})} j^{n+1} \binom{n}{j} \binom{2j}{j} \binom{j}{k-j} \quad (5.10)$$

The equation for the i^{th} isotope may be calculated by recursively calculating the equations by numeric inversion, starting from the first (parent isotope) and inserting the result into each subsequent recursion until the i^{th} isotope is reached (changing the equations appropriately for the parent, unstable daughter and stable daughter isotopes).

5.1.4 Analytic Solution by Partial Fraction Expansion

The equation for the i^{th} isotope in the s domain can be written in full by substituting the preceding equation until the parent isotope is reached, and this full equation may be rearranged with the production amount of each isotope and starting amount of each isotope in individual terms. Each of these terms is multiplied by a fraction that can be expanded, using partial fractions, and inverted analytically.

This is illustrated with an example unstable isotope, fourth in the decay chain (including the parent isotope):

$$\begin{aligned} N_4(s) = & \frac{1}{(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\ & + \frac{1}{(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\ & + \frac{1}{(s + \lambda_3)(s + \lambda_4)} b_4 \lambda_3 N_3(0) \\ & + \frac{1}{(s + \lambda_4)} N_4(0) \quad (5.11) \\ & + \frac{1}{s(s + \lambda_1)(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\ & + \frac{1}{s(s + \lambda_2)(s + \lambda_3)(s + \lambda_4)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\ & + \frac{1}{s(s + \lambda_3)(s + \lambda_4)} b_4 \lambda_3 \omega_3 \\ & + \frac{1}{s(s + \lambda_4)} \omega_4 \end{aligned}$$

An example stable isotope, fourth (last) in the decay chain (including the parent isotope):

$$\begin{aligned}
N_4(s) = & \frac{1}{s(s+\lambda_1)(s+\lambda_2)(s+\lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 N_1(0) \\
& + \frac{1}{s(s+\lambda_2)(s+\lambda_3)} b_3 b_4 \lambda_2 \lambda_3 N_2(0) \\
& + \frac{1}{s(s+\lambda_3)} b_4 \lambda_3 N_3(0) \\
& + N_4(0) \\
& + \frac{1}{s^2(s+\lambda_1)(s+\lambda_2)(s+\lambda_3)} b_2 b_3 b_4 \lambda_1 \lambda_2 \lambda_3 \omega_1 \\
& + \frac{1}{s^2(s+\lambda_2)(s+\lambda_3)} b_3 b_4 \lambda_2 \lambda_3 \omega_2 \\
& + \frac{1}{s^2(s+\lambda_3)} b_4 \lambda_3 \omega_3 \\
& + \frac{1}{s^2} \omega_4
\end{aligned} \tag{5.12}$$

By using partial fraction expansion and standard Laplace Transforms, the set of equations below is used to calculate the amount of the m^{th} isotope in the decay chain, providing the m^{th} isotope is unstable.

$$N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = \sum_{k=1, m} r(k; \vec{\lambda}, \vec{b}) [f(t; k, m, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k] \tag{5.13}$$

$$r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k, m-1} (b_{i+1} \lambda_i), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \tag{5.14}$$

$$f(t; k, m, \vec{\lambda}) = (-1)^{m-k} \sum_{i=k, m} \left[\exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \tag{5.15}$$

$$g(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k, m} \lambda_i} + (-1)^{m-k+1} \sum_{i=k, m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k, m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \tag{5.16}$$

The set of equations below is used to calculate the amount of the m^{th} isotope in the decay chain, where the m^{th} isotope is stable.

$$N_m(t; \vec{\lambda}, \vec{b}, \vec{w}) = N_m + w_m t + \sum_{k=1, m-1} r(k; \vec{\lambda}, \vec{b}) [f(t; k, m-1, \vec{\lambda}) N_k(0) + g(t; k, m, \vec{\lambda}) w_k] \tag{5.17}$$

$$r(k, m, \vec{\lambda}) = \begin{cases} \prod_{i=k, m-1} (b_{i+1} \lambda_i), & \text{if } k < m \\ 1, & \text{if } k = m \end{cases} \tag{5.18}$$

$$f(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k,m} \lambda_i} + (-1)^{m-k+1} \sum_{i=k,m} \left[\frac{1}{\lambda_i} \exp(-\lambda_i t) \prod_{j=k,m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (5.19)$$

$$g(t; k, m, \vec{\lambda}) = \frac{1}{\prod_{i=k,m} \lambda_i} t + \frac{\sum_{i=k,m} \left[\prod_{j=k,m; j \neq i} \lambda_j \right]}{\prod_{i=k,m} \lambda_i^2} + (-1)^{m-k+1} \sum_{i=k,m} \left[\frac{1}{\lambda_i^2} \exp(-\lambda_i t) \prod_{j=k,m; j \neq i} \left(\frac{1}{\lambda_i - \lambda_j} \right) \right] \quad (5.20)$$

5.1.5 Preference: Analytic over Numeric

The numeric solution only requires the equation to be solved in the s-domain; the Gaver-Stehfest algorithm performs the inversion. It is worth the extra effort to derive and implement an analytic solution, as the numeric is only an approximation. Examples of the pitfalls of the numeric solution are that it can give negative amounts of an isotope and the difference between the numeric and analytic calculated amounts can become quite large when the isotope decays away to a very small value. Figure 5.2 shows the predicted decay of a sample of Po-218 irradiated for 1,000s, and sampled until 10,000s. In the region between 4,000s and 9,000s the amount from the numeric calculation drops below zero, whereas the analytic calculation remains above zero, as would be expected.

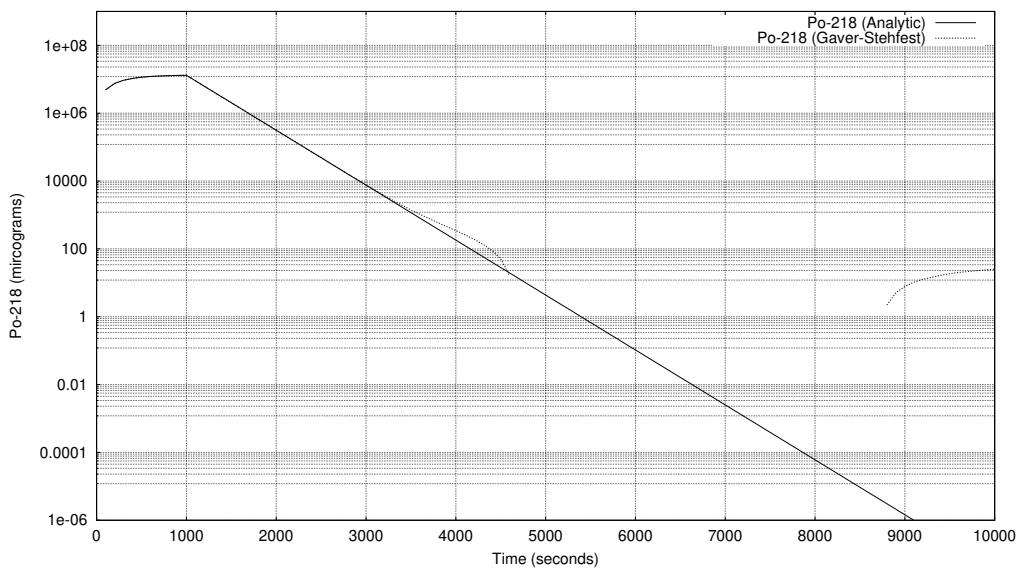


Figure 5.2: Decay of Po-218: Analytic and Gaver-Stehfest Calculations [jeff311]

5.2 Python Isotopes Class

The decay equations are computed within the isotopes class in the isotopes.py file.

```

1
2     class isotopes:
3
4     ######
5     # DECAY EQUATIONS
6     #####
7
8     @staticmethod
9     def calculate_activity(t, l, b, w, n0):
10    nt = numpy.zeros((len(n0),))
11    for m in range(0,len(n0)):
12        if(l[m] > 0.0):
13            nt[m] = isotopes.activity_unstable(t, l, b, w, n0, m)
14        elif(l[m] == 0.0):
15            nt[m] = isotopes.activity_stable(t, l, b, w, n0, m)
16    return nt
17
18     @staticmethod
19     def activity_unstable(t, l, b, w, n0, m):
20        s = 0.0
21        for k in range(0, m+1):
22            s = s + isotopes.r(k, m, b, l) * ( isotopes.f_unstable(t,k,m,l) * n0[k] + isotopes.g_unstable(t,
23                                         k,m,l) * w[k])
24    return s
25
26     @staticmethod
27     def activity_stable(t, l, b, w, n0, m):
28        s = n0[m] + w[m] * t

```

```

28     for k in range(0, m):
29         s = s + isotopes.r(k, m, b, l) * (isotopes.f_stable(t,k,m,l) * n0[k] + isotopes.g_stable(t,k,m,l)
30             ) * w[k])
31     return s
32
33 @staticmethod
34 def r(k, m, b, l):
35     if(k == m):
36         return 1.0
37     else:
38         p = 1.0
39         for i in range(k, m):
40             p = p * (b[i] * l[i])
41         return p
42
43 @staticmethod
44 def f_unstable(t,k,m,l):
45     s = 0.0
46     for i in range(k, m+1):
47         p = 1.0
48         for j in range(k, m+1):
49             if(i != j):
50                 p = p * (1 / (l[i] - l[j]))
51             s = s + numpy.exp(-1 * l[i] * t) * p
52     s = (-1)**(m-k) * s
53     return s
54
55 @staticmethod
56 def g_unstable(t,k,m,l):
57     pa = 1.0
58     for i in range(k,m+1):
59         pa = pa * l[i]
60     pa = 1.0 / pa
61     s = 0.0
62     for i in range(k, m+1):
63         pb = 1.0
64         for j in range(k, m+1):
65             if(i != j):
66                 pb = pb * (1 / (l[i]-l[j]))
67             s = s + (1/l[i]) * numpy.exp(-l[i]*t) * pb
68     return pa + s * (-1)**(m-k+1)
69
70 @staticmethod
71 def f_stable(t,k,m_in,l):
72     m = m_in - 1
73
74     p = 1.0
75     for i in range(k, m+1):
76         p = p * l[i]
77
78     s = 0.0
79     for i in range(k, m+1):

```

```

80     r = l[i]
81     for j in range(k, m+1):
82         if(i != j):
83             r = r * (l[i] - l[j])
84         s = s + (1/r)*numpy.exp(-l[i]*t)
85
86     return (1.0/p) + s * (-1.0)**(m-k+1)
87
88
89 @staticmethod
90 def g_stable(t,k,m_in,l):
91     m = m_in - 1
92
93     pa = 1.0
94     for i in range(k,m+1):
95         pa = pa * l[i]
96     pa = 1.0 / pa
97
98     sa = 0.0
99     for i in range(k, m+1):
100        pb = 1.0
101        for j in range(k,m+1):
102            if(j != i):
103                pb = pb * l[j]
104            sa = sa + pb
105        pc = 1.0
106        for i in range(k, m+1):
107            pc = pc * l[i]**2
108
109        sb = 0.0
110        for i in range(k, m+1):
111            pd = 1.0
112            for j in range(k, m+1):
113                if(i != j):
114                    pd = pd * (1 / (l[i]-l[j]))
115                sb = sb + (1/(l[i]**2)) * numpy.exp(-l[i]*t) * pd
116
117    return pa * t + sa / pc + sb * (-1)**(m-k+1)

```

Chapter 6

Future Plans

6.1 Selection of Cross Section Databases

The Scanditronix MC40 Cyclotron at the University of Birmingham has several beamlines and is capable of accelerating protons, deuterons, Helium 3 and Helium 4 and fluxes and energy ranges detailed below.

Particle	Energy (MeV)	Max Current (micro A)	Flux (ions per second)
p	8-40	60	3.75×10^{14}
d	8-40	30	1.87×10^{14}
$^4He^{2+}$	8-53	30	9.36×10^{13}
$^3He^{2+}$	4-20	60	1.87×10^{14}

Table 6.1: Beam Characteristics of the Scanditronix MC-40

The current data file is for Protons, and the range of energies (for Fe-54 and Fe-56 at the very least) only range up to 30MeV. Previous versions of the TENDL data files have covered larger ranges (TENDL-2009 extended up to 200MeV).

As the energies do not cover the full range of possible energies for our own cyclotron, it would be desirable to use the TALYS program to calculate the reaction cross sections for a range up to at least 100MeV (for Protons, Deuterons, Helium 3 and Helium 4 ions) and save this cross section data into an alternate database.

Bibliography

- [1] A.J. Koning et al. “TENDL: Complete Nuclear Data Library for Innovative Nuclear Science and Technology”. In: *Nuclear Data Sheets* 155 (2019).
- [2] A.J.M. Plompen et al. “The JEFF-3.3 Nuclear Data Library”. In: *Eur. Phys. J. A* 56 (181 2020).
- [3] J. P. Biersack James F. Ziegler M. D. Ziegler. “SRIM - The stopping and range of ions in matter”. In: *Nuclear Instruments and Methods in Physics Research Section B* 268 (2010), pp. 1818–1823.
- [4] H. Stehfest. “Algorithm 368: Numerical Inversion of Laplace Transform”. In: *Communications of the ACM* 13 (1970), pp. 47–49.

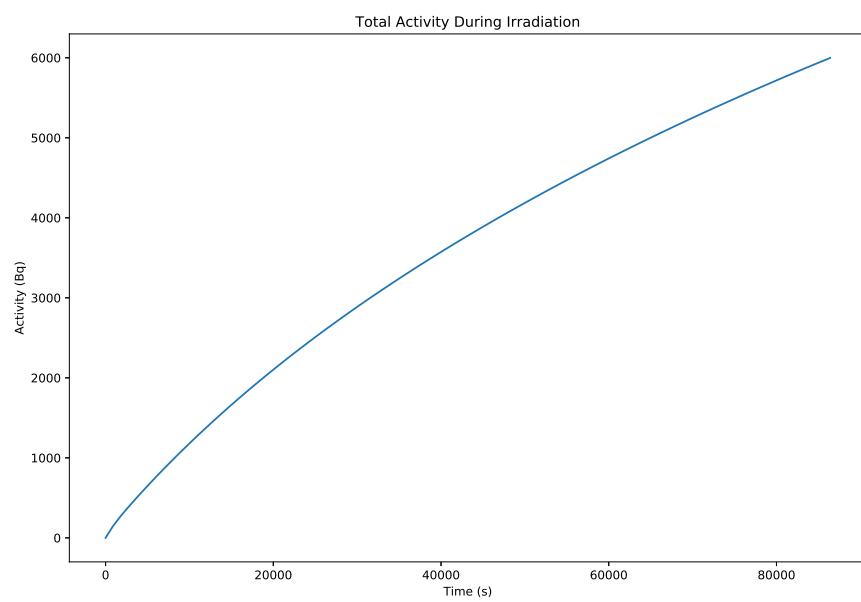


Figure F.5: Gradual saturation of steel with radioactive isotopes under proton irradiation

Appendix G

Activity V1 Full Results for Iron

Fe irradiated by 36MeV protons: output file produced by Activity V1: <https://github.com/BenPalmer1983/activity>

Listing G.1: Isotope Activity at End of Simulation

```
1 =====
2 ACTIVITY code University of Birmingham
3 =====
4
5 Compiled: 15:08:41 11/01/2016
6 MPI Processes: 1
7
8 Date: 21:21 24/06/2016
9 Read user input          0.0000
10 Read Isotope Data       0.0000
11 Read Activity User Input File      0.0720
12 Read Trajectory File        0.0720
13 Data point count: 100887    0.9560
14
15 XS Data Files Loaded:
16 Reading xs file 1_1-26_54_0.dat
17 Reading xs file 1_1-26_56_0.dat
18 Reading xs file 1_1-26_57_0.dat
19 Reading xs file 1_1-26_58_0.dat
20
21 Calculation Settings
22
23 Beam Duration/s:        0.3000000000D+03
24 Beam Current/uA:         0.5000000000D+00
25 Beam Energy/MeV:         0.3600000000D+02
26 Beam Area/mm2:           0.1000000000D+03
27 Target Thickness/Angstrom: 0.5000000000D+07
28 Target Density/kgm-3:     0.8000000000D+04
29 Activity Measurement Time/s: 0.2600000000D+06
30 Input File:              /home/ben/activity/examples/Fe36MeV.in
31 Isotope File:             /home/ben/activity/data/isotopes.txt
32 Decay Modes File:        /home/ben/activity/data/decaymodes.txt
33 Gamma Energies File:     /home/ben/activity/data/gammaenergies.txt
34 XS File Directory:       /home/ben/activity/data/xs
35
36
37
38 Create material tally      1.2640
39 Fill tally with blank data 1.2640
40 Fill tally with isotope data 1.2640
41
```

```

42 -----
43 Starting Isotope Tally - Input Material
44 Element Z A M mk sim Half life Decay Constant Activity Reaction Rate Atoms/mg
45 -----
46     FE 026 054 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.5405383878D+28
47     FE 026 056 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.7890272999D+29
48     FE 026 057 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.1758807969D+28
49     FE 026 058 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.2263810953D+27
50 Total starting isotopes: 4
51
52
53 -----
54 Pre-Simulation Summary
55 1.2840
56 -----
57
58 Beam flux ions/s: 0.3120754681E+13
59
60
61 Calculate projectile-target-product reaction rates 1.2840
62 Reaction rates to calculate: 424
63 Trajectory fit equation:
64     0 0.3665497627E+05
65     1 -0.1783622595E-02
66     2 0.2947794093E-09
67     3 -0.3735372827E-16
68     4 0.1919778602E-23
69     5 -0.3566818320E-31
70 Affected material depth: 0.5000000000E+07
71
72 -----
73 Calculated variables
74 -----
75 Beam flux ions/s: 0.3120754681E+13
76 Affected depth/m: 0.5000000000E-03
77 Affected depth/ang: 0.5000000000E+07
78 Affected volume/m3: 0.500000054E-07
79 Number Density: 0.8626809746E+29
80 VPI: 0.6020000000E+02
81 DPA: 0.1306643605E-04
82
83 -----
84 Reaction rates in the target material 1.2840
85      Sym Z A M      Sym Z A M Reaction Rate Average XS Number Density
86 -----
87     1: FE 26 54 0 NN 0 1 0 0.3508066664E+09 0.4160433001E-28 0.5403805016E+28
88     2: FE 26 54 0 H 1 1 0 0.1351444089E+10 0.1602761044E-27 0.5403805016E+28
89     3: FE 26 54 0 H 1 2 0 0.3205323639E+08 0.3801391343E-29 0.5403805016E+28
90     4: FE 26 54 0 H 1 3 0 0.7165361660E+06 0.8497845101E-31 0.5403805016E+28
91     5: FE 26 54 0 HE 2 3 0 0.8617826538E+07 0.1022041294E-29 0.5403805016E+28
92     6: FE 26 54 0 HE 2 4 0 0.5675883691E+08 0.6731381151E-29 0.5403805016E+28
93     67: FE 26 54 0 TI 22 46 0 0.4687831440E+01 0.5559588940E-36 0.5403805016E+28
94     76: FE 26 54 0 V 23 47 0 0.3452942314E+06 0.4095057628E-31 0.5403805016E+28
95     78: FE 26 54 0 V 23 49 0 0.5791127831E+07 0.6868056297E-30 0.5403805016E+28
96     85: FE 26 54 0 CR 24 49 0 0.2416421986E+07 0.2865784131E-30 0.5403805016E+28
97     86: FE 26 54 0 CR 24 50 0 0.4290008294E+08 0.5087785892E-29 0.5403805016E+28
98     87: FE 26 54 0 CR 24 51 0 0.2135062196E+07 0.2532102172E-30 0.5403805016E+28
99     88: FE 26 54 0 CR 24 52 0 0.8659953579E+08 0.1027037400E-28 0.5403805016E+28
100    92: FE 26 54 0 MN 25 50 0 0.1542445948E+07 0.1829281950E-30 0.5403805016E+28

```

101	93:	FE	26	54	0	MN	25	51	0	0.1778372502E+07	0.2109081827E-30	0.5403805016E+28
102	94:	FE	26	54	0	MN	25	52	0	0.9863724488E+08	0.1169800030E-28	0.5403805016E+28
103	95:	FE	26	54	0	MN	25	53	0	0.2232277074E+09	0.2647395303E-28	0.5403805016E+28
104	99:	FE	26	54	0	FE	26	52	0	0.1143967233E+08	0.1356701421E-29	0.5403805016E+28
105	100:	FE	26	54	0	FE	26	53	0	0.1888610876E+09	0.2239820325E-28	0.5403805016E+28
106	101:	FE	26	54	0	FE	26	54	0	0.1103802931E+09	0.1309068094E-28	0.5403805016E+28
107	104:	FE	26	54	0	CO	27	53	0	0.4251641576E+06	0.5042284431E-31	0.5403805016E+28
108	105:	FE	26	54	0	CO	27	54	0	0.8026455518E+07	0.9519069510E-30	0.5403805016E+28
109	106:	FE	26	54	0	CO	27	55	0	0.1175568630E+06	0.1394179471E-31	0.5403805016E+28
110	107:	FE	26	56	0	NN	0	1	0	0.1294835964E+11	0.1052010224E-27	0.7887968323E+29
111	108:	FE	26	56	0	H	1	1	0	0.1322708565E+11	0.1074655765E-27	0.7887968323E+29
112	109:	FE	26	56	0	H	1	2	0	0.6485018658E+09	0.5268857307E-29	0.7887968323E+29
113	110:	FE	26	56	0	H	1	3	0	0.2804930290E+08	0.2278910553E-30	0.7887968323E+29
114	111:	FE	26	56	0	HE	2	3	0	0.1152252073E+09	0.9361656570E-30	0.7887968323E+29
115	112:	FE	26	56	0	HE	2	4	0	0.8396082900E+09	0.6821532069E-29	0.7887968323E+29
116	169:	FE	26	56	0	TI	22	48	0	0.7417367941E+02	0.6026359420E-36	0.7887968323E+29
117	179:	FE	26	56	0	V	23	49	0	0.1002075285E+08	0.8141521197E-31	0.7887968323E+29
118	181:	FE	26	56	0	V	23	51	0	0.3129496674E+07	0.2542609711E-31	0.7887968323E+29
119	188:	FE	26	56	0	CR	24	50	0	0.3047989029E+03	0.2476387520E-35	0.7887968323E+29
120	189:	FE	26	56	0	CR	24	51	0	0.1903672236E+09	0.1546669008E-29	0.7887968323E+29
121	190:	FE	26	56	0	CR	24	52	0	0.2832038545E+09	0.2300935089E-29	0.7887968323E+29
122	191:	FE	26	56	0	CR	24	53	0	0.1158364999E+07	0.9411322022E-32	0.7887968323E+29
123	192:	FE	26	56	0	CR	24	54	0	0.1165756880E+08	0.9471378543E-31	0.7887968323E+29
124	196:	FE	26	56	0	MN	25	51	0	0.9817102039E+01	0.7976061834E-37	0.7887968323E+29
125	197:	FE	26	56	0	MN	25	52	0	0.1545890268E+09	0.1255983315E-29	0.7887968323E+29
126	198:	FE	26	56	0	MN	25	53	0	0.8815593471E+08	0.7162370153E-30	0.7887968323E+29
127	199:	FE	26	56	0	MN	25	54	0	0.1267995071E+10	0.1030202911E-28	0.7887968323E+29
128	200:	FE	26	56	0	MN	25	55	0	0.7253812202E+09	0.5893475939E-29	0.7887968323E+29
129	204:	FE	26	56	0	FE	26	53	0	0.2156782879E+02	0.1752312805E-36	0.7887968323E+29
130	205:	FE	26	56	0	FE	26	54	0	0.2975937641E+10	0.2417848215E-28	0.7887968323E+29
131	206:	FE	26	56	0	FE	26	55	0	0.4948211136E+10	0.4020253414E-28	0.7887968323E+29
132	207:	FE	26	56	0	FE	26	56	0	0.1551234326E+10	0.1260325181E-28	0.7887968323E+29
133	210:	FE	26	56	0	CO	27	55	0	0.3083905589E+09	0.2505568504E-29	0.7887968323E+29
134	211:	FE	26	56	0	CO	27	56	0	0.2971765420E+09	0.2414458427E-29	0.7887968323E+29
135	212:	FE	26	56	0	CO	27	57	0	0.3041404939E+07	0.2471038170E-31	0.7887968323E+29
136	213:	FE	26	57	0	NN	0	1	0	0.3649443293E+09	0.1330164742E-27	0.1758294237E+28
137	214:	FE	26	57	0	H	1	1	0	0.2670335825E+09	0.9732954535E-28	0.1758294237E+28
138	215:	FE	26	57	0	H	1	2	0	0.1672940269E+08	0.6097604437E-29	0.1758294237E+28
139	216:	FE	26	57	0	H	1	3	0	0.12754404383E+07	0.4648779807E-30	0.1758294237E+28
140	217:	FE	26	57	0	HE	2	3	0	0.2836366558E+07	0.1033811047E-29	0.1758294237E+28
141	218:	FE	26	57	0	HE	2	4	0	0.2332170710E+08	0.8500395818E-29	0.1758294237E+28
142	273:	FE	26	57	0	TI	22	49	0	0.7752833017E+00	0.2825785826E-36	0.1758294237E+28
143	282:	FE	26	57	0	V	23	49	0	0.3281858226E+02	0.1196185760E-34	0.1758294237E+28
144	283:	FE	26	57	0	V	23	50	0	0.1835618293E+06	0.6690540276E-31	0.1758294237E+28
145	284:	FE	26	57	0	V	23	51	0	0.6679861824E+00	0.2434704685E-36	0.1758294237E+28
146	285:	FE	26	57	0	V	23	52	0	0.3050010636E+01	0.1111680957E-35	0.1758294237E+28
147	293:	FE	26	57	0	CR	24	51	0	0.3269383508E+04	0.1191638922E-32	0.1758294237E+28
148	294:	FE	26	57	0	CR	24	52	0	0.7230173808E+07	0.2635284756E-29	0.1758294237E+28
149	295:	FE	26	57	0	CR	24	53	0	0.1653080211E+07	0.6025217646E-30	0.1758294237E+28
150	296:	FE	26	57	0	CR	24	54	0	0.4518951495E+05	0.1647086821E-31	0.1758294237E+28
151	297:	FE	26	57	0	CR	24	55	0	0.7219801309E+04	0.2631504144E-32	0.1758294237E+28
152	301:	FE	26	57	0	MN	25	52	0	0.2058316554E+07	0.7502240448E-30	0.1758294237E+28
153	302:	FE	26	57	0	MN	25	53	0	0.9698739328E+07	0.3535038104E-29	0.1758294237E+28
154	303:	FE	26	57	0	MN	25	54	0	0.2412989059E+07	0.8794966005E-30	0.1758294237E+28
155	304:	FE	26	57	0	MN	25	55	0	0.1977898329E+08	0.7209128654E-29	0.1758294237E+28
156	305:	FE	26	57	0	MN	25	56	0	0.5440385638E+07	0.1982935089E-29	0.1758294237E+28
157	309:	FE	26	57	0	FE	26	54	0	0.2296404190E+07	0.8370032473E-30	0.1758294237E+28
158	310:	FE	26	57	0	FE	26	55	0	0.9449535161E+08	0.3444207100E-28	0.1758294237E+28
159	311:	FE	26	57	0	FE	26	56	0	0.1023020929E+09	0.3728750553E-28	0.1758294237E+28
160	312:	FE	26	57	0	FE	26	57	0	0.2808424683E+08	0.1023626672E-28	0.1758294237E+28
161	315:	FE	26	57	0	CO	27	55	0	0.4515569494E+07	0.1645854134E-29	0.1758294237E+28

```

162 316: FE 26 57 0 CO 27 56 0 0.1114292806E+08 0.4061422208E-29 0.1758294237E+28
163 317: FE 26 57 0 CO 27 57 0 0.5878105202E+07 0.2142476992E-29 0.1758294237E+28
164 318: FE 26 57 0 CO 27 58 0 0.3965181061E+05 0.1445246198E-31 0.1758294237E+28
165 319: FE 26 58 0 NN 0 1 0 0.5793566623E+08 0.1640601429E-27 0.2263149715E+27
166 320: FE 26 58 0 H 1 1 0 0.2791132972E+08 0.7903830301E-28 0.2263149715E+27
167 321: FE 26 58 0 H 1 2 0 0.2037087602E+07 0.5768551650E-29 0.2263149715E+27
168 322: FE 26 58 0 H 1 3 0 0.1593981581E+06 0.4513779906E-30 0.2263149715E+27
169 323: FE 26 58 0 HE 2 3 0 0.2598326212E+06 0.7357847030E-30 0.2263149715E+27
170 324: FE 26 58 0 HE 2 4 0 0.2551528049E+07 0.7225325669E-29 0.2263149715E+27
171 376: FE 26 58 0 TI 22 50 0 0.9741805634E+00 0.2758649600E-35 0.2263149715E+27
172 385: FE 26 58 0 V 23 50 0 0.2503943210E+00 0.7090576626E-36 0.2263149715E+27
173 386: FE 26 58 0 V 23 51 0 0.9310532844E+04 0.2636523317E-31 0.2263149715E+27
174 388: FE 26 58 0 V 23 53 0 0.1852730029E-06 0.5246494484E-42 0.2263149715E+27
175 396: FE 26 58 0 CR 24 52 0 0.4688542450E+04 0.1327684645E-31 0.2263149715E+27
176 397: FE 26 58 0 CR 24 53 0 0.1090733133E+06 0.3088698988E-30 0.2263149715E+27
177 398: FE 26 58 0 CR 24 54 0 0.7904949881E+05 0.2238495372E-30 0.2263149715E+27
178 399: FE 26 58 0 CR 24 55 0 0.5894353392E-05 0.1669141865E-40 0.2263149715E+27
179 400: FE 26 58 0 CR 24 56 0 0.3341354293E-04 0.9461927315E-40 0.2263149715E+27
180 405: FE 26 58 0 MN 25 53 0 0.1200142917E+07 0.3398521693E-29 0.2263149715E+27
181 406: FE 26 58 0 MN 25 54 0 0.1053914972E+07 0.2984438639E-29 0.2263149715E+27
182 407: FE 26 58 0 MN 25 55 0 0.2219588486E+06 0.6285351112E-30 0.2263149715E+27
183 408: FE 26 58 0 MN 25 56 0 0.4947583955E+06 0.1401039090E-29 0.2263149715E+27
184 409: FE 26 58 0 MN 25 57 0 0.3707982841E+06 0.1050013290E-29 0.2263149715E+27
185 414: FE 26 58 0 FE 26 55 0 0.4036879251E+06 0.1143148996E-29 0.2263149715E+27
186 415: FE 26 58 0 FE 26 56 0 0.1504003971E+08 0.4258984535E-28 0.2263149715E+27
187 416: FE 26 58 0 FE 26 57 0 0.8926880873E+07 0.2527882127E-28 0.2263149715E+27
188 417: FE 26 58 0 FE 26 58 0 0.3876895164E+07 0.1097845276E-28 0.2263149715E+27
189 421: FE 26 58 0 CO 27 56 0 0.2685676346E+07 0.7605202007E-29 0.2263149715E+27
190 422: FE 26 58 0 CO 27 57 0 0.3377108415E+07 0.9563174557E-29 0.2263149715E+27
191 423: FE 26 58 0 CO 27 58 0 0.7015543919E+06 0.1986636580E-29 0.2263149715E+27
192 424: FE 26 58 0 CO 27 59 0 0.7735041844E+04 0.2190381423E-31 0.2263149715E+27
193
194
195 Prepare full isotope tally 1.2840
196
197 -----

```

```

198          Full Starting Isotope Tally
199 Key   Element Z A M mk sim Half life      Activity      Reaction Rate      Atoms
200 -----

```

```

201 000001  NN 000 001 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
202 000591  H 001 001 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
203 000592  H 001 002 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
204 000593  H 001 003 0 000 001 0.3885235200D+09 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
205 001183  HE 002 003 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
206 001184  HE 002 004 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
207 008881  P 015 031 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
208 009472  S 016 032 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
209 009473  S 016 033 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
210 009474  S 016 034 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
211 009476  S 016 036 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
212 010062  CL 017 032 0 000 001 0.2980000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
213 010063  CL 017 033 0 000 001 0.2511000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
214 010064  CL 017 034 0 000 001 0.1526400000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
215 010065  CL 017 035 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
216 010066  CL 017 036 0 000 001 0.9492336000D+13 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
217 010067  CL 017 037 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
218 010068  CL 017 038 0 000 001 0.2234400000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
219 010069  CL 017 039 0 000 001 0.3372000000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
220 010070  CL 017 040 0 000 001 0.8100000000D+02 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00

```

221 010071 CL 017 041 0 000 001 0.3840000000D+02 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 222 010653 AR 018 033 0 000 001 0.1730000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 223 010654 AR 018 034 0 000 001 0.8445000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 224 010655 AR 018 035 0 000 001 0.1775000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 225 010656 AR 018 036 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 226 010657 AR 018 037 0 000 001 0.3027456000D+07 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 227 010658 AR 018 038 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 228 010659 AR 018 039 0 000 001 0.8483184000D+10 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 229 010660 AR 018 040 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 230 010661 AR 018 041 0 000 001 0.6576600000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 231 010662 AR 018 042 0 000 001 0.1037534400D+10 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 232 010663 AR 018 043 0 000 001 0.3222000000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 233 011245 K 019 035 0 000 001 0.1780000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 234 011246 K 019 036 0 000 001 0.3420000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 235 011247 K 019 037 0 000 001 0.1226000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 236 011248 K 019 038 0 000 001 0.4581600000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 237 011249 K 019 039 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 238 011250 K 019 040 0 000 001 0.3935692800D+17 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 239 011251 K 019 041 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 240 011252 K 019 042 0 000 001 0.4449600000D+05 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 241 011253 K 019 043 0 000 001 0.8028000000D+05 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 242 011254 K 019 044 0 000 001 0.1327800000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 243 011255 K 019 045 0 000 001 0.1038000000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 244 011256 K 019 046 0 000 001 0.1050000000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 245 011837 CA 020 037 0 000 001 0.1811000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 246 011838 CA 020 038 0 000 001 0.4400000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 247 011839 CA 020 039 0 000 001 0.8596000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 248 011840 CA 020 040 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 249 011841 CA 020 041 0 000 001 0.3216672000D+13 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 250 011842 CA 020 042 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 251 011843 CA 020 043 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 252 011844 CA 020 044 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 253 011845 CA 020 045 0 000 001 0.1404950400D+08 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 254 011846 CA 020 046 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 255 011847 CA 020 047 0 000 001 0.3919104000D+06 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 256 011848 CA 020 048 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 257 012429 SC 021 039 0 000 001 0.3000000000D-06 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 258 012430 SC 021 040 0 000 001 0.1823000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 259 012431 SC 021 041 0 000 001 0.5963000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 260 012432 SC 021 042 0 000 001 0.6813000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 261 012433 SC 021 043 0 000 001 0.1400760000D+05 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 262 012434 SC 021 044 0 000 001 0.1429200000D+05 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 263 012435 SC 021 045 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 264 012436 SC 021 046 0 000 001 0.7239456000D+07 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 265 012437 SC 021 047 0 000 001 0.2893708800D+06 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 266 012438 SC 021 048 0 000 001 0.1572120000D+06 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 267 012439 SC 021 049 0 000 001 0.3432000000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 268 012440 SC 021 050 0 000 001 0.1025000000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 269 013021 TI 022 041 0 000 001 0.8040000000D-01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 270 013022 TI 022 042 0 000 001 0.1990000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 271 013023 TI 022 043 0 000 001 0.5090000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 272 013024 TI 022 044 0 000 001 0.1892160000D+10 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 273 013025 TI 022 045 0 000 001 0.1108800000D+05 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 274 013026 TI 022 046 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 275 013027 TI 022 047 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 276 013028 TI 022 048 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 277 013029 TI 022 049 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 278 013030 TI 022 050 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 279 013031 TI 022 051 0 000 001 0.3456000000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 280 013032 TI 022 052 0 000 001 0.1020000000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 281 013613 V 023 043 0 000 001 0.8000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00

282 013614 V 023 044 0 000 001 0.1110000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 283 013615 V 023 045 0 000 001 0.5470000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 284 013616 V 023 046 0 000 001 0.4225000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 285 013617 V 023 047 0 000 001 0.1956000000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 286 013618 V 023 048 0 000 001 0.1380110400D+07 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 287 013619 V 023 049 0 000 001 0.2851200000D+08 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 288 013620 V 023 050 0 000 001 0.4415040000D+25 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 289 013621 V 023 051 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 290 013622 V 023 052 0 000 001 0.2245800000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 291 013623 V 023 053 0 000 001 0.9600000000D+02 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 292 013624 V 023 054 0 000 001 0.4980000000D+02 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 293 013625 V 023 055 0 000 001 0.6540000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 294 014205 CR 024 045 0 000 001 0.5000000000D-01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 295 014206 CR 024 046 0 000 001 0.2600000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 296 014207 CR 024 047 0 000 001 0.5000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 297 014208 CR 024 048 0 000 001 0.776160000D+05 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 298 014209 CR 024 049 0 000 001 0.253800000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 299 014210 CR 024 050 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 300 014211 CR 024 051 0 000 001 0.2393496000D+07 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 301 014212 CR 024 052 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 302 014213 CR 024 053 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 303 014214 CR 024 054 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 304 014215 CR 024 055 0 000 001 0.2098200000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 305 014216 CR 024 056 0 000 001 0.356400000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 306 014797 MN 025 047 0 000 001 0.1000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 307 014798 MN 025 048 0 000 001 0.1581000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 308 014799 MN 025 049 0 000 001 0.3820000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 309 014800 MN 025 050 0 000 001 0.2838800000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 310 014801 MN 025 051 0 000 001 0.2772000000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 311 014802 MN 025 052 0 000 001 0.4830624000D+06 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 312 014803 MN 025 053 0 000 001 0.1179446400D+15 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 313 014804 MN 025 054 0 000 001 0.2696716800D+08 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 314 014805 MN 025 055 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 315 014806 MN 025 056 0 000 001 0.9284040000D+04 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 316 014807 MN 025 057 0 000 001 0.8540000000D+02 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 317 015389 FE 026 049 0 000 001 0.7000000000D-01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 318 015390 FE 026 050 0 000 001 0.1550000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 319 015391 FE 026 051 0 000 001 0.3050000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 320 015392 FE 026 052 0 000 001 0.2979000000D+05 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 321 015393 FE 026 053 0 000 001 0.5106000000D+03 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 322 015394 FE 026 054 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.5405383878D+28
 323 015395 FE 026 055 0 000 001 0.8631403200D+08 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 324 015396 FE 026 056 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.7890272999D+29
 325 015397 FE 026 057 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.1758807969D+28
 326 015398 FE 026 058 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.2263810953D+27
 327 015981 CO 027 051 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 328 015982 CO 027 052 0 000 001 0.1150000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 329 015983 CO 027 053 0 000 001 0.2400000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 330 015984 CO 027 054 0 000 001 0.1932300000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 331 015985 CO 027 055 0 000 001 0.6310800000D+05 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 332 015986 CO 027 056 0 000 001 0.6672931200D+07 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 333 015987 CO 027 057 0 000 001 0.2347833600D+08 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 334 015988 CO 027 058 0 000 001 0.6122304000D+07 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 335 015989 CO 027 059 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00
 336
 337
 338
 339
 340
 341
 342

```

343
344
345
346
347 =====
348
349 START of SIMULATION
350
351 Simulation time from 0.000000000E+00 to 0.260000000E+06
352 Beam from 0.000000000E+00 to 0.300000000E+03
353
354 =====
355
356
357
358
359 -----
360 Simulation tally at 0.300000000E+03
361 -----
362
363 -----
364 Isotope Tally 1.2880
365 Key Element Z A M mk sim Half life Decay Constant Activity Reaction Rate Start Atoms Beam
      End Atoms Atoms At Time
366 -----
367 000001 NN 000 001 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.4116613889D+13 0.4116613889D+13
368 000593 H 001 003 0 000 001 0.3885235200D+09 0.1784054619D-08 0.1616389319D+02 0.000000000D+00 0.000000000D
      +00 0.9060200854D+10 0.9060200854D+10
369 001183 HE 002 003 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.242400000D+04 0.242400000D+04
370 013027 TI 022 047 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.5316233671D+07 0.5316233671D+07
371 013029 TI 022 049 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.1736864062D+05 0.1736864062D+05
372 013617 V 023 047 0 000 001 0.195600000D+04 0.3543697242D-03 0.3482463420D+05 0.000000000D+00 0.000000000D
      +00 0.9827203574D+08 0.9827203574D+08
373 013619 V 023 049 0 000 001 0.285120000D+08 0.2431071761D-07 0.1160219131D+03 0.000000000D+00 0.000000000D
      +00 0.4772459413D+10 0.4772459413D+10
374 013621 V 023 051 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.2509195042D+07 0.2509195042D+07
375 013622 V 023 052 0 000 001 0.224580000D+03 0.3086415445D-02 0.1841703348D+01 0.000000000D+00 0.000000000D
      +00 0.5967127177D+03 0.5967127177D+03
376 013623 V 023 053 0 000 001 0.960000000D+02 0.7220283131D-02 0.1640359913D-06 0.000000000D+00 0.000000000D
      +00 0.2271877547D-04 0.2271877547D-04
377 014209 CR 024 049 0 000 001 0.253800000D+04 0.2731076362D-03 0.1900894319D+06 0.000000000D+00 0.000000000D
      +00 0.6960238630D+09 0.6960238630D+09
378 014210 CR 024 050 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.4621020723D+09 0.4621020723D+09
379 014211 CR 024 051 0 000 001 0.2393496000D+07 0.2895961307D-06 0.1672958542D+05 0.000000000D+00 0.000000000D
      +00 0.5776867728D+11 0.5776867728D+11
380 014212 CR 024 052 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.1648354473D+08 0.1648354473D+08
381 014213 CR 024 053 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.4169728000D+07 0.4169728000D+07
382 014214 CR 024 054 0 000 001 -0.100000000D+01 0.000000000D+00 0.000000000D+00 0.000000000D+00 0.000000000D
      +00 0.1470632000D+07 0.1470632000D+07

```

383 014215 CR 024 055 0 000 001 0.2098200000D+03 0.3303532459D-02 0.4539932896D+04 0.0000000000D+00 0.0000000000D
 +00 0.1374266169D+07 0.1374266169D+07
 384 014216 CR 024 056 0 000 001 0.3564000000D+03 0.1944857409D-02 0.1476992152D-04 0.0000000000D+00 0.0000000000D
 +00 0.7594346737D-02 0.7594346737D-02
 385 014800 MN 025 050 0 000 001 0.2838800000D+00 0.2441690787D+01 0.1542445948D+07 0.0000000000D+00 0.0000000000D
 +00 0.6317122367D+06 0.6317122367D+06
 386 014801 MN 025 051 0 000 001 0.2772000000D+04 0.2500530954D-03 0.1285259866D+06 0.0000000000D+00 0.0000000000D
 +00 0.5139947831D+09 0.5139947831D+09
 387 014802 MN 025 052 0 000 001 0.4830624000D+06 0.1434901952D-05 0.1098860016D+06 0.0000000000D+00 0.0000000000D
 +00 0.7658084333D+11 0.7658084333D+11
 388 014803 MN 025 053 0 000 001 0.1179446400D+15 0.5876885805D-14 0.6277577829D-03 0.0000000000D+00 0.0000000000D
 +00 0.1068181012D+12 0.1068181012D+12
 389 014804 MN 025 054 0 000 001 0.2696716800D+08 0.2570337310D-07 0.9804220660D+04 0.0000000000D+00 0.0000000000D
 +00 0.3814371220D+12 0.3814371220D+12
 390 014805 MN 025 055 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
 +00 0.2614130226D+07 0.2614130226D+07
 391 014806 MN 025 056 0 000 001 0.9284040000D+04 0.7466008123D-04 0.1314578074D+06 0.0000000000D+00 0.0000000000D
 +00 0.1760750930D+10 0.1760750930D+10
 392 014807 MN 025 057 0 000 001 0.8540000000D+02 0.8116477524D-02 0.3383153467D+06 0.0000000000D+00 0.0000000000D
 +00 0.4168253355D+08 0.4168253355D+08
 393 015392 FE 026 052 0 000 001 0.2979000000D+05 0.2326778048D-04 0.7957468329D+05 0.0000000000D+00 0.0000000000D
 +00 0.3419951609D+10 0.3419951609D+10
 394 015393 FE 026 053 0 000 001 0.5106000000D+03 0.1357515042D-02 0.6332089595D+08 0.0000000000D+00 0.0000000000D
 +00 0.4664471036D+11 0.4664471036D+11
 395 015394 FE 026 054 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.5405383878D
 +28 0.5405383878D+28 0.5405383878D+28
 396 015395 FE 026 055 0 000 001 0.8631403200D+08 0.8030527187D-08 0.1215087645D+05 0.0000000000D+00 0.0000000000D
 +00 0.1513085775D+13 0.1513085775D+13
 397 015396 FE 026 056 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.7890272999D
 +29 0.7890272999D+29 0.7890272999D+29
 398 015397 FE 026 057 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.1758807969D
 +28 0.1758807969D+28 0.1758807969D+28
 399 015398 FE 026 058 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.2263810953D
 +27 0.2263810953D+27 0.2263810953D+27
 400 015983 CO 027 053 0 000 001 0.2400000000D+00 0.2888113252D+01 0.4251641576D+06 0.0000000000D+00 0.0000000000D
 +00 0.1472117332D+06 0.1472117332D+06
 401 015984 CO 027 054 0 000 001 0.1932300000D+00 0.3587161313D+01 0.8026455518D+07 0.0000000000D+00 0.0000000000D
 +00 0.2237550759D+07 0.2237550759D+07
 402 015985 CO 027 055 0 000 001 0.6310800000D+05 0.1098350733D-04 0.1029731940D+07 0.0000000000D+00 0.0000000000D
 +00 0.9375256095D+11 0.9375256095D+11
 403 015986 CO 027 056 0 000 001 0.6672931200D+07 0.1038744683D-06 0.9691497262D+04 0.0000000000D+00 0.0000000000D
 +00 0.9330009018D+11 0.9330009018D+11
 404 015987 CO 027 057 0 000 001 0.2347833600D+08 0.2952284100D-07 0.1089088520D+03 0.0000000000D+00 0.0000000000D
 +00 0.3688969230D+10 0.3688969230D+10
 405 015988 CO 027 058 0 000 001 0.6122304000D+07 0.1132167205D-06 0.2517465310D+02 0.0000000000D+00 0.0000000000D
 +00 0.2223580845D+09 0.2223580845D+09
 406
 407 -----
 408 Simulation tally at 0.2600000000E+06
 409 -----
 410
 411 -----

412 Isotope Tally 1.7640
 413 Key Element Z A M mk sim Half life Decay Constant Activity Reaction Rate Start Atoms Beam
 End Atoms At Time
 414 -----

415 000001 NN 000 001 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
 +00 0.4116613889D+13 0.4116613889D+13
 416 000593 H 001 003 0 000 001 0.3885235200D+09 0.1784054619D-08 0.1615640588D+02 0.0000000000D+00 0.0000000000D

```

+00 0.9060200854D+10 0.9056004063D+10
417 001183 HE 002 003 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.2424000000D+04 0.4199214759D+07
418 013027 TI 022 047 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.5316233671D+07 0.1035882694D+09
419 013029 TI 022 049 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.1736864062D+05 0.6440824599D+08
420 013619 V 023 049 0 000 001 0.2851200000D+08 0.2431071761D-07 0.1313773643D+03 0.0000000000D+00 0.0000000000D
+00 0.4772459413D+10 0.5404092399D+10
421 013621 V 023 051 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.2509195042D+07 0.8409826886D+10
422 014210 CR 024 050 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.4621020723D+09 0.4627337845D+09
423 014211 CR 024 051 0 000 001 0.2393496000D+07 0.2895961307D-06 0.1444370965D+05 0.0000000000D+00 0.0000000000D
+00 0.5776867728D+11 0.4987535437D+11
424 014212 CR 024 052 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.1648354473D+08 0.4857309090D+11
425 014213 CR 024 053 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.4169728000D+07 0.4170359063D+07
426 014214 CR 024 054 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.1470632000D+07 0.2539147607D+10
427 014802 MN 025 052 0 000 001 0.4830624000D+06 0.1434901952D-05 0.4510767081D+05 0.0000000000D+00 0.0000000000D
+00 0.7658084333D+11 0.3143606485D+11
428 014803 MN 025 053 0 000 001 0.1179446400D+15 0.5876885805D-14 0.1176009917D-02 0.0000000000D+00 0.0000000000D
+00 0.1068181012D+12 0.2001076685D+12
429 014804 MN 025 054 0 000 001 0.2696716800D+08 0.2570337310D-07 0.9738993802D+04 0.0000000000D+00 0.0000000000D
+00 0.3814371220D+12 0.3788994450D+12
430 014805 MN 025 055 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.0000000000D
+00 0.2614130226D+07 0.6439401457D+10
431 015392 FE 026 052 0 000 001 0.2979000000D+05 0.2326778048D-04 0.1890116552D+03 0.0000000000D+00 0.0000000000D
+00 0.3419951609D+10 0.8123321232D+07
432 015394 FE 026 054 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.5405383878D
+28 0.5405383878D+28 0.5405383878D+28
433 015395 FE 026 055 0 000 001 0.8631403200D+08 0.8030527187D-08 0.1280863433D+05 0.0000000000D+00 0.0000000000D
+00 0.1513085775D+13 0.1594992960D+13
434 015396 FE 026 056 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.7890272999D
+29 0.7890272999D+29 0.7890272999D+29
435 015397 FE 026 057 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.1758807969D
+28 0.1758807969D+28 0.1758807969D+28
436 015398 FE 026 058 0 000 001 -0.1000000000D+01 0.0000000000D+00 0.0000000000D+00 0.0000000000D+00 0.2263810953D
+27 0.2263810953D+27 0.2263810953D+27
437 015985 CO 027 055 0 000 001 0.6310800000D+05 0.1098350733D-04 0.5942036078D+05 0.0000000000D+00 0.0000000000D
+00 0.9375256095D+11 0.5409962319D+10
438 015986 CO 027 056 0 000 001 0.6672931200D+07 0.1038744683D-06 0.9433552328D+04 0.0000000000D+00 0.0000000000D
+00 0.9330009018D+11 0.9081685308D+11
439 015987 CO 027 057 0 000 001 0.2347833600D+08 0.2952284100D-07 0.1080770318D+03 0.0000000000D+00 0.0000000000D
+00 0.3688969230D+10 0.3660793752D+10
440 015988 CO 027 058 0 000 001 0.6122304000D+07 0.1132167205D-06 0.2444523420D+02 0.0000000000D+00 0.0000000000D
+00 0.2223580845D+09 0.2159154063D+09
441
442 -----
443 Gamma Lines at end of Simulation 0.2600000000E+06
444 -----

```

Z	A	M	Gamma Energy/KeV	Gamma count/s
24	51	0	0.3200824000E+03	0.1432815997E+04
25	52	0	0.2005800000E+03	0.3428182982E+02
25	52	0	0.3460199000E+03	0.4420551289E+03
25	52	0	0.3980899000E+03	0.4014582702E+02
25	52	0	0.3995700000E+03	0.8254703759E+02
25	52	0	0.5020600000E+03	0.9472610871E+02
25	52	0	0.6001600000E+03	0.1759199162E+03

453	25	52	0	0.6474699000E+03	0.1804306833E+03
454	25	52	0	0.7442329000E+03	0.4059690373E+05
455	25	52	0	0.8481800000E+03	0.1497574671E+04
456	25	52	0	0.9018899000E+03	0.1984737065E+02
457	25	52	0	0.9355440000E+03	0.4262674892E+05
458	25	52	0	0.1045750000E+04	0.3157536957E+02
459	25	52	0	0.1246278000E+04	0.1899032941E+04
460	25	52	0	0.1247879000E+04	0.1714091491E+03
461	25	52	0	0.1333648000E+04	0.2286958910E+04
462	25	52	0	0.1434092000E+04	0.4510767081E+05
463	25	52	0	0.1645819000E+04	0.2120060528E+02
464	25	52	0	0.1981120000E+04	0.1533660808E+02
465	25	52	0	0.2257419000E+04	0.1217907112E+01
466	25	54	0	0.8348479000E+03	0.9736656444E+04
467	26	52	0	0.1686880000E+03	0.1874526871E+03
468	26	52	0	0.3777479000E+03	0.3106047431E+01
469	26	52	0	0.1039928000E+04	0.1798237987E+00
470	26	55	0	0.1259999000E+03	0.1639505194E-04
471	27	55	0	0.9190000000E+02	0.6907616941E+03
472	27	55	0	0.3853999000E+03	0.3208699482E+03
473	27	55	0	0.4114999000E+03	0.6372833694E+03
474	27	55	0	0.4771999000E+03	0.1198805779E+05
475	27	55	0	0.5199999000E+03	0.4902179764E+03
476	27	55	0	0.8036999000E+03	0.1109675238E+04
477	27	55	0	0.8270000000E+03	0.1247827576E+03
478	27	55	0	0.9311000000E+03	0.4456527059E+05
479	27	55	0	0.9845999000E+03	0.3075003670E+03
480	27	55	0	0.1212799000E+04	0.1559783876E+03
481	27	55	0	0.1316600000E+04	0.4211417476E+04
482	27	55	0	0.1369999000E+04	0.1733589026E+04
483	27	55	0	0.1408499000E+04	0.1002718588E+05
484	27	55	0	0.1555999000E+04	0.2718481506E+02
485	27	55	0	0.1622299000E+04	0.2673916235E+02
486	27	55	0	0.1792100000E+04	0.4857614494E+02
487	27	55	0	0.1940599000E+04	0.8467401411E+01
488	27	55	0	0.2144200000E+04	0.5347832470E+02
489	27	55	0	0.2177600000E+04	0.1738045553E+03
490	27	55	0	0.2578700000E+04	0.2540220423E+02
491	27	55	0	0.2872400000E+04	0.6996747482E+02
492	27	55	0	0.2938900000E+04	0.3386960565E+02
493	27	55	0	0.3108300000E+04	0.3119568941E+01
494	27	56	0	0.2634099000E+03	0.2075380569E+01
495	27	56	0	0.4113799000E+03	0.2358388082E+01
496	27	56	0	0.4865399000E+03	0.5754466920E+01
497	27	56	0	0.6550000000E+03	0.3584749885E+01
498	27	56	0	0.6746999000E+03	0.3584749885E+01
499	27	56	0	0.7335109000E+03	0.1792374942E+02
500	27	56	0	0.7877419000E+03	0.2971568983E+02
501	27	56	0	0.8467709000E+03	0.9427420519E+04
502	27	56	0	0.8527800000E+03	0.4716776164E+01
503	27	56	0	0.8965309000E+03	0.8112855002E+01
504	27	56	0	0.9773729000E+03	0.1366921732E+03
505	27	56	0	0.9973299000E+03	0.1216928250E+02
506	27	56	0	0.1037839000E+04	0.1336733422E+04
507	27	56	0	0.1089030000E+04	0.4716776164E+01
508	27	56	0	0.1140404000E+04	0.1292396669E+02
509	27	56	0	0.1159919000E+04	0.8961874712E+01
510	27	56	0	0.1175101000E+04	0.2158396773E+03
511	27	56	0	0.1198780000E+04	0.4811111687E+01
512	27	56	0	0.1238281000E+04	0.6311046508E+04
513	27	56	0	0.1272200000E+04	0.2264052559E+01

514	27	56	0	0.1335389000E+04	0.1113159175E+02
515	27	56	0	0.1360215000E+04	0.4046993949E+03
516	27	56	0	0.1442750000E+04	0.1745207181E+02
517	27	56	0	0.1462340000E+04	0.6131809013E+01
518	27	56	0	0.1640404000E+04	0.6792156733E+01
519	27	56	0	0.1771351000E+04	0.1459370545E+04
520	27	56	0	0.1810771000E+04	0.6018606385E+02
521	27	56	0	0.1963714000E+04	0.6829891886E+02
522	27	56	0	0.2015181000E+04	0.2867799908E+03
523	27	56	0	0.2034755000E+04	0.7443072787E+03
524	27	56	0	0.2113123000E+04	0.3547015675E+02
525	27	56	0	0.2212933000E+04	0.3726253170E+02
526	27	56	0	0.2276360000E+04	0.1207494698E+02
527	27	56	0	0.2373700000E+04	0.7735512909E+01
528	27	56	0	0.2523860000E+04	0.6414815583E+01
529	27	56	0	0.2598459000E+04	0.1632004553E+04
530	27	56	0	0.2657400000E+04	0.1981045989E+01
531	27	56	0	0.3009595000E+04	0.1094292070E+03
532	27	56	0	0.3201962000E+04	0.3131939373E+03
533	27	56	0	0.3253415000E+04	0.7660044491E+03
534	27	56	0	0.3272990000E+04	0.1820674656E+03
535	27	56	0	0.3369690000E+04	0.1037689813E+01
536	27	56	0	0.3451152000E+04	0.9169411920E+02
537	27	56	0	0.3547930000E+04	0.1886710466E+02
538	27	56	0	0.3600700000E+04	0.1556536134E+01
539	27	56	0	0.3611799000E+04	0.8018519479E+00
540	27	57	0	0.1441290000E+02	0.9899856117E+01
541	27	57	0	0.1220606000E+03	0.9251393926E+02
542	27	57	0	0.1364735000E+03	0.1154261619E+02
543	27	57	0	0.3396899000E+03	0.3998850178E-02
544	27	57	0	0.3523300000E+03	0.3242310955E-02
545	27	57	0	0.3667999000E+03	0.1296924382E-02
546	27	57	0	0.5700900000E+03	0.1707617103E-01
547	27	57	0	0.6924099000E+03	0.1610347774E+00
548	27	57	0	0.7065399000E+03	0.5403851592E-02
549	27	58	0	0.8107592000E+03	0.2431078541E+02
550	27	58	0	0.8639510000E+03	0.1686720915E+00
551	27	58	0	0.1674725000E+04	0.1271152178E+00
552					
553					
554					Activities Ordered by Most Active
555					
556					
557	15985 CO	27	55	0	0.5942036078E+05
558	14802 MN	25	52	0	0.4510767081E+05
559	14211 CR	24	51	0	0.1444370965E+05
560	15395 FE	26	55	0	0.1280863433E+05
561	14804 MN	25	54	0	0.9738993802E+04
562	15986 CO	27	56	0	0.9433552328E+04
563	15392 FE	26	52	0	0.1890116552E+03
564	13619 V	23	49	0	0.1313773643E+03
565	15987 CO	27	57	0	0.1080770318E+03
566	15988 CO	27	58	0	0.2444523420E+02
567	593 H	1	3	0	0.1615640588E+02
568	14803 MN	25	53	0	0.1176009917E-02
569					
570					

571	Totals	1.8880
572		

573
574 Total Activity/Bq: 0.1514219906E+06
575 Total Gamma Power/eV/s: 0.2559334931E+12
576 Total Gamma Power/Watts: 0.4100506435E-07
577 Absorbed Dose*/Grays/s: 0.4078849177E-10
578 Absorbed Dose*/Grays/hr: 0.1468385704E-06
579 Fraction of annual dosage if exposed for 1 hr: 0.1468385634E-03
580
581 Absorbed dose, assumes all energy absorbed, 80kg human, 1m from point-target, 1m surface area exposed to
irradiation.
582
583 Dose Limits:
584 employees 18+ 20 millisieverts/year
585 trainees 18+ 6 millisieverts/year
586 public and under 18s 1 millisieverts/year
587 public and under 18s millisieverts averaged per hour: 0.1140771128E-03
588 Dose averaged over area of skin not exceeding 1cm²
589 Source: <http://www.hse.gov.uk/radiation/ionising/doses/>
590
591
592
593 -----
594 Activity calculation complete. 1.8880
595 -----

Appendix H

Activity V2 Full Results for Iron

Fe irradiated by 36MeV protons: output file produced by Activity V2: https://github.com/BenPalmer1983/activity_v2

Listing H.1: Isotope Activity at End of Simulation

1	3.209E-03	### Start	
2	4.769E-01	### Loading isotope tendl xs	
3	5.289E+00	### Load complete	
4	5.292E+00	### Load EXYZ.txt	
5	5.500E+00	### Run Simulation	
6	5.502E+00	### Prep sim1	
7	5.514E+00	###	
8	5.517E+00	### #####	
9	5.520E+00	### Starting Tally	
10	5.522E+00	### #####	
11	5.523E+00	###	
12	5.526E+00	### 26054 Fe54	1.630585560412975e+20
13	5.528E+00	### 26056 Fe56	2.468222518377967e+21
14	5.530E+00	### 26057 Fe57	5.600277953618698e+19
15	5.531E+00	### 26058 Fe58	7.334611933451605e+18
16	5.533E+00	### 27053 Co53	0.0
17	5.535E+00	### 25050 Mn50	0.0
18	5.536E+00	### 26053 Fe53	0.0
19	5.538E+00	### 26052 Fe52	0.0
20	5.540E+00	### 25051 Mn51	0.0
21	5.542E+00	### 25052 Mn52	0.0
22	5.543E+00	### 24049 Cr49	0.0
23	5.545E+00	### 27055 Co55	0.0
24	5.547E+00	### 23047 V47	0.0
25	5.548E+00	### 25053 Mn53	0.0
26	5.550E+00	### 24050 Cr50	0.0
27	5.552E+00	### 26055 Fe55	0.0
28	5.553E+00	### 25054 Mn54	0.0
29	5.555E+00	### 24051 Cr51	0.0
30	5.557E+00	### 27057 Co57	0.0
31	5.558E+00	### 23049 V49	0.0
32	5.560E+00	### 25055 Mn55	0.0
33	5.562E+00	### 24052 Cr52	0.0
34	5.563E+00	### 27056 Co56	0.0
35	5.565E+00	### 27058 Co58	0.0
36	5.567E+00	### 23050 V50	0.0
37	5.568E+00	### 25056 Mn56	0.0
38	5.570E+00	### 24053 Cr53	0.0
39	5.572E+00	### 27059 Co59	0.0
40	5.573E+00	### 23051 V51	0.0
41	5.575E+00	### 25057 Mn57	0.0

```

42  5.577E+00 ### 24054      Cr54      0.0
43  5.578E+00 ### 1          Nn1       0.0
44  5.580E+00 ### 2004      He4       0.0
45  5.582E+00 ### 1001      H1        0.0
46  5.584E+00 ### 1002      H2        0.0
47  5.585E+00 ### 2003      He3       0.0
48  5.587E+00 ### 1003      H3        0.0
49  5.589E+00 ### 1025052   Mn52-M    0.0
50  5.590E+00 ### 22049     Ti49       0.0
51  5.592E+00 ### 22047     Ti47       0.0
52  5.594E+00 ### 22050     Ti50       0.0
53  5.596E+00 ### ##### #####
54  5.601E+00 ### Run sim1
55  5.603E+00 ###
56  5.605E+00 ### ##### #####
57  5.606E+00 ### ##### #####
58  5.608E+00 ### Run Sim
59  5.610E+00 ### ##### #####
60  5.612E+00 ### ##### #####
61  5.613E+00 ###
62  5.615E+00 ###
63  5.617E+00 ### ##### #####
64  5.618E+00 ### Plot Ion Energy Lost
65  5.620E+00 ### ##### #####
66  5.622E+00 ###
67  1.424E+01 ###
68  1.424E+01 ### ##### #####
69  1.424E+01 ### Number Density
70  1.424E+01 ### ##### #####
71  1.424E+01 ###
72  1.425E+01 ### Fe54      5.095579876290546e+27
73  1.425E+01 ### Fe56      7.713195369931147e+28
74  1.425E+01 ### Fe57      1.750086860505843e+27
75  1.425E+01 ### Fe58      2.2920662292036265e+26
76  1.426E+01 ###
77  1.426E+01 ### ##### #####
78  1.426E+01 ### Calculate reaction rates
79  1.426E+01 ### ##### #####
80  1.426E+01 ###
81  1.429E+01 ### Fe54      --> Co53      307283.956904
82  1.432E+01 ### Fe54      --> Mn50      413574.876348
83  1.436E+01 ### Fe54      --> Fe53      236075945.829
84  1.440E+01 ### Fe54      --> Fe52      131778.988756
85  1.444E+01 ### Fe54      --> Mn51      3915365.42731
86  1.449E+01 ### Fe54      --> Mn52      4550309.60658
87  1.452E+01 ### Fe54      --> Cr49      22.5007021259
88  1.457E+01 ### Fe54      --> Co55      207974.632938
89  1.461E+01 ### Fe54      --> Fe54      114453151.897
90  1.464E+01 ### Fe54      --> V47       12118.304033
91  1.467E+01 ### Fe54      --> Mn53      403877118.991
92  1.471E+01 ### Fe54      --> Cr50      37369410.8242
93  1.475E+01 ### Fe56      --> Co55      845335432.044
94  1.478E+01 ### Fe56      --> Mn52      329450373.437
95  1.481E+01 ### Fe56      --> Mn51      2.53001350218
96  1.485E+01 ### Fe56      --> Fe55      8050196691.17
97  1.490E+01 ### Fe56      --> Fe54      378463315.972
98  1.493E+01 ### Fe56      --> Fe53      8.38602414208
99  1.498E+01 ### Fe56      --> Mn53      109015450.05
100 1.503E+01 ### Fe56      --> Mn54      56915074.2745
101 1.506E+01 ### Fe56      --> Cr51      120955.374078
102 1.510E+01 ### Fe56      --> Co57      5848557.76264

```

103	1.513E+01	### Fe56	--> Fe56	1680968637.75
104	1.517E+01	### Fe56	--> V49	1174843.18811
105	1.522E+01	### Fe56	--> Mn55	715362266.42
106	1.526E+01	### Fe56	--> Cr52	374707737.696
107	1.529E+01	### Fe57	--> Fe54	52183.7852258
108	1.532E+01	### Fe57	--> Co56	36640319.083
109	1.535E+01	### Fe57	--> Co55	1074873.1321
110	1.538E+01	### Fe57	--> Mn53	19968722.6313
111	1.541E+01	### Fe57	--> Mn52	4360.74832679
112	1.547E+01	### Fe57	--> Fe56	164897136.47
113	1.552E+01	### Fe57	--> Fe55	33178633.716
114	1.558E+01	### Fe57	--> Mn54	2427065.011
115	1.563E+01	### Fe57	--> Mn55	2319212.32735
116	1.568E+01	### Fe57	--> Cr52	551009.138609
117	1.573E+01	### Fe57	--> Co58	102227.683704
118	1.577E+01	### Fe57	--> Fe57	27066748.1304
119	1.580E+01	### Fe57	--> V50	28017.8372826
120	1.583E+01	### Fe57	--> Mn56	5147637.47599
121	1.587E+01	### Fe57	--> Cr53	1928918.81162
122	1.590E+01	### Fe58	--> Fe55	6414.68780283
123	1.594E+01	### Fe58	--> Co57	11190415.5404
124	1.598E+01	### Fe58	--> Co56	692558.874587
125	1.606E+01	### Fe58	--> Mn54	2024539.65839
126	1.611E+01	### Fe58	--> Mn53	34624.323756
127	1.616E+01	### Fe58	--> Fe57	11896984.9875
128	1.621E+01	### Fe58	--> Fe56	7789545.04264
129	1.627E+01	### Fe58	--> Mn55	232716.626174
130	1.632E+01	### Fe58	--> Mn56	46497.0277258
131	1.635E+01	### Fe58	--> Cr53	607.862336665
132	1.639E+01	### Fe58	--> Co59	13432.4749748
133	1.643E+01	### Fe58	--> Fe58	4608236.90701
134	1.647E+01	### Fe58	--> V51	3894.08635149
135	1.651E+01	### Fe58	--> Mn57	97973.3183318
136	1.654E+01	### Fe58	--> Cr54	69429.6222128
137	1.655E+01	###		
138	1.665E+01	### Fe54	--> Nn1	223779307.551
139	1.675E+01	### Fe54	--> He4	41722308.3148
140	1.690E+01	### Fe54	--> H1	1185614187.94
141	1.696E+01	### Fe54	--> H2	16816480.0199
142	1.701E+01	### Fe54	--> He3	1195803.75217
143	1.706E+01	### Fe54	--> H3	63495.6126689
144	1.719E+01	### Fe56	--> Nn1	10459239812.3
145	1.729E+01	### Fe56	--> He4	815365435.08
146	1.742E+01	### Fe56	--> H1	11592247091.2
147	1.748E+01	### Fe56	--> H2	390691314.24
148	1.753E+01	### Fe56	--> H3	7727635.45562
149	1.758E+01	### Fe56	--> He3	19451763.8089
150	1.776E+01	### Fe57	--> Nn1	317362293.916
151	1.783E+01	### Fe57	--> H2	11740193.0149
152	1.795E+01	### Fe57	--> He4	24925914.0685
153	1.811E+01	### Fe57	--> H1	229114996.365
154	1.817E+01	### Fe57	--> H3	530012.860186
155	1.820E+01	### Fe57	--> He3	621761.67143
156	1.836E+01	### Fe58	--> Nn1	52586048.5862
157	1.843E+01	### Fe58	--> H2	1338414.90645
158	1.855E+01	### Fe58	--> He4	2368936.32236
159	1.869E+01	### Fe58	--> H1	23169852.7974
160	1.873E+01	### Fe58	--> H3	63386.958919
161	1.878E+01	### Fe58	--> He3	45022.5126778
162	1.878E+01	###		
163	1.878E+01	###		

```

164 1.878E+01 ### #####
165 1.878E+01 ### Saturation Times
166 1.878E+01 ### #####
167 1.879E+01 ###
168 1.879E+01 ### Time where saturation is 95%
169 1.879E+01 ### Co53      1.03726274277
170 1.879E+01 ### Mn50      1.22392681719
171 1.880E+01 ### Fe53      2206.77648525
172 1.880E+01 ### Fe52      128719.120064
173 1.880E+01 ### Mn51      11980.384679
174 1.880E+01 ### Mn52      2089254.61649
175 1.881E+01 ### Cr49      10865.3272305
176 1.881E+01 ### Co55      272748.238212
177 1.881E+01 ### V47       8453.6913536
178 1.881E+01 ### Mn53      5.01914153515e+14
179 1.881E+01 ### Cr50      2.45501506924e+25
180 1.881E+01 ### Fe55      374654548.569
181 1.882E+01 ### Mn54      116576230.889
182 1.882E+01 ### Cr51      10345096.7458
183 1.882E+01 ### Co57      101493998.416
184 1.882E+01 ### V49       123226813.841
185 1.882E+01 ### Co56      28841047.3435
186 1.883E+01 ### Co58      26456423.5172
187 1.883E+01 ### V50       1.90941486654e+25
188 1.883E+01 ### Mn56      40123.094881
189 1.883E+01 ### Mn57      369.092659303
190 1.883E+01 ### Nn1       2656.25700712
191 1.883E+01 ### H3        1681683831.36
192 1.884E+01 ###
193 2.136E+01 ###
194 2.136E+01 ### #####
195 2.137E+01 ### End of Beam Tally
196 2.137E+01 ### #####
197 2.137E+01 ###
198 2.137E+01 ### 26054     Fe54       1.63058555949e+20    0.0
199 2.137E+01 ### 26056     Fe56       2.4682225152e+21    0.0
200 2.138E+01 ### 26057     Fe57       5.60027794695e+19    0.0
201 2.138E+01 ### 26058     Fe58       7.33461192328e+18    0.0
202 2.138E+01 ### 27053     Co53      106396.089785      307283.956904
203 2.138E+01 ### 26053     Fe53       58250591793.0      79076054.5775
204 2.138E+01 ### 25053     Mn53      172540199605.0      0.00102982599877
205 2.139E+01 ### 24053     Cr53      578858002.186      0.0
206 2.139E+01 ### 25050     Mn50      0.0          0.0
207 2.139E+01 ### 24050     Cr50      0.0          0.0
208 2.139E+01 ### 26052     Fe52       39396004.7064      916.8791916
209 2.139E+01 ### 1025052   Mn52-M    130481.519119      71.1029065191
210 2.139E+01 ### 24052     Cr52      112577624050.0      0.0
211 2.140E+01 ### 25052     Mn52      100179964870.0      143645.657909
212 2.140E+01 ### 25051     Mn51      1131634453.28      282968.697949
213 2.140E+01 ### 24051     Cr51      79259717.7947      22.9520226273
214 2.140E+01 ### 23051     V51       1168225.90545      0.0
215 2.140E+01 ### 24049     Cr49      6478.58118515      1.78624116249
216 2.141E+01 ### 23049     V49       352451942.801      8.56835965333
217 2.141E+01 ### 22049     Ti49      704918128.027      0.0
218 2.141E+01 ### 27055     Co55      253567495453.0      2785060.44487
219 2.141E+01 ### 26055     Fe55      2.42542960147e+12    19393.6995083
220 2.141E+01 ### 25055     Mn55      215374258612.0      0.0
221 2.141E+01 ### 23047     V47       3448914.86405      1222.19100921
222 2.142E+01 ### 22047     Ti47      7084406.07395      0.0
223 2.142E+01 ### 25054     Mn54      18409932719.4      473.091548602
224 2.142E+01 ### 24054     Cr54      20828886.6638      0.0

```

225	2.142E+01	###	27057	Co57	5111669359.24	150.877816523
226	2.142E+01	###	27056	Co56	11199688888.9	1163.31661116
227	2.143E+01	###	27058	Co58	30667784.2188	3.47259601749
228	2.143E+01	###	23050	V50	0.0	0.0
229	2.143E+01	###	22050	Ti50	0.0	0.0
230	2.143E+01	###	25056	Mn56	1540918363.29	115050.418855
231	2.143E+01	###	27059	Co59	4029742.49245	0.0
232	2.143E+01	###	25057	Mn57	11013470.9442	89390.5893812
233	2.144E+01	###	1	Nn1	2.81319554815e+12	3172727892.23
234	2.144E+01	###	1001	H1	3.90904383849e+12	0.0
235	2.144E+01	###	2004	He4	265314778136.0	0.0
236	2.144E+01	###	1002	H2	126175920654.0	0.0
237	2.144E+01	###	2003	He3	11425023383.8	0.0
238	2.145E+01	###	1003	H3	2515358594.54	4.48083092714
239	2.145E+01	###				
240	3.152E+01	###				
241	3.152E+01	###				
242	3.152E+01	###	End of Sim Tally			
243	3.153E+01	###				
244	3.153E+01	###				
245	3.153E+01	###	26054	Fe54	1.63058555949e+20	0.0
246	3.153E+01	###	26056	Fe56	2.4682225152e+21	0.0
247	3.153E+01	###	26057	Fe57	5.60027794696e+19	0.0
248	3.153E+01	###	26058	Fe58	7.33461192328e+18	0.0
249	3.154E+01	###	27053	Co53	0.0	0.0
250	3.154E+01	###	26053	Fe53	4.53141271569e-143	6.15146092407e-146
251	3.154E+01	###	25053	Mn53	230790897436.0	0.00137750197927
252	3.155E+01	###	24053	Cr53	578858002.186	0.0
253	3.155E+01	###	25050	Mn50	0.0	0.0
254	3.156E+01	###	24050	Cr50	0.0	0.0
255	3.156E+01	###	26052	Fe52	93439.6978154	2.17465997465
256	3.157E+01	###	1025052	Mn52-M	4168.78150109	2.27168171686
257	3.157E+01	###	24052	Cr52	143724178764.0	0.0
258	3.157E+01	###	25052	Mn52	69033917523.9	98986.0848283
259	3.157E+01	###	25051	Mn51	7.09820416491e-20	1.7749279235e-23
260	3.157E+01	###	24051	Cr51	1124387120.91	325.59993095
261	3.158E+01	###	23051	V51	1168225.90545	0.0
262	3.158E+01	###	24049	Cr49	5.183538587e-28	1.42917866225e-31
263	3.158E+01	###	23049	V49	350240187.817	8.51459030248
264	3.158E+01	###	22049	Ti49	704918128.027	0.0
265	3.158E+01	###	27055	Co55	14632033320.7	160711.045225
266	3.158E+01	###	26055	Fe55	2.65898143279e+12	21261.1765249
267	3.159E+01	###	25055	Mn55	215374258612.0	0.0
268	3.159E+01	###	23047	V47	3.71227906462e-34	1.31551930833e-37
269	3.159E+01	###	22047	Ti47	7084406.07395	0.0
270	3.159E+01	###	25054	Mn54	18287479903.7	469.944802056
271	3.159E+01	###	24054	Cr54	20828886.6638	0.0
272	3.160E+01	###	27057	Co57	5072636183.79	149.725699695
273	3.160E+01	###	27056	Co56	10901613948.1	1132.35543595
274	3.160E+01	###	27058	Co58	29779081.9005	3.37196585428
275	3.160E+01	###	23050	V50	0.0	0.0
276	3.161E+01	###	22050	Ti50	0.0	0.0
277	3.161E+01	###	25056	Mn56	5.84462973621	0.000436380742305
278	3.161E+01	###	27059	Co59	4029742.49245	0.0
279	3.161E+01	###	25057	Mn57	0.0	0.0
280	3.161E+01	###	1	Nn1	1.77254842634e-115	1.99908386613e-118
281	3.161E+01	###	1001	H1	3.90904383849e+12	0.0
282	3.162E+01	###	2004	He4	265314778136.0	0.0
283	3.162E+01	###	1002	H2	126175920654.0	0.0
284	3.162E+01	###	2003	He3	11426186786.5	0.0
285	3.162E+01	###	1003	H3	2514195191.88	4.478758455

```
286 3.162E+01 ###
287 4.344E+01 ###
288 4.344E+01 ### Gamma Dose - Beam End
289 4.344E+01 ### Activity/Bq           82822873.7128
290 4.345E+01 ### Power eV/s          1.86841008076e+16
291 4.345E+01 ### Power J/s           0.00299352926319
292 4.345E+01 ### Dose Gy/s          2.97685885361e-06
293 4.345E+01 ### Dose Gy/hr          0.010716691873
294 4.345E+01 ### Percentage of annual dose/hr 9394.25237015
295 4.345E+01 ###
296 4.346E+01 ###
297 4.346E+01 ### Gamma Dose - Sim End
298 4.346E+01 ### Activity/Bq           283043.751191
299 4.346E+01 ### Power eV/s          5.1310279645e+14
300 4.346E+01 ### Power J/s           8.22083038417e-05
301 4.347E+01 ### Dose Gy/s          1.96294899335e-07
302 4.347E+01 ### Dose Gy/hr          0.000294301803729
303 4.347E+01 ### Percentage of annual dose/hr 257.984968681
304 4.347E+01 ###
```

Appendix I

Neutron Activity Code

I.1 Full Source Code

The full source code is available to download from github.

https://github.com/BenPalmer1983/neutron_activation

I.2 Implementation of the Activity Equations in Python

Listing I.1: Decay Equation Code in Python3

```
1 #####  
2 # DECAY EQUATIONS  
3 #####  
4  
5 @staticmethod  
6 def activity(t, l, b, w, n0):  
7     nt = numpy.zeros((len(n0),))  
8     for m in range(0,len(n0)):  
9         if(l[m] > 0.0):  
10             nt[m] = isotopes.activity_unstable(t, l, b, w, n0, m)  
11         elif(l[m] == 0.0):  
12             nt[m] = isotopes.activity_stable(t, l, b, w, n0, m)  
13     return nt  
14  
15 @staticmethod  
16 def activity_unstable(t, l, b, w, n0, m):  
17     s = 0.0  
18     for k in range(0, m+1):  
19         s = s + isotopes.r(k, m, b, l) * ( isotopes.f_unstable(t,k,m,l) * n0[k] + isotopes.g_unstable(t,k,m,l) * w[k] )  
20     return s  
21  
22 @staticmethod  
23 def activity_stable(t, l, b, w, n0, m):  
24     s = n0[m] + w[m] * t  
25     for k in range(0, m):  
26         s = s + isotopes.r(k, m, b, l) * (isotopes.f_stable(t,k,m,l) * n0[k] + isotopes.g_stable(t,k,m,l) * w[k])  
27     return s  
28  
29 @staticmethod  
30 def r(k, m, b, l):
```

```

31     if(k == m):
32         return 1.0
33     else:
34         p = 1.0
35         for i in range(k, m):
36             p = p * (b[i] * l[i])
37         return p
38
39 @staticmethod
40 def f_unstable(t,k,m,l):
41     s = 0.0
42     for i in range(k, m+1):
43         p = 1.0
44         for j in range(k, m+1):
45             if(i != j):
46                 p = p * (1 / (l[i] - l[j]))
47             s = s + numpy.exp(-1 * l[i] * t) * p
48     s = (-1)**(m-k) * s
49     return s
50
51 @staticmethod
52 def g_unstable(t,k,m,l):
53     pa = 1.0
54     for i in range(k,m+1):
55         pa = pa * l[i]
56     pa = 1.0 / pa
57     s = 0.0
58     for i in range(k, m+1):
59         pb = 1.0
60         for j in range(k, m+1):
61             if(i != j):
62                 pb = pb * (1 / (l[i]-l[j]))
63             s = s + (1/l[i]) * numpy.exp(-1*i*t) * pb
64     return pa + s * (-1)**(m-k+1)
65
66
67 @staticmethod
68 def f_stable(t,k,m_in,l):
69     m = m_in - 1
70
71     p = 1.0
72     for i in range(k, m+1):
73         p = p * l[i]
74
75     s = 0.0
76     for i in range(k, m+1):
77         r = l[i]
78         for j in range(k, m+1):
79             if(i != j):
80                 r = r * (l[i] - l[j])
81             s = s + (1/r)*numpy.exp(-1*l[i]*t)
82
83     return (1.0/p) + s * (-1.0)**(m-k+1)
84
85
86 @staticmethod
87 def g_stable(t,k,m_in,l):
88     m = m_in - 1
89
90     pa = 1.0
91     for i in range(k,m+1):

```

```

92     pa = pa * l[i]
93     pa = 1.0 / pa
94
95     sa = 0.0
96     for i in range(k, m+1):
97         pb = 1.0
98         for j in range(k,m+1):
99             if(j != i):
100                 pb = pb * l[j]
101             sa = sa + pb
102         pc = 1.0
103         for i in range(k, m+1):
104             pc = pc * l[i]**2
105
106         sb = 0.0
107         for i in range(k, m+1):
108             pd = 1.0
109             for j in range(k, m+1):
110                 if(i != j):
111                     pd = pd * (1 / (l[i]-l[j]))
112             sb = sb + (1/(l[i]**2)) * numpy.exp(-l[i]*t) * pd
113
114     return pa * t + sa / pc + sb * (-1)**(m-k+1)

```

Appendix J

PGM Activity after Neutron Irradiation

The neutron activity code was used to roughly approximate activities and compare that of 304SS doped with 1% of the following: Mn, Mo, Pd, Pt and Ru; against an plain sample (Fe 19 Cr 10 Ni). The simulation was 1MeV neutrons in a Maxwell-Bolztmann distribution for 10 hours of irradiation with 1 hour of cooling. The activities are relative to one another. Mn as an additive is known to be a problem element due to the creation of ^{56}Mn and has been included for comparison.

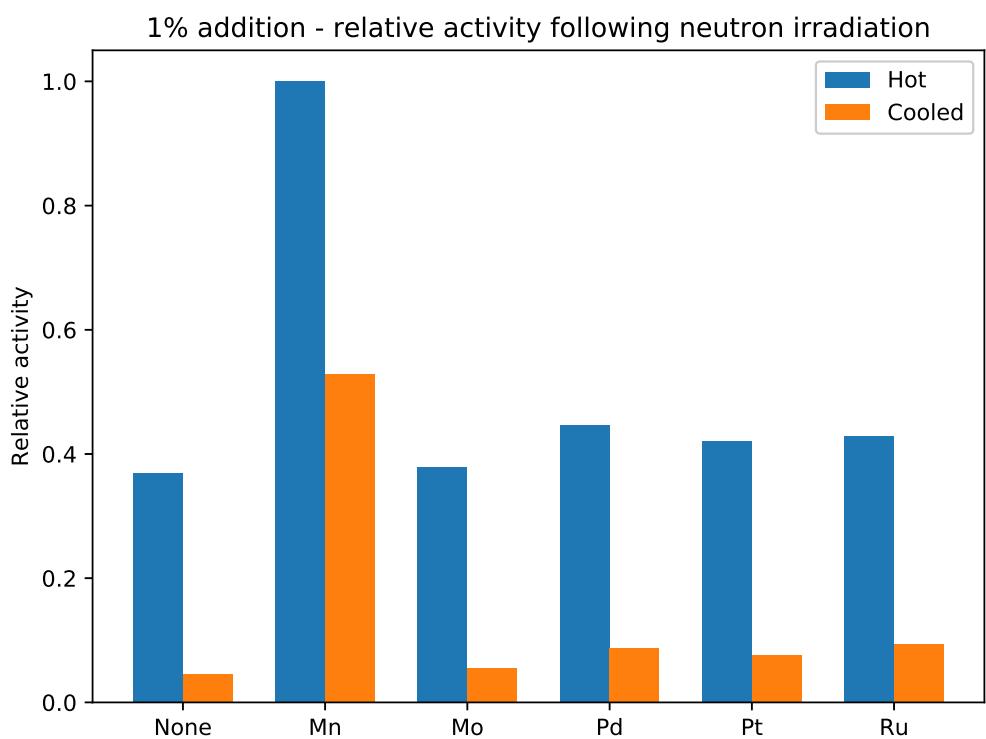


Figure J.1: Relative activity hot (just after irradiation) and cooled (1 hour of cooling) for 1MeV neutron irradiated SS304 and SS304 with 1% addition of Mn, Mo, Pd, Pt and Ru

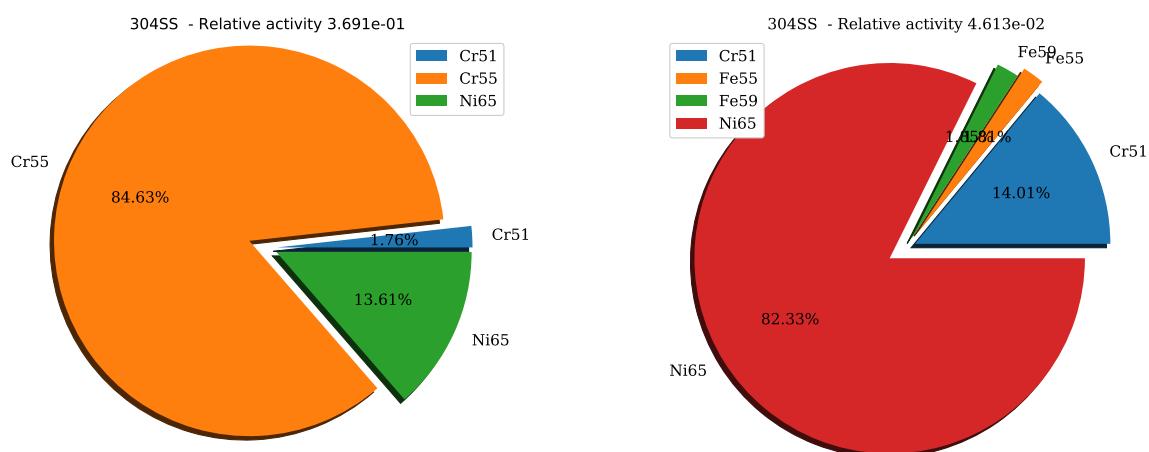


Figure J.2: Primary isotopes contributing to activity for plain SS304. Left: hot, right: cooled.

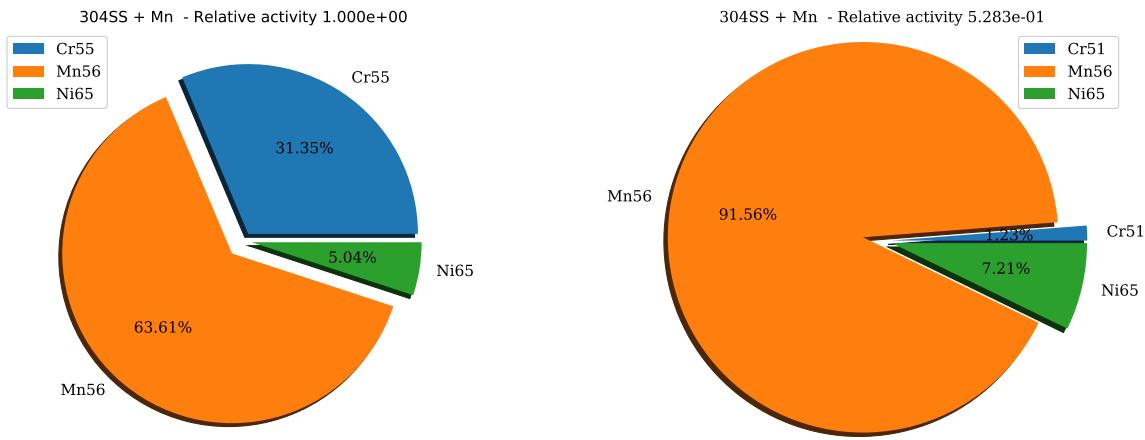


Figure J.3: Primary isotopes contributing to activity for SS304 + 1%Mn. Left: hot, right: cooled.

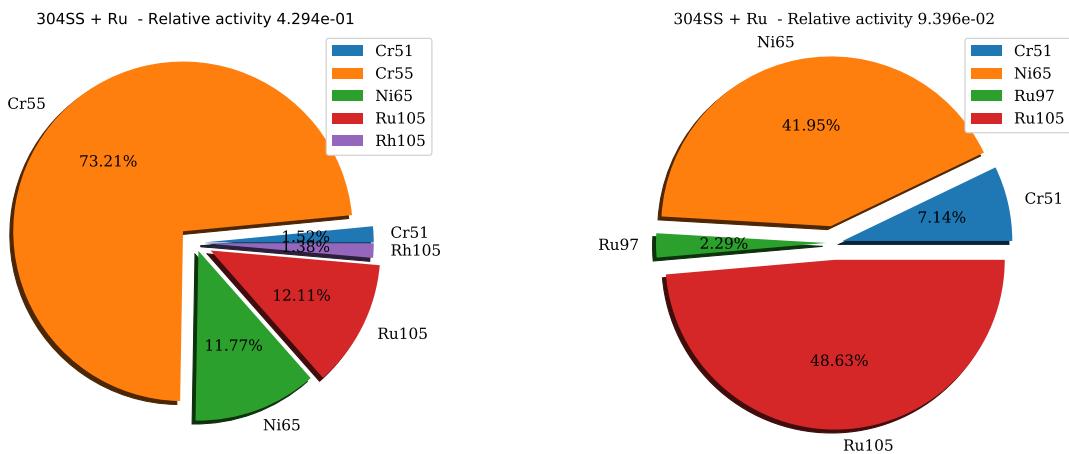


Figure J.4: Primary isotopes contributing to activity for SS304 + 1%Ru. Left: hot, right: cooled.

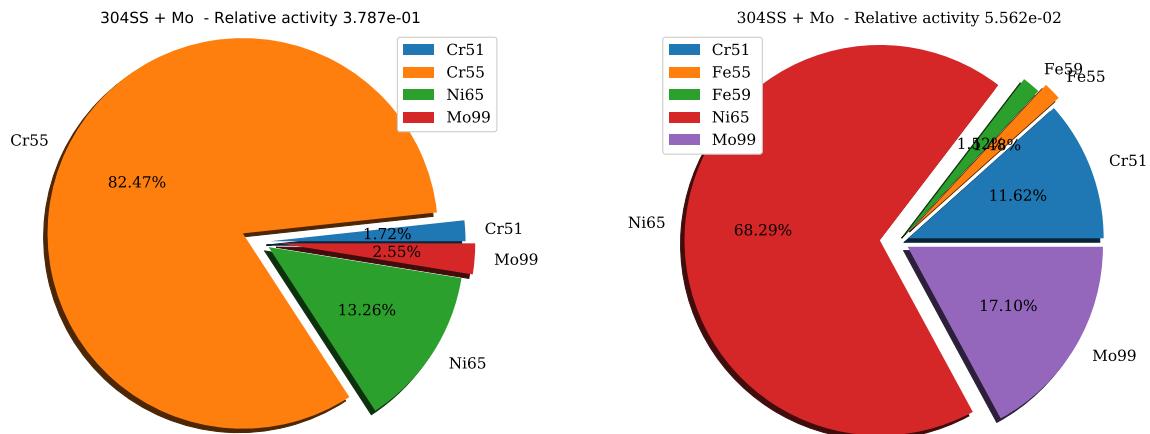


Figure J.5: Primary isotopes contributing to activity for SS304 + 1%Mo. Left: hot, right: cooled.

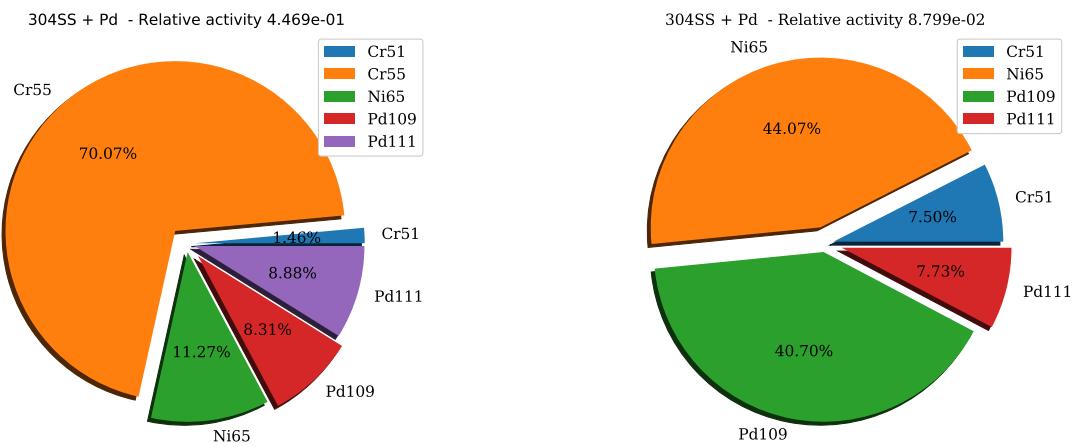


Figure J.6: Primary isotopes contributing to activity for SS304 + 1%Pd. Left: hot, right: cooled.

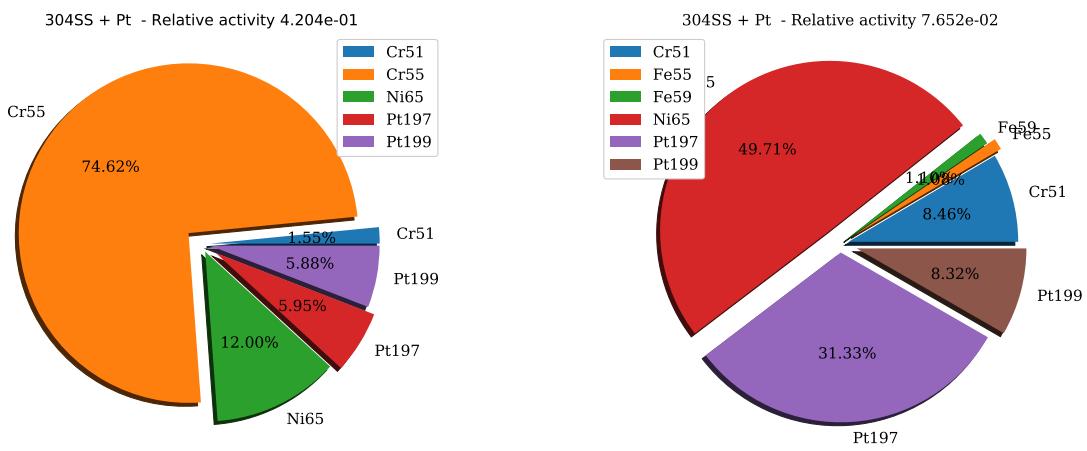


Figure J.7: Primary isotopes contributing to activity for SS304 + 1%Pd. Left: hot, right: cooled.

Appendix K

Iron Activity: Irradiated to 100DPA

K.1 Activity vs Beam Energy

The beam energy was set to 5MeV, 10MeV, 15MeV, 20MeV and 25MeV within SRIM, and the Activity V2 code was used to calculate how radioactive each sample would be if irradiated to 100 DPA of damage.

The beam settings are:

- target material = pure iron
- target thickness = 0.5mm
- beam current = 60 micro amps (maximum proton current for the Scanditronix MC-40)
- beam area = 64mm^2

K.1.1 5MeV Proton Beam - 60 microamp 100DPA

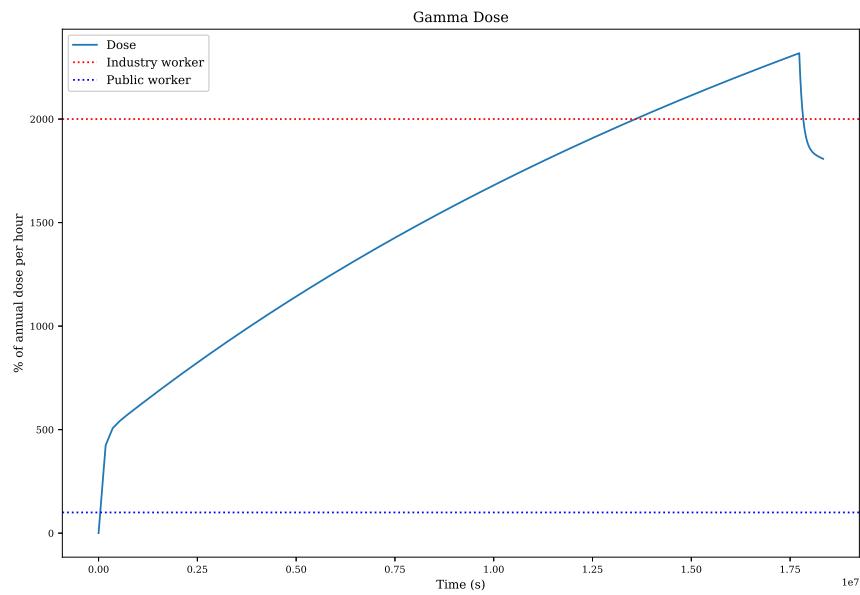


Figure K.1: Percentage of Annual Dose Equivalent

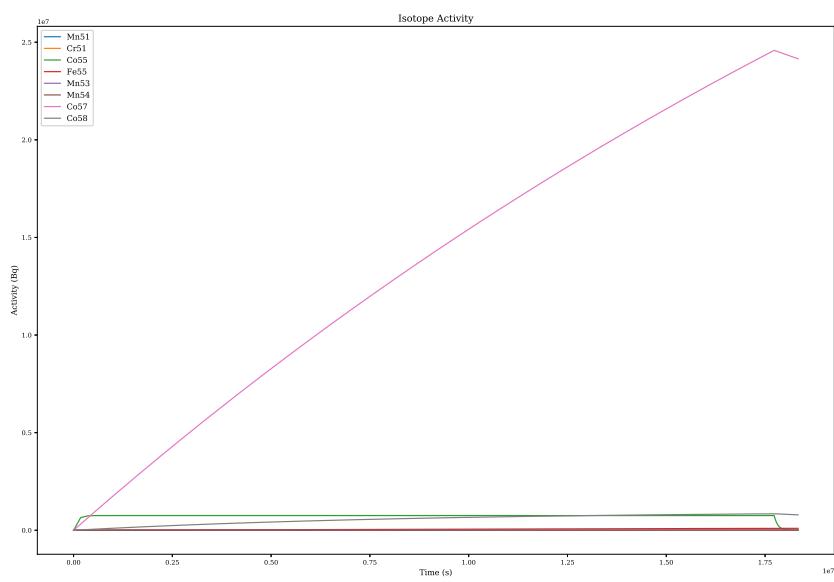


Figure K.2: Activity of all isotopes over time

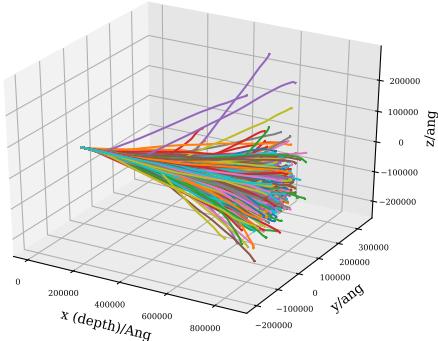


Figure K.3: Ion trajectory in the target

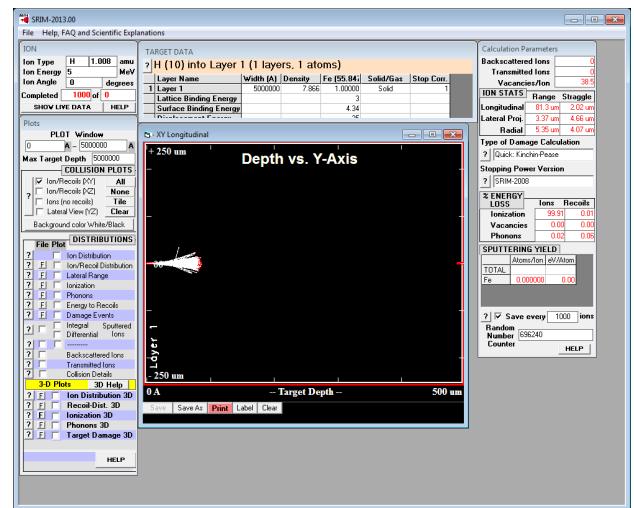


Figure K.4: SRIM output summary

K.1.2 10MeV Proton Beam - 60 microamp 100DPA

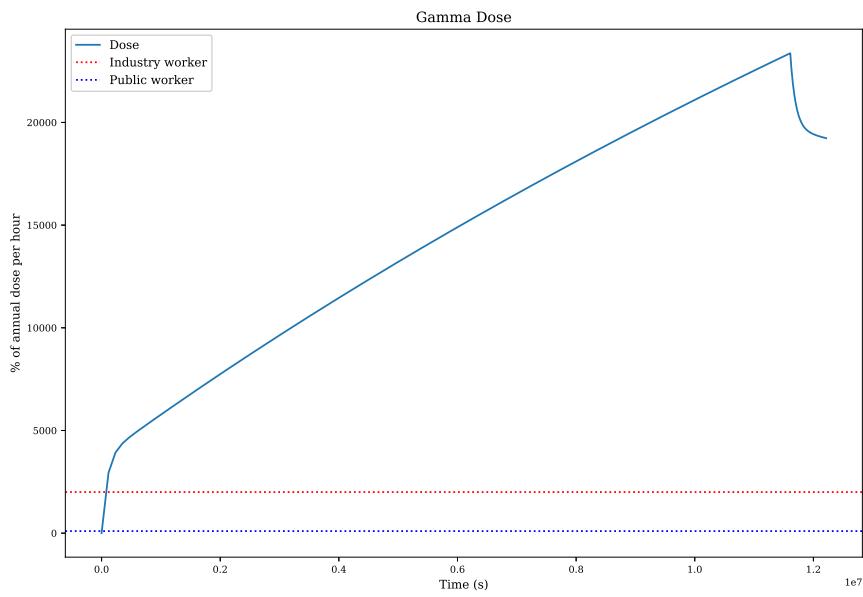


Figure K.5: Percentage of Annual Dose Equivalent

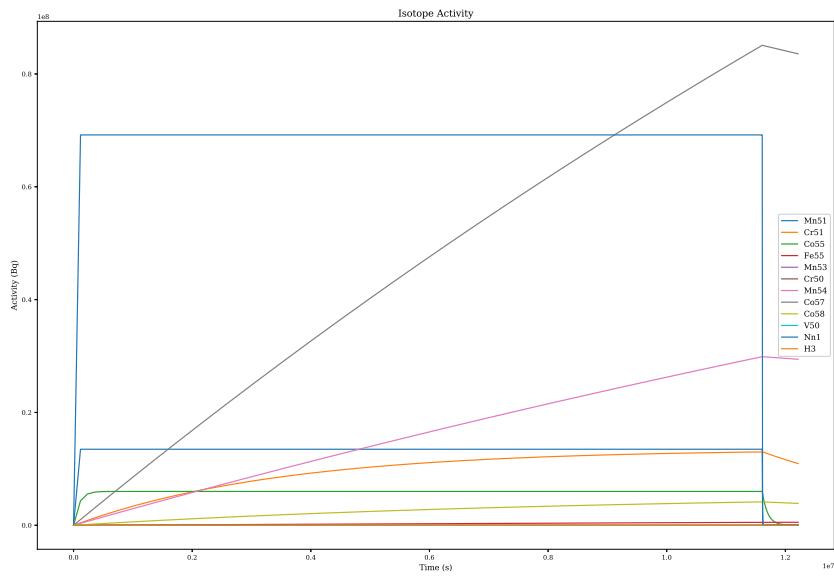


Figure K.6: Activity of all isotopes over time

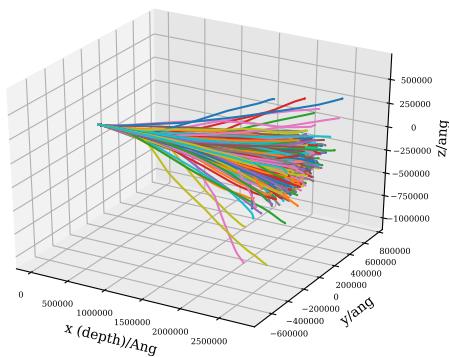


Figure K.7: Ion trajectory in the target

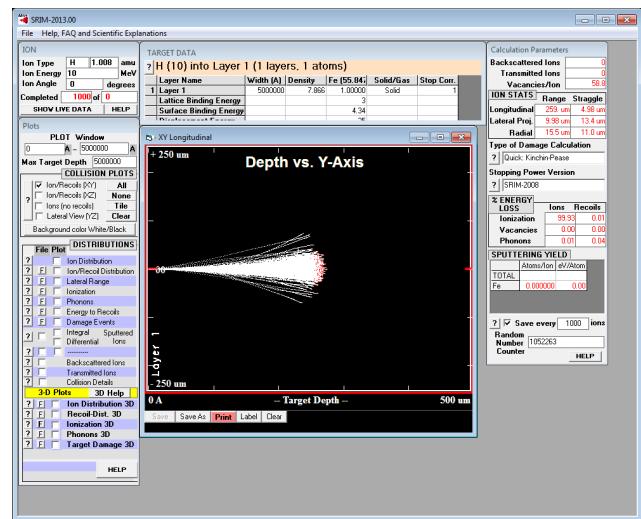


Figure K.8: SRIM output summary

K.1.3 15MeV Proton Beam - 60 microamp 100DPA

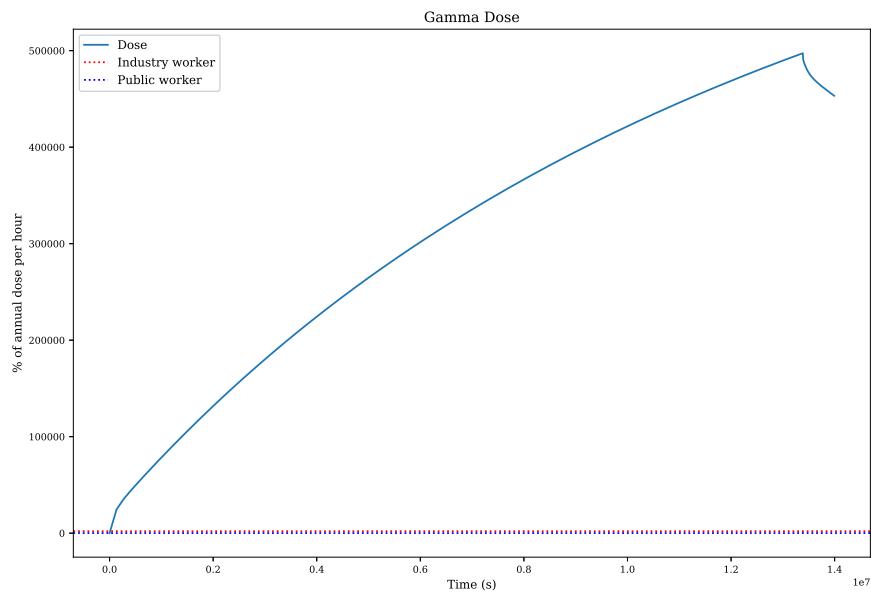


Figure K.9: Percentage of Annual Dose Equivalent

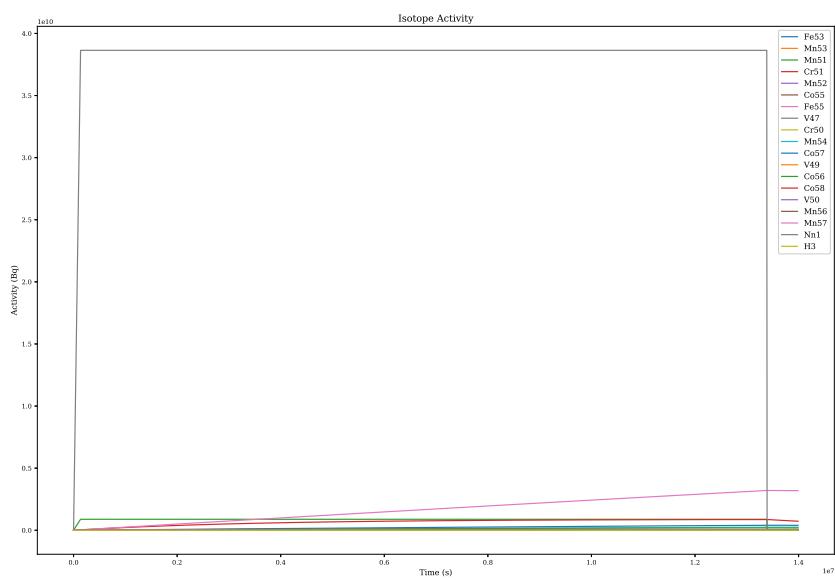


Figure K.10: Activity of all isotopes over time

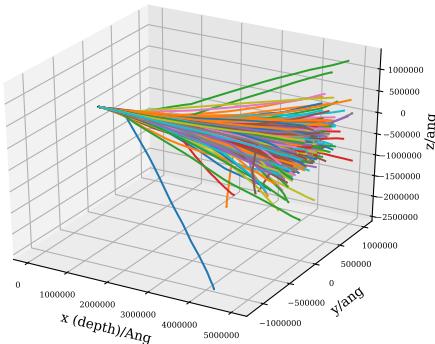


Figure K.11: Ion trajectory in the target

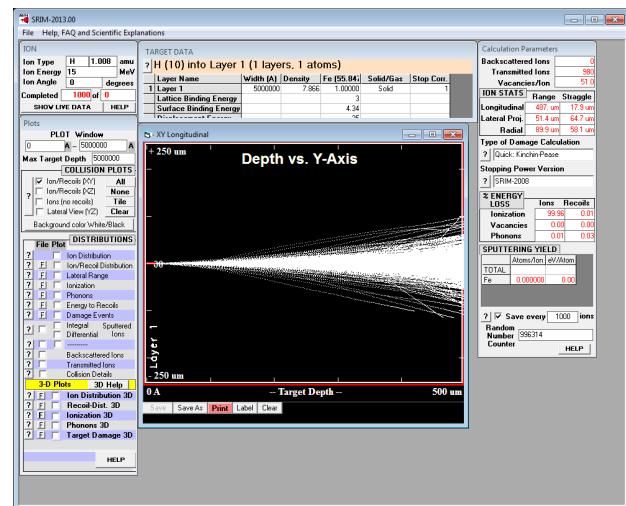


Figure K.12: SRIM output summary

K.1.4 20MeV Proton Beam - 60 microamp 100DPA

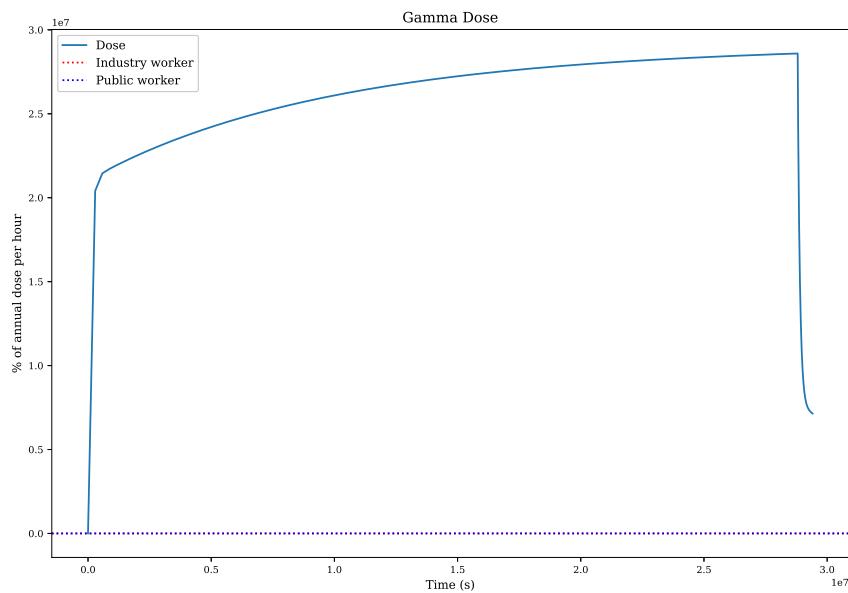


Figure K.13: Percentage of Annual Dose Equivalent

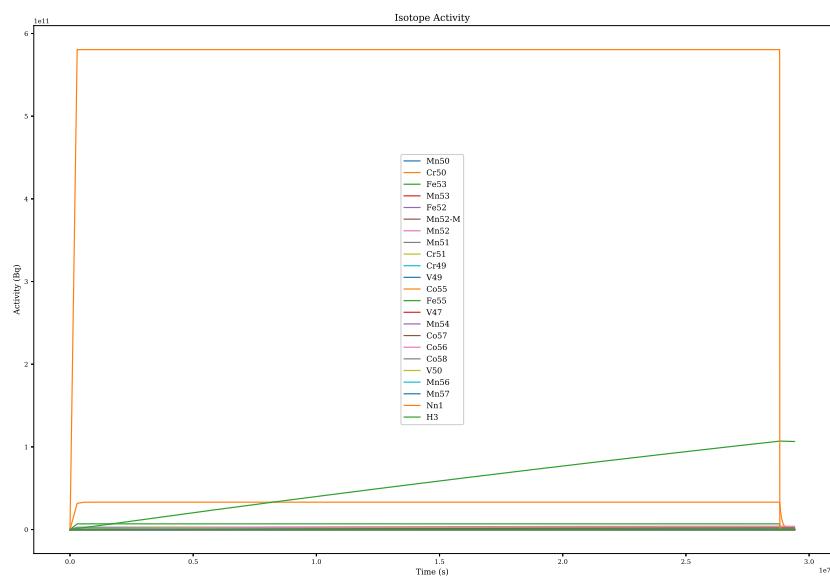


Figure K.14: Activity of all isotopes over time

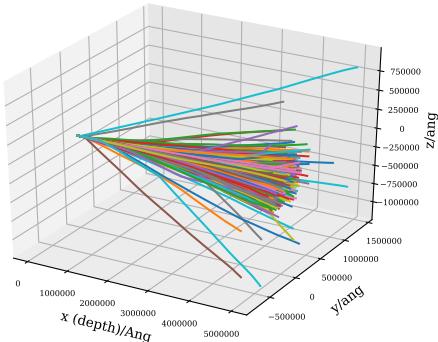


Figure K.15: Ion trajectory in the target

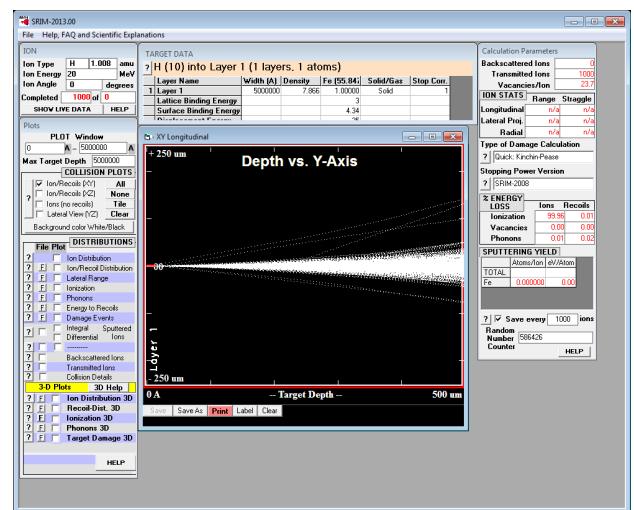


Figure K.16: SRIM output summary

K.1.5 25MeV Proton Beam - 60 microamp 100DPA

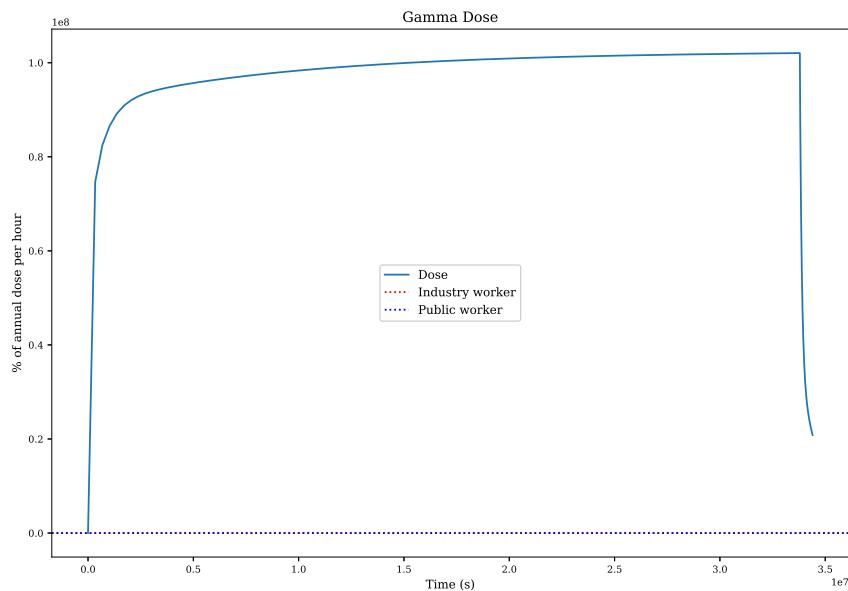


Figure K.17: Percentage of Annual Dose Equivalent

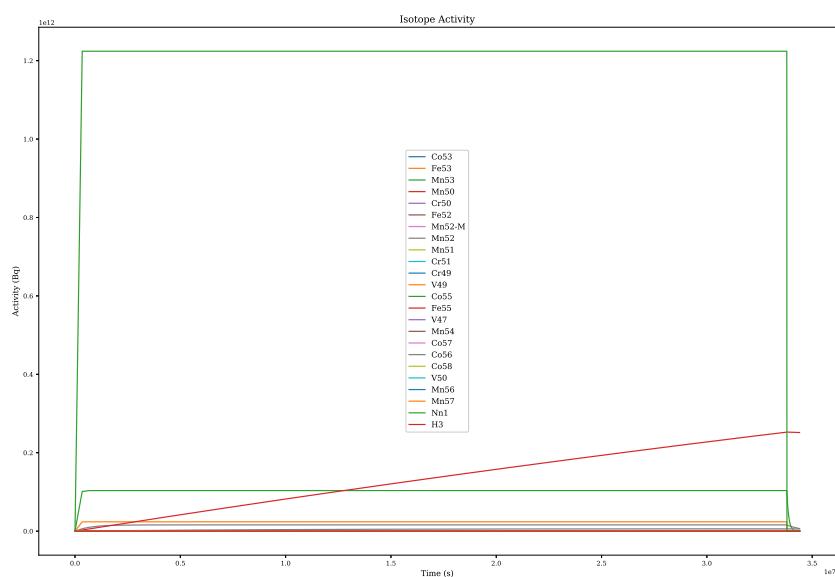


Figure K.18: Activity of all isotopes over time

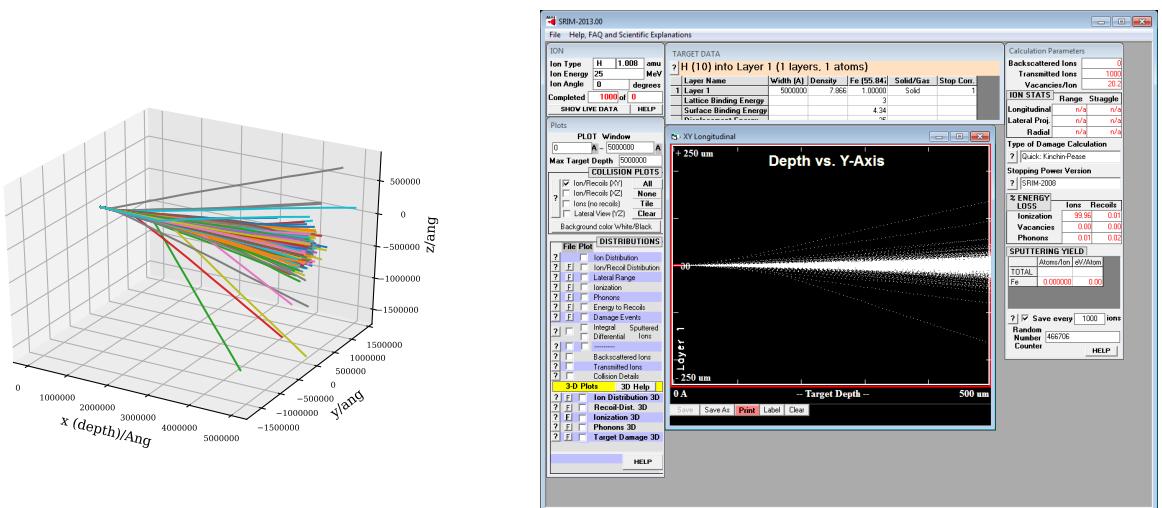


Figure K.19: Ion trajectory in the target

Figure K.20: SRIM output summary

K.1.6 Results Summary

	Background	5MeV	10MeV	15MeV	20MeV	25MeV
VPI	-	38.5	58.8	51.0	23.7	20.2
DPA/year	-	178	272	236	109	93
Days for 100 DPA	-	205	134	155	333	391
Damage Depth (mm)	-	Under 0.1	0.25	0.5+	0.5+	0.5+
Dose (Gy/hr)	1.14×10^{-7}	2.64×10^{-3}	2.67×10^{-2}	5.67×10^{-1}	3.26×10^1	1.16×10^2
Dose (Gy/hr) (1 week of cooling)	1.14×10^{-7}	2.04×10^{-3}	2.19×10^{-2}	5.17×10^{-1}	8.15×10^0	2.38×10^1

Table K.1: Comparison of proton irradiated iron up to 100DPA at a range of proton energies. The background dose is given in sieverts, as the dose equivalent, and is the limit for members of the public per hour, when averaged over a year. The damage depth is the approximate maximum depth ions will reach.

Appendix L

PBE Functional

The GGA PBE functional was the primary type used in this work for the DFT calculated data. The LDA, LSDA and GGA are described in detail in “Density functionals from LDA to GGA”[116].

The local density approximation is:

$$E_{xc}^{LDA}[\rho] = \int d^3r \rho(\vec{r}) [e_x(\rho(\vec{r})) + e_c(\rho(\vec{r}))] \quad (\text{L.1})$$

Exchange energy per electron of the unpolarized uniform electron gas:

$$e_x(\rho) \quad (\text{L.2})$$

Correlation energy per electron of the unpolarized uniform electron gas:

$$e_c(\rho) \quad (\text{L.3})$$

Splitting the density into spin up and spin down:

$$\rho_{\uparrow}(\vec{r}) = \sum_k^{occ} |\phi_{k,\uparrow}(\vec{r})|^2 \quad \rho_{\downarrow}(\vec{r}) = \sum_k^{occ} |\phi_{k,\downarrow}(\vec{r})|^2 \quad \rho(\vec{r}) = \rho_{\uparrow}(\vec{r}) + \rho_{\downarrow}(\vec{r}) \quad (\text{L.4})$$

The relative spin polarization is:

$$\zeta = \frac{\rho_{\uparrow} - \rho_{\downarrow}}{\rho_{\uparrow} + \rho_{\downarrow}} \quad (\text{L.5})$$

The LSDA is:

$$E_{xc}^{LSDA}[\rho_{\uparrow}, \rho_{\downarrow}] = \int d^3r \rho(\vec{r}) [e_x(\rho(\vec{r})) f(\zeta(\vec{r})) + e_c(r_s(\vec{r}), \zeta(\vec{r}))] \quad (\text{L.6})$$

where

$$f(\zeta) = \frac{1}{2} \left[(1 + \zeta)^{\frac{4}{3}} + (1 - \zeta)^{\frac{4}{3}} \right] \quad (\text{L.7})$$

The correlation energy per electron of the spin-polarized uniform electron gas is:

$$e_c(r_s, \zeta) \quad (\text{L.8})$$

where r_s is the local Seitz radius:

$$\begin{aligned} \rho &= 3/4\pi r_s^3 \\ r_s &= \sqrt[3]{\frac{4\rho}{3\pi}} \end{aligned} \quad (\text{L.9})$$

The PBE correlation energy functional is:

$$E_c^{PBE}[\rho_\uparrow, \rho_\downarrow] = \int d^3r \rho [e_c(r_s, \zeta) + H(r_s, \zeta, t)] \quad (\text{L.10})$$

where

$$t = \left(\frac{\pi}{4}\right)^{\frac{1}{2}} \left(\frac{9\pi}{4}\right)^{\frac{1}{6}} \frac{s}{\phi r_s^{\frac{1}{2}}} \quad (\text{L.11})$$

$$s = \frac{|\nabla \rho|}{2(3\pi^2)^{\frac{1}{3}} \rho^{\frac{4}{3}}} = \frac{3}{2} \left(\frac{4}{9\pi}\right)^{\frac{1}{3}} a_B |\nabla r_s| \quad (\text{L.12})$$

$$a_B = \frac{\hbar^2}{m\epsilon^2} \text{ (the Bohr radius)} \quad (\text{L.13})$$

$$\phi(\zeta) = \frac{1}{2} \left[(1 + \zeta)^{\frac{2}{3}} + (1 - \zeta)^{\frac{2}{3}} \right] \quad (\text{L.14})$$

$$H = \gamma \phi^3 \ln \left[1 + \frac{\beta}{\gamma} t^2 \left(\frac{1 + At^2}{1 + At^2 + A^2 t^4} \right) \right] \quad (\text{L.15})$$

$$A = \left(\frac{\beta}{\gamma}\right) \frac{1}{\exp(-e_c(r_s, \zeta)/(\gamma \phi^3)) - 1} \quad (\text{L.16})$$

where $\beta = 0.066725$ and $\gamma = 0.031091$ Hartree.

The PBE exchange energy functional is:

$$E_x^{PBE}[\rho] = \int d^3r \rho e_x(\rho) F_x(s) \quad (\text{L.17})$$

where

$$F_x(s) = 1 + \kappa - \frac{\kappa}{1 + \mu s^2 / \kappa} \quad (\text{L.18})$$

where $\kappa = 0.804$ and $\mu = 0.21951$.

The spin density functional is:

$$E_x^{PBE}[\rho_\uparrow, \rho_\downarrow] = \frac{1}{2} E_x^{PBE}[2\rho_\uparrow] + \frac{1}{2} E_x^{PBE}[2\rho_\downarrow] \quad (\text{L.19})$$

Appendix M

DFT Convergence Plots

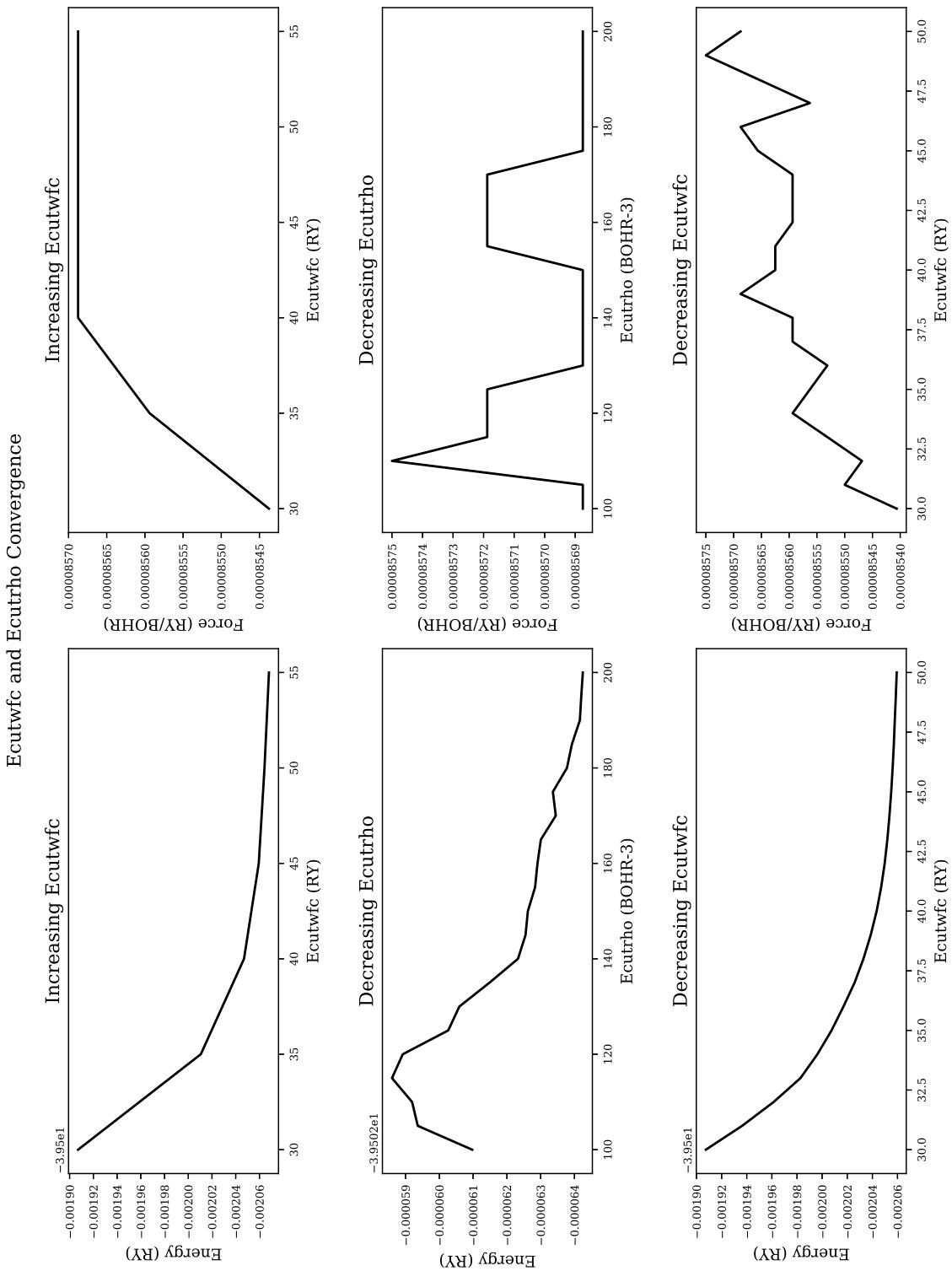


Figure M.1: Ecutfc and Ecutrho convergence for aluminium (energy and force)

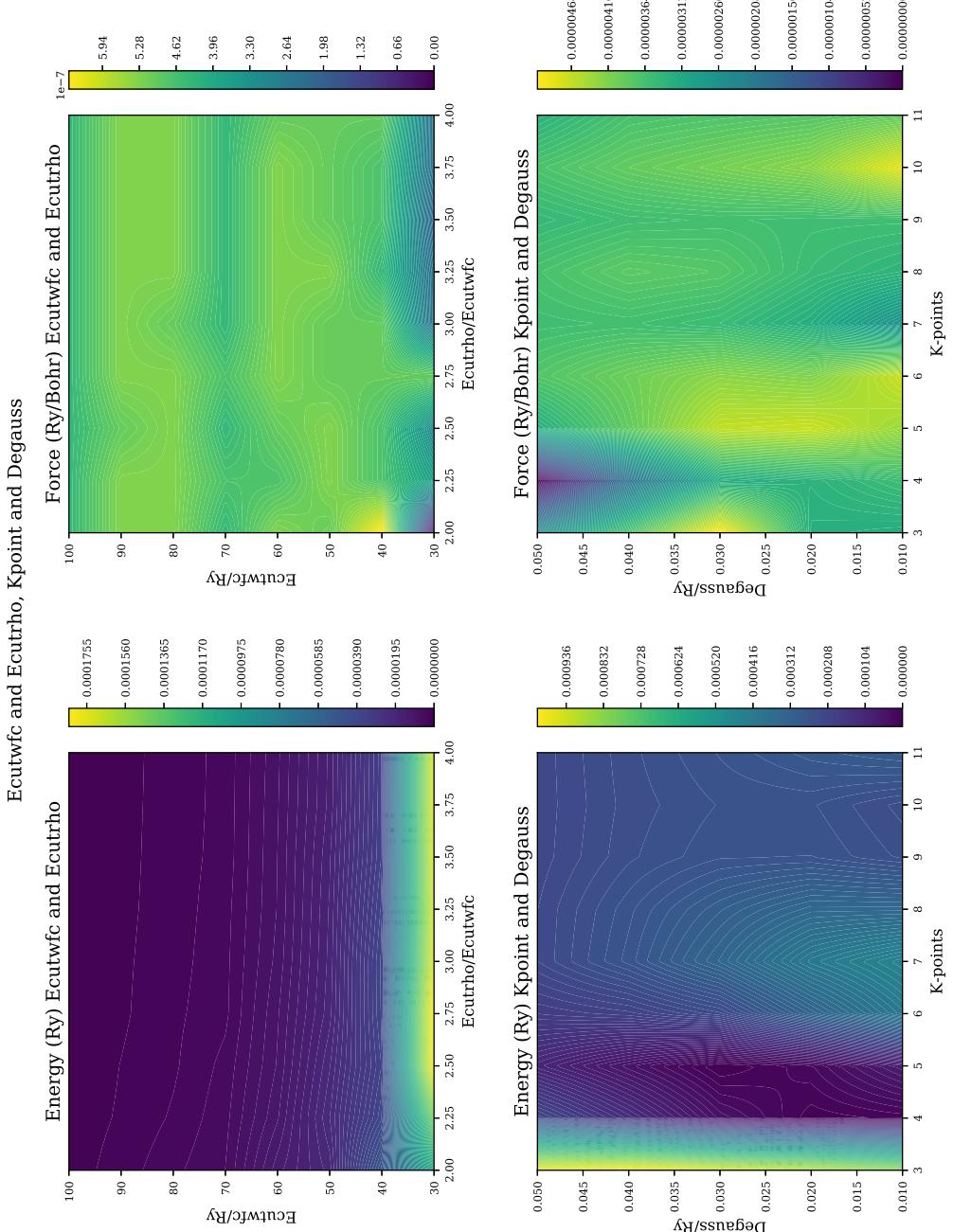


Figure M.2: 2D density map - force and energy convergence with k-points and smearing - alluminium

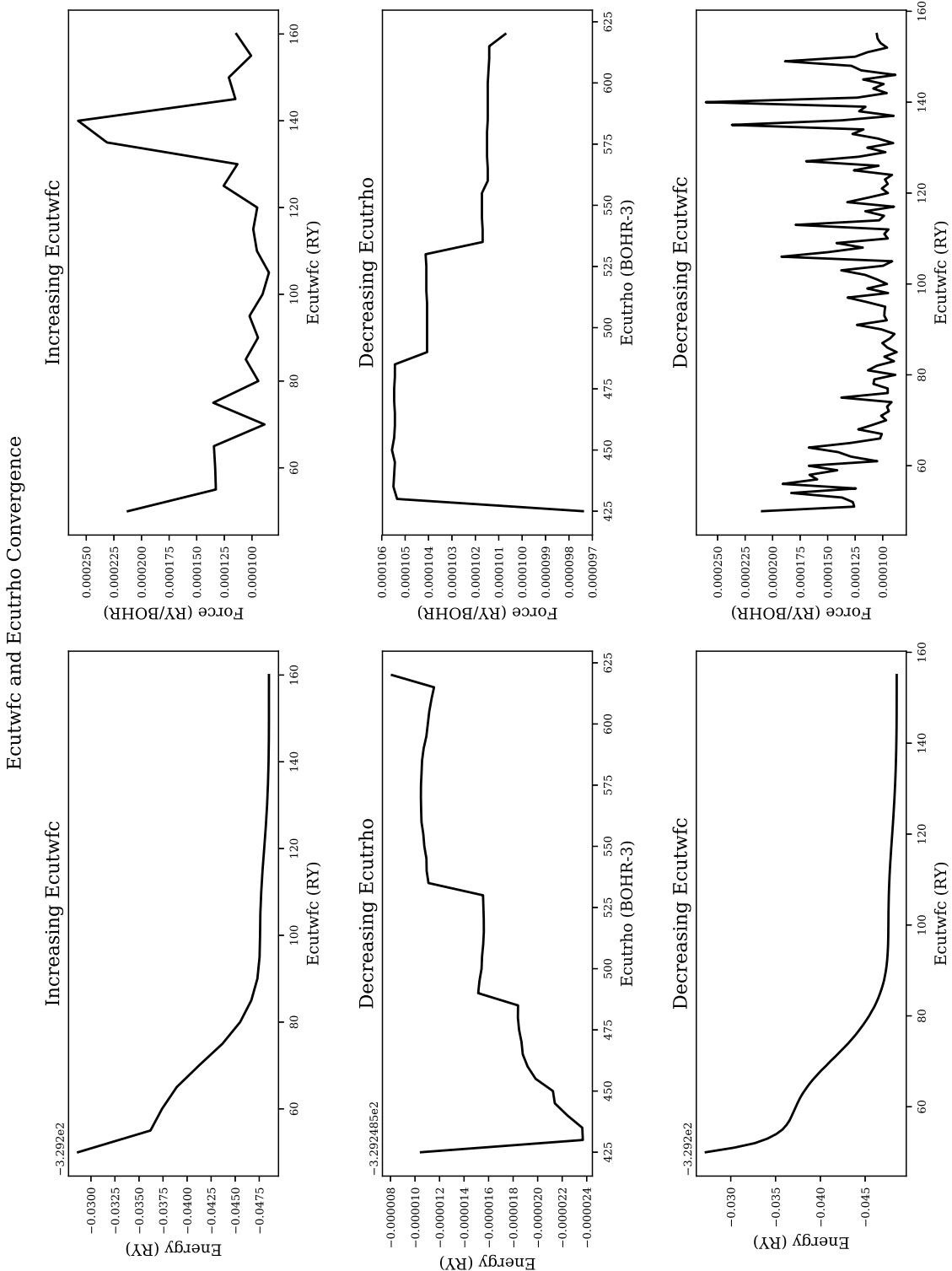


Figure M.3: E_{cutwfc} and E_{cutrfo} convergence for iron (energy and force)

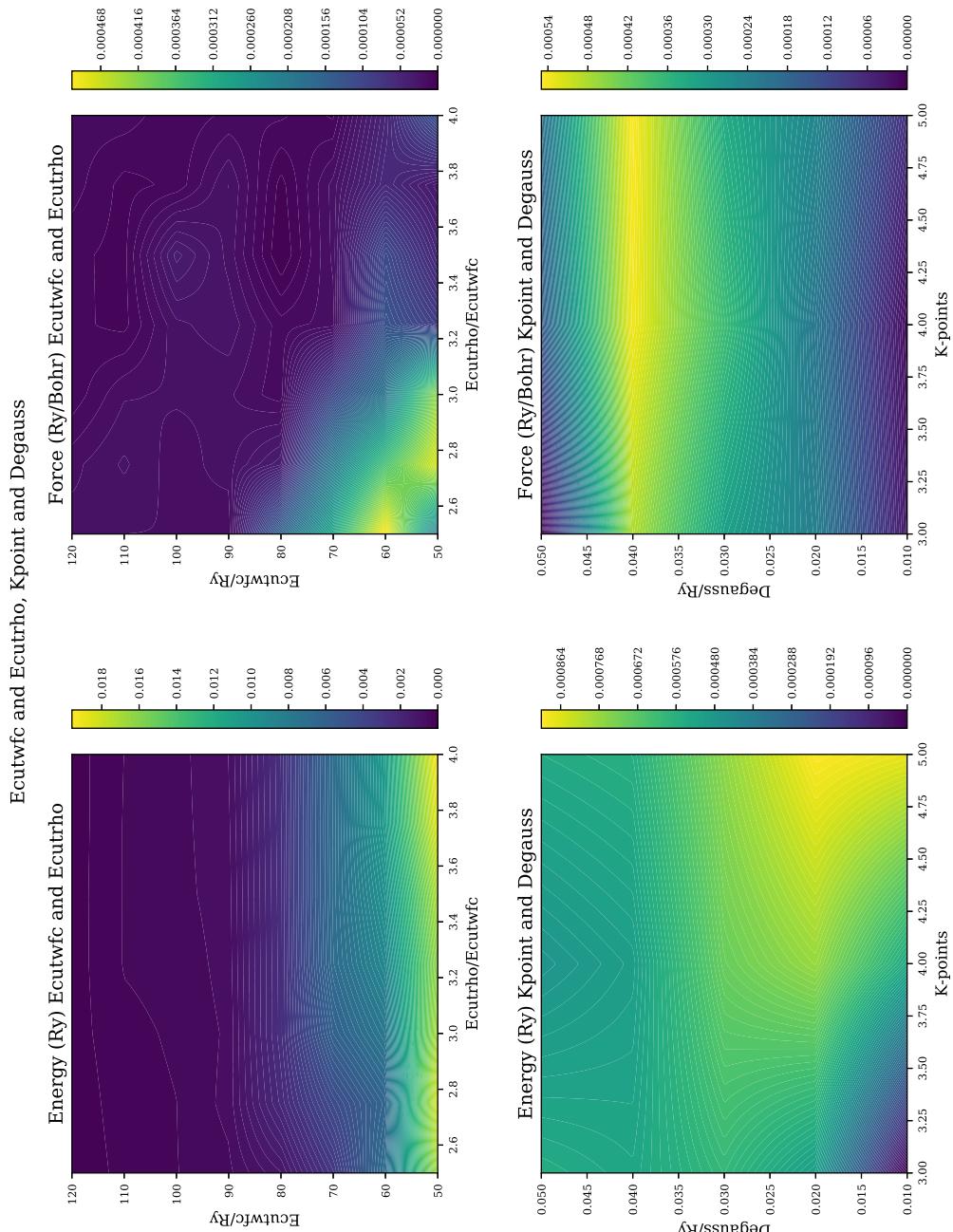


Figure M.4: 2D density map - force and energy convergence with k-points and smearing - iron

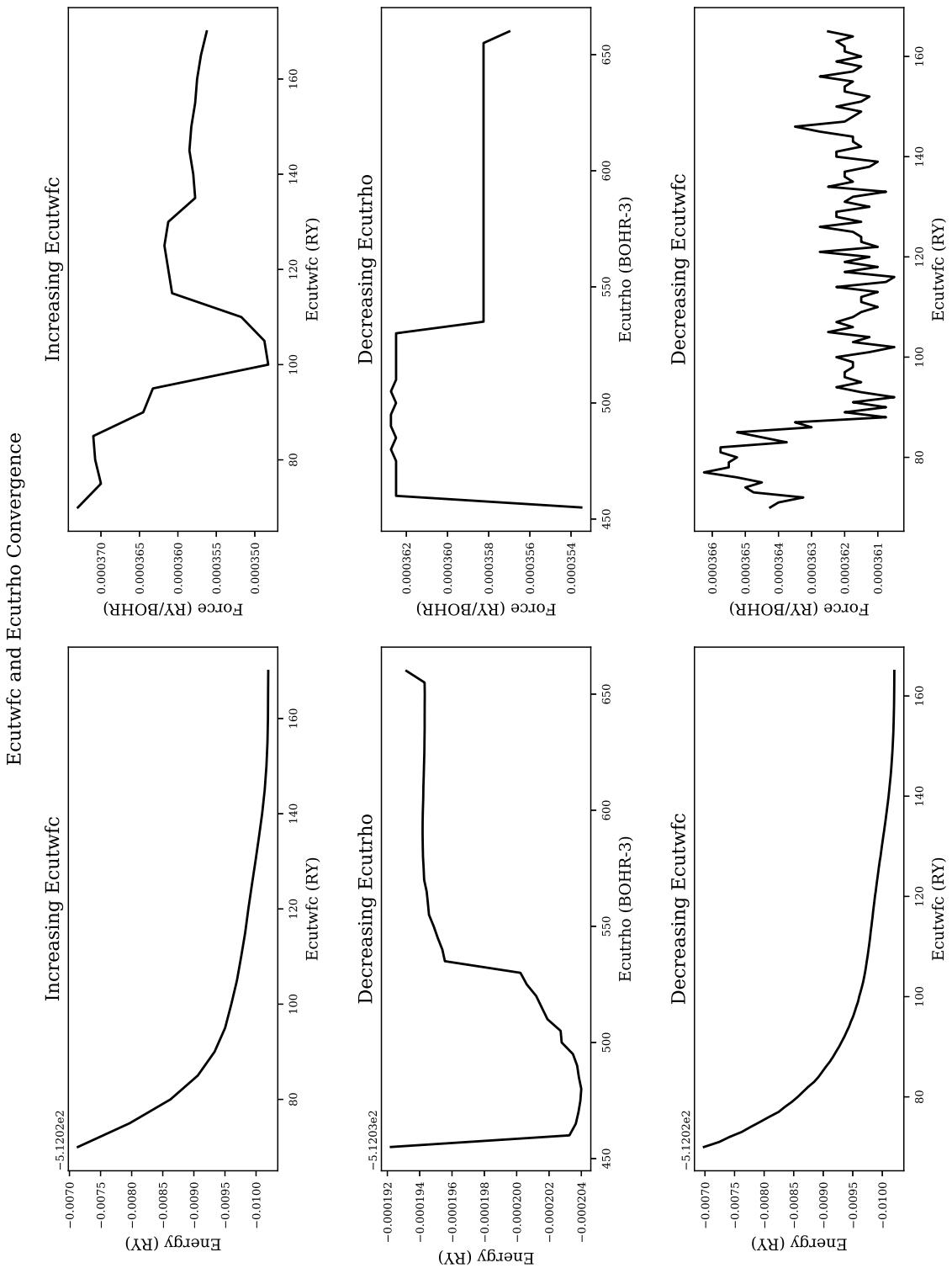


Figure M.5: Ecutwfc and Ecutrho convergence for palladium (energy and force)

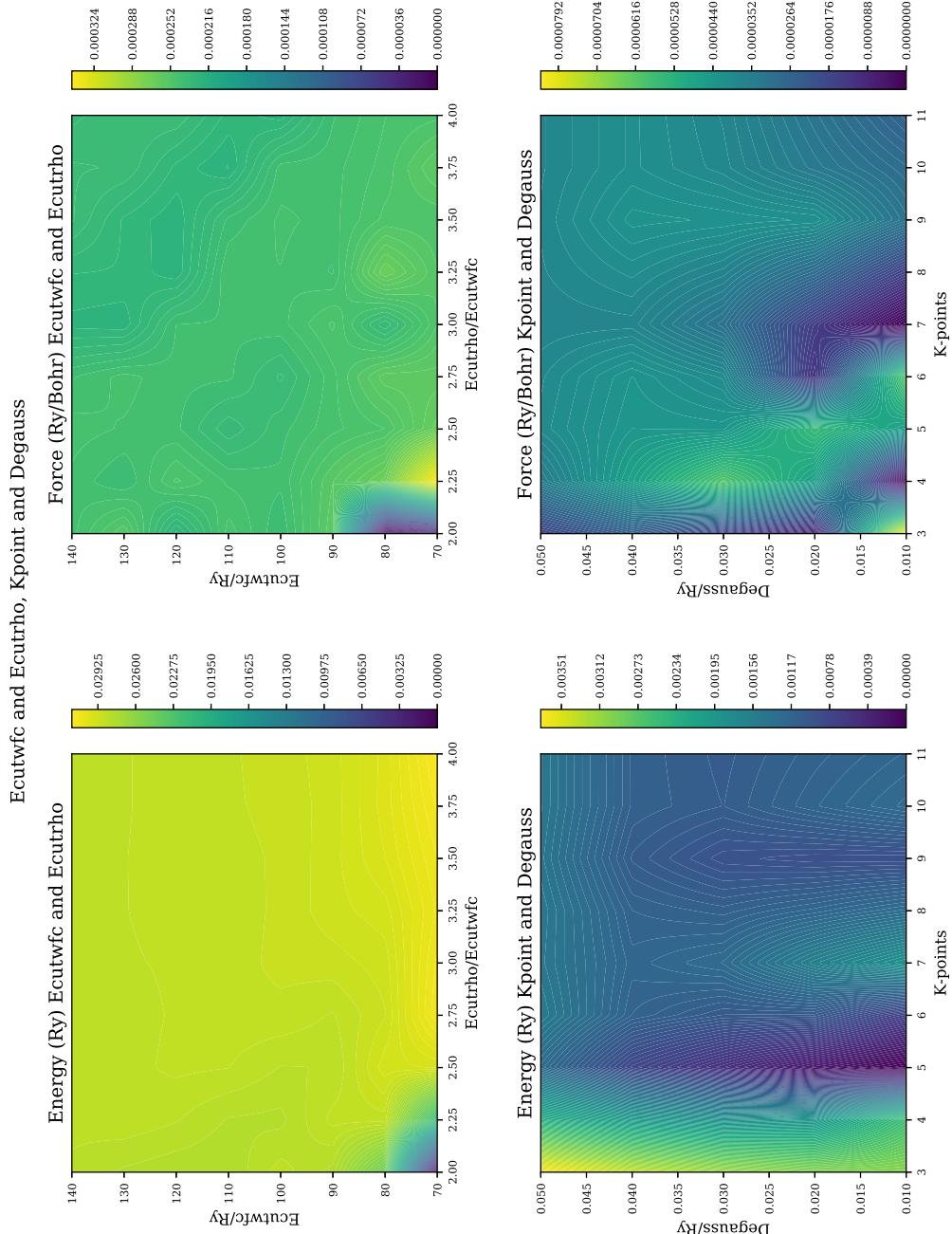


Figure M.6: 2D density map - force and energy convergence with k-points and smearing - palladium

Appendix N

QEEOS

Appendix O

DFT Calculated Properties

O.1 QEEOS

These results were calculated using the QEEOS python program that automatically creates and runs PWscf input files, collects the output files and processes the data.

O.2 FCC Aluminium

O.2.1 DFT Settings for PWscf

The calculations were performed with the Quantum Espresso PWscf package along with the QEEOS python program.

Ecutwfc: 50

Ecutrho: 200

Smearing: 0.04

K-points: 11 11 11 1 1 1

Nspin: 1 (non-polarized calculation)

Pseudopotential: Al.pbe-nl-kjpaw-psl.1.0.0.UPF

O.2.2 Equation of State and Elastic Constants

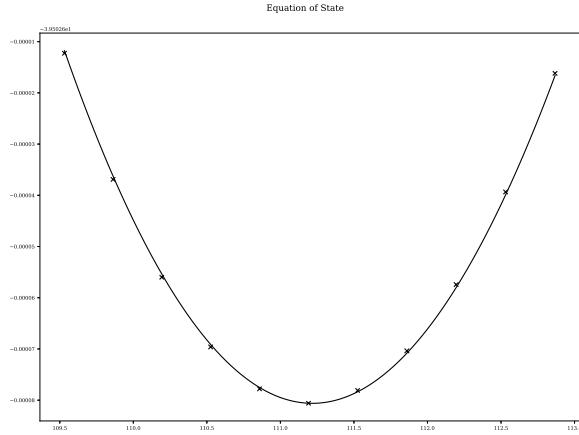


Figure O.1: Aluminium FCC equation of state plot

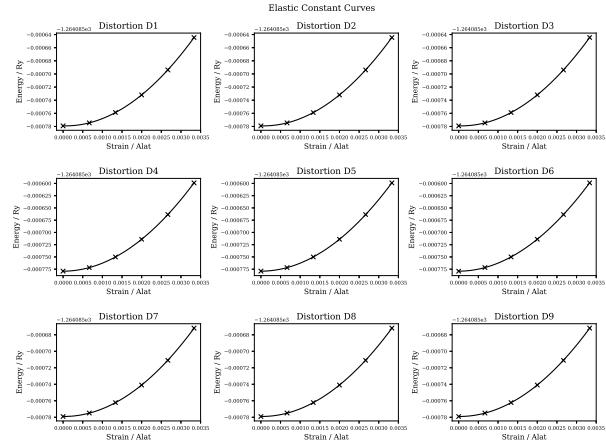


Figure O.2: Aluminium FCC elastic constants plot

O.2.3 Aluminium FCC Results File

Listing O.1: Aluminium FCC DFT Calculated Properties

```

1 ######
2 #          RESULTS          #
3 #####
4 Thu, 26 Mar 2020 13:32:45 +0000
5
6 INPUT SETTINGS
7
8 Structure:           fcc
9 Size:                2
10 Alat (Bohr):        6.9
11 Alat (Bohr) (exp.): 13.8
12 Alat (Ang):         3.6501000000000006
13 Alat (Ang) (exp.):  7.300200000000001
14 Atoms in crystal:   32
15 Atoms in crystal (exp.): 32
16 AMU per crystal:    863.423999999994
17
18 Ecutfwfc:           50
19 Ecutrho:             200
20 Degauss:             0.04
21 Kpoints:             automatic 11 11 11 1 1 1
22
23 EOS Points:          11
24 Total Strain:         0.005
25
26 V0 (Bohr^3):         111.192295309
27 E0:                  -39.5027342866
28 B0 (RY/BOHR3):        0.00531912635008
29 B0 (GPA):              77.572208693
30 BOP:                 2.0
31
32 EC Points:            9
33 Total Strain:          0.005
34
35
36 Relaxed alat (Bohr):  7.6325799
37 Relaxed alat (Bohr) (exp): 15.2651598
38 Relaxed alat (Ang):     4.0376347671
39 Relaxed alat (Ang) (exp): 8.0752695342
40 Relaxed cp:             1.0   0.0   0.0
41                      0.0   1.0   0.0
42                      0.0   0.0   1.0
43 Relaxed volume (Bohr^3): 3557.16544569
44 Relaxed density (kgm^-3): 2722.71416748
45
46 Stiffness (RY/BOHR3):   0.0076028 0.0039542 0.0039563 0.0   0.0   0.0
47                      0.0039542 0.0075973 0.003945 0.0   0.0   0.0
48                      0.0039563 0.003945 0.0076068 0.0   0.0   0.0
49                      0.0   0.0   0.0   0.0023326 0.0   0.0
50                      0.0   0.0   0.0   0.0   0.0023349 0.0
51                      0.0   0.0   0.0   0.0   0.0   0.0023339
52
53 Stiffness (GPA):       110.877131 57.6666837 57.6978857 0.0   0.0   0.0
54                      57.6666837 110.796070 57.5331112 0.0   0.0   0.0
55                      57.6978857 57.5331112 110.934314 0.0   0.0   0.0
56                      0.0   0.0   0.0   34.0184756 0.0   0.0
57                      0.0   0.0   0.0   0.0   34.0513488 0.0
58                      0.0   0.0   0.0   0.0   0.0   34.037456

```

59
60 Compliance (1/GPA): 0.0140133 -0.0048022 -0.0047979 0.0 0.0 0.0
61 -0.0048022 0.0139977 -0.0047619 0.0 0.0 0.0
62 -0.0047979 -0.0047619 0.0139794 0.0 0.0 0.0
63 0.0 -0.0 -0.0 0.0293958 -0.0 -0.0
64 0.0 0.0 0.0 0.0 0.0293674 0.0
65 0.0 0.0 0.0 0.0 0.0 0.0293794
66
67 Bulk Modulus B (GPA): 75.378064563
68 BR (GPA): 75.3780315039
69 BV (GPA): 75.378097622
70
71 Shear Modulus G (GPA): 30.8455908955
72 GR: 30.6224032645
73 GV: 31.0687785264
74
75 Young's Modulus E (GPA): 81.4294731854
76
77 Poisson's Ratio v: 0.319953205977
78
79
80 L Elastic Wave V: 0.206857953612
81 T Elastic Wave V: 0.106437709845
82 M Elastic Wave V: 0.119193708075
83
84 Debye Temperature: 0.00139515659606
85
86 Melting Point: 1300.0K

O.3 BCC Iron [No Magnetism]

O.3.1 Major DFT Settings

Ecutwfc: 71

Ecutrho: 430

Smearing: 0.04

K-points: 9 9 9 1 1 1

Nspin: 1 (non-polarized calculation)

Pseudopotential: Fe.pbe-spn-kjpaw-psl.1.0.0.UPF

O.3.2 Equation of State and Elastic Constants

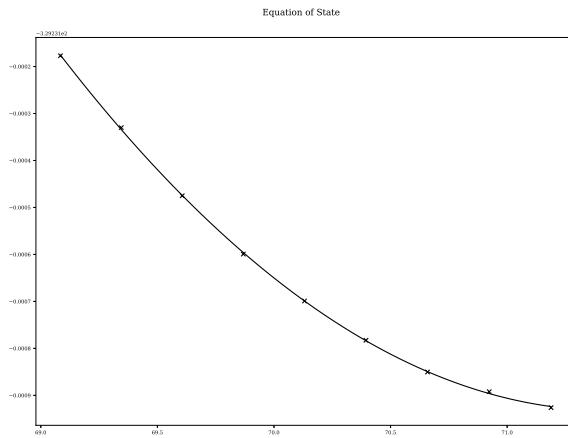


Figure O.3: Iron BCC equation of state plot

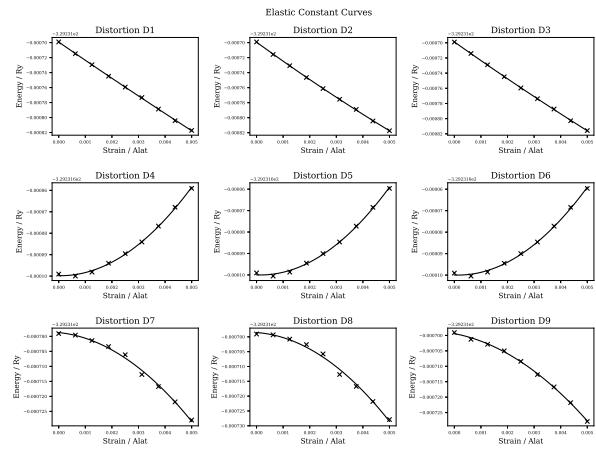


Figure O.4: Iron BCC elastic constants plot

O.3.3 Iron BCC Results File

Listing O.2: Iron BCC DFT Calculated Properties

```
1 ######
2 #          RESULTS          #
3 #####
4 Tue, 03 Nov 2020 06:01:22 +0000
5
6 INPUT SETTINGS
7 #####
8
9 Structure:           bcc
10 Size:                2
11 AO:                 2.8
12 AO Units:           ang
13 AO (Bohr):          10.584
14 AO (Angs):          5.598936
15
16 Ecutwfc:             71
17 Ecutrho:              430
18 Degauss:              0.04
19 Kpoints:              9 9 9 1 1 1
20 Kpoints Type:        automatic
21 EOS Points:           9
22 EOS Strain:            0.005
23 EC Points:             9
24 EC Strain:              0.005
25
26
27 RELAXED
28 #####
29
30 a0 (Bohr):           10.391466456
31 CP:                  1.0   0.0   0.0
32                      0.0   1.0   0.0
33                      0.0   0.0   1.0
34 Volume (Bohr^3/unit cell): 1122.09730705
35 Density KG/m^3:         8932.14485799
36 AMU per crystal unit:  893.5200000000003
37 Atoms per crystal unit: 16
38 Total Energy/Ry:        -5267.70721439
39 Energy per Atom/Ry:     -329.231700899375
40 Total Force Ry/Bohr:    1.7e-05
41 Force per atom Ry/Bohr: 1.0625e-06
42
43
44 a0 primitive (Bohr):    5.195733228
45 a0 primitive (Ang):      2.748542877612
46
47
48 EOS
49 #####
50
51 V0 (Bohr^3 / atom):    71.4368199046
52 E0 (Ry / atom):         -329.231932186
53 B0 (RY/BOHR3):          0.0195284174344
54 B0 (GPA):               287.272784669
55 BOP:                   4.98933790197
56
57
58 EC
```

```

59 #####
60
61 Stiffness (RY/BOHR3):   0.0043229 0.0225755 0.0216226 0.0      0.0      0.0
62                      0.0225755 0.0121066 0.0217004 0.0      0.0      0.0
63                      0.0216226 0.0217004 0.0083439 0.0      0.0      0.0
64                      0.0      0.0      0.0      0.0121324 0.0      0.0
65                      0.0      0.0      0.0      0.0      0.0123943 0.0
66                      0.0      0.0      0.0      0.0      0.0      0.0123441
67
68 Stiffness (GPA):      63.04417 329.233427 315.336007 0.0      0.0      0.0
69                      329.233427 176.558337 316.470988 0.0      0.0      0.0
70                      315.336007 316.470988 121.684505 0.0      0.0      0.0
71                      0.0      0.0      0.0      176.934632 0.0      0.0
72                      0.0      0.0      0.0      0.0      180.753497 0.0
73                      0.0      0.0      0.0      0.0      0.0      180.022531
74
75 Compliance (1/GPA): -0.0026218 0.0019907 0.001617 0.0      0.0      0.0
76                      0.0019907 -0.0030583 0.0027951 0.0      0.0      0.0
77                      0.001617 0.0027951 -0.0032415 0.0      0.0      0.0
78                      0.0      0.0      0.0      0.0056518 0.0      0.0
79                      0.0      0.0      0.0      0.0      0.0055324 0.0
80                      0.0      0.0      0.0      0.0      0.0      0.0055549
81
82 Stability:
83 C11:                  63.044169998 (Stable)
84 C11C22 - C12C12:     -97263.6756949 (Unstable)
85 C11*C22*C33+2*C12*C13*C23-C11*C23*C23-C33*C12*C12: 47561899.9206 (Stable)
86 C44:                  176.934632292 (Stable)
87 C55:                  180.753497554 (Stable)
88 C66:                  180.022531471 (Stable)
89
90 Bulk Modulus B (GPA): 255.592980231
91 BR (GPA):             257.478420517
92 BV (GPA):             253.707539944
93
94 Shear Modulus G (GPA): -643.096162505
95 GR:                   -1353.7508966
96 GV:                   67.5585715904
97
98 Young's Modulus E (GPA): -11960.7418627
99
100 Poisson's Ratio v:    8.29934165372
101
102
103 L Elastic Wave V:    nan
104 T Elastic Wave V:    nan
105 M Elastic Wave V:    nan
106
107 Debye Temperature:   nan
108
109 Melting Point:       1357.0K
110
111
112
113
114
115
116
117
118 References
119 =====

```

120
121 First Principles Calculations of Elastic Properties of Metals
122 M. J. Mehl, B. M. Klein, D. A. Papaconstantopoulos
123 1994
124
125 Ab Initio Study of the Elastic and Mechanical Properties of B19 TiAl
126 Y. Wen, L. Wang, H. Liu and L. Song
127 Crystals
128 2017
129
130 Density functional theory for calculation of elastic properties of orthorhombic crystals - applications to TiSi₂
131 P. Ravindran, Lars Fast, P. A. Korzhavyi, B. Johansson
132 Journal of Applied Physics
133 1998

O.4 BCC Iron [Ferromagnetic]

O.4.1 Major DFT Settings

Ecutwfc: 71

Ecutrho: 430

Smearing: 0.04

K-points: 9 9 9 1 1 1

Nspin: 2 (Collinear Spin)

Pseudopotential: Fe.pbe-spn-kjpaw-psl.1.0.0.UPF

O.4.2 Equation of State and Elastic Constants

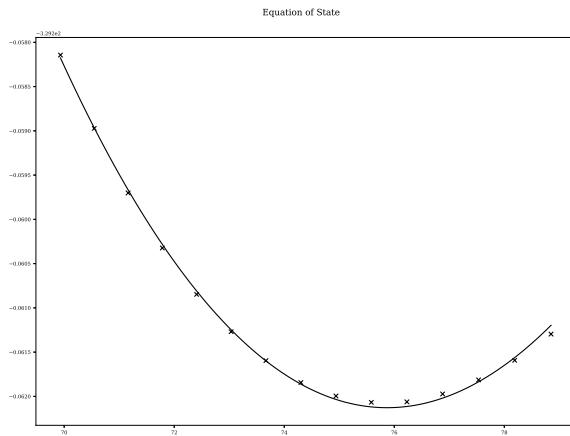


Figure O.5: Iron BCC (magnetic) equation of state plot

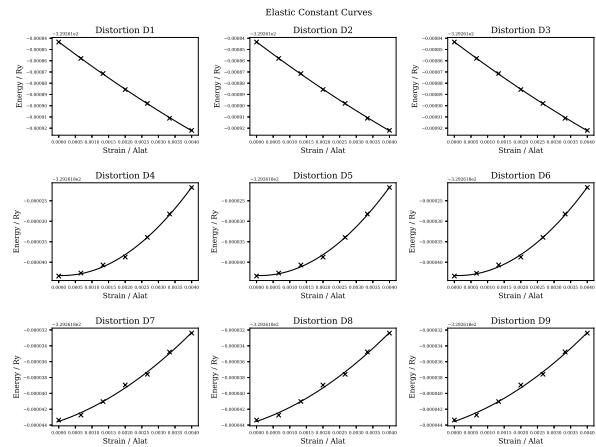


Figure O.6: Iron BCC (magnetic) elastic constants plot

O.4.3 Iron Ferromagnetic BCC Results File

Listing O.3: Iron BCC DFT calculated properties

```
1 ######
2 #          RESULTS          #
3 ######
4
5 INPUT SETTINGS
6 #####
7
8 Structure:           bcc
9 Size:                2
10 AO:                 2.8
11 AO Units:           ang
12 AO (Bohr):          10.584
13 AO (Angs):          5.598936
14
15 Ecutfc:              71
16 Ecutrho:             430
17 Degauss:              0.04
18 Kpoints:              9 9 9 1 1 1
19 Kpoints Type:         automatic
20 EOS Points:           15
21 EOS Strain:            0.02
22 EC Points:             7
23 EC Strain:             0.004
24
25
26 RELAXED
27 #####
28
29 a0 (Bohr):           10.5935195777
30 CP:                  1.0   0.0   0.0
31                      0.0   1.0   0.0
32                      0.0   0.0   1.0
33 Volume (Bohr^3/unit cell): 1188.83291445
34 Density KG/m^3:        8430.73536197
35 AMU per crystal unit:  893.5200000000003
36 Atoms per crystal unit: 16
37 Total Energy/Ry:       -5268.18846365
38 Energy per Atom/Ry:    -329.261778978125
39 Total Force Ry/Bohr:   2.2e-05
40 Force per atom Ry/Bohr: 1.375e-06
41
42
43 a0 primitive (Bohr):   5.2967597888519995
44 a0 primitive (Ang):     2.8019859283027078
45
46
47 EOS
48 #####
49
50 V0 (Bohr^3 / atom):    75.8702665574
51 E0 (Ry / atom):         -329.262127846
52 B0 (RY/BOHR3):          0.0162627705061
53 B0 (GPA):               239.23348553
54 BOP:                   3.31889649178
55
56
57 EC
58 ######
```

59
 60 Stiffness (RY/BOHR3): 0.0171516 0.0124976 0.0125676 0.0 0.0 0.0
 61 0.0124976 0.0170822 0.0125444 0.0 0.0 0.0
 62 0.0125676 0.0125444 0.0172259 0.0 0.0 0.0
 63 0.0 0.0 0.0 0.0095442 0.0 0.0
 64 0.0 0.0 0.0 0.0 0.0095793 0.0
 65 0.0 0.0 0.0 0.0 0.0 0.0095537
 66
 67 Stiffness (GPA): 250.132354 182.259753 183.281653 0.0 0.0 0.0
 68 182.259753 249.120311 182.943207 0.0 0.0 0.0
 69 183.281653 182.943207 251.216687 0.0 0.0 0.0
 70 0.0 0.0 0.0 139.188774 0.0 0.0
 71 0.0 0.0 0.0 0.0 139.701368 0.0
 72 0.0 0.0 0.0 0.0 0.0 139.328420
 73
 74 Compliance (1/GPA): 0.0104302 -0.0043908 -0.0044121 0.0 0.0 0.0
 75 -0.0043908 0.0104768 -0.0044261 0.0 0.0 0.0
 76 -0.0044121 -0.0044261 0.0104228 0.0 0.0 0.0
 77 0.0 0.0 0.0 0.0071845 0.0 0.0
 78 0.0 0.0 0.0 0.0 0.0071581 0.0
 79 0.0 0.0 0.0 0.0 0.0 0.0071773
 80
 81 Stability:
 82 C11: 250.132354807 (Stable)
 83 C11C22 - C12C12: 29094.4322943 (Stable)
 84 C11*C22*C33+2*C12*C13*C23-C11*C23*C23-C33*C12*C12: 11159910.6986 (Stable)
 85 C44: 139.188774163 (Stable)
 86 C55: 139.701368243 (Stable)
 87 C66: 139.328420919 (Stable)
 88
 89 Bulk Modulus B (GPA): 205.266912094
 90 BR (GPA): 205.262870468
 91 BV (GPA): 205.270953721
 92
 93 Shear Modulus G (GPA): 79.4449473647
 94 GR: 61.7805328168
 95 GV: 97.1093619125
 96
 97 Young's Modulus E (GPA): 211.100586016
 98
 99 Poisson's Ratio v: 0.328596676183
 100
 101
 102 L Elastic Wave V: 0.192124405208
 103 T Elastic Wave V: 0.0970734382597
 104 M Elastic Wave V: 0.108830261337
 105
 106 Debye Temperature: 0.000728462251696
 107
 108 Melting Point: 2076.0K
 109
 110
 111
 112
 113
 114
 115
 116
 117 References
 118 =====
 119

120 First Principles Calculations of Elastic Properties of Metals
121 M. J. Mehl, B. M. Klein, D. A. Papaconstantopoulos
122 1994
123
124 Ab Initio Study of the Elastic and Mechanical Properties of B19 TiAl
125 Y. Wen, L. Wang, H. Liu and L. Song
126 Crystals
127 2017
128
129 Density functional theory for calculation of elastic properties of orthorhombic crystals - applications to TiSi₂
130 P. Ravindran, Lars Fast, P. A. Korzhavyi, B. Johansson
131 Journal of Applied Physics
132 1998

O.5 FCC Iron [Antiferromagnetic]

O.5.1 Major DFT Settings

Ecutwfc: 71

Ecutrho: 430

Smearing: 0.04

K-points: 9 9 9 1 1 1

Nspin: 2 (Collinear Spin)

Pseudopotential: Fe.pbe-spn-kjpaw-psl.1.0.0.UPF

O.5.2 Equation of State and Elastic Constants

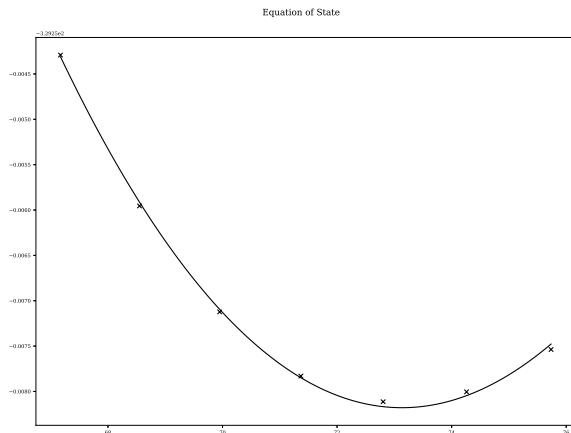


Figure O.7: Iron FCC (magnetic) equation of state plot

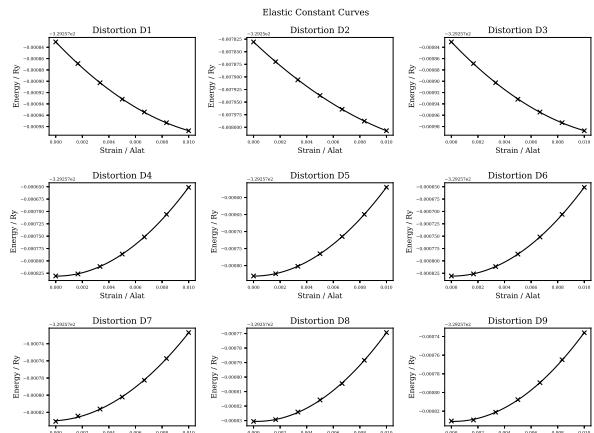


Figure O.8: Iron FCC (magnetic) elastic constants plot

O.5.3 Iron FCC Results File

Listing O.4: Iron FCC DFT Calculated Properties

```
1 ######
2 #          RESULTS          #
3 ######
4
5 INPUT SETTINGS
6 #####
7
8 Structure:           fcc
9 Size:                2
10 AO:                 3.4278156
11 AO Units:           ang
12 AO (Bohr):          12.957142968
13 AO (Angs):           6.854328630072
14
15 Ecutwfc:             71
16 Ecutrho:              430
17 Degauss:              0.04
18 Kpoints:              9 9 9 1 1 1
19 Kpoints Type:         automatic
20 EOS Points:           7
21 EOS Strain:            0.02
22 EC Points:              7
23 EC Strain:              0.01
24
25
26 RELAXED
27 #####
28
29 a0 (Bohr):           12.9381866998
30 CP:                  1.0    0.0    0.0
31                      0.0    1.05447369502   0.0
32                      0.0    0.0    0.999986980953
33 Volume (Bohr^3/unit cell): 2283.75934496
34 Density KG/m^3:        8777.4009231
35 AMU per crystal unit: 1787.040000000006
36 Atoms per crystal unit: 32
37 Total Energy/Ry:       -10536.25057837
38 Energy per Atom/Ry:     -329.2578305740625
39 Total Force Ry/Bohr:    5.1e-05
40 Force per atom Ry/Bohr: 1.59375e-06
41
42
43 a0 primitive (Bohr):    6.4690933498955
44 a0 primitive (Ang):      3.4221503820947197
45
46
47 EOS
48 #####
49
50 V0 (Bohr^3 / atom):    73.1321918692
51 E0 (Ry / atom):         -329.258178984
52 B0 (RY/BOHR3):          0.0153732026949
53 B0 (GPA):               226.147498244
54 BOP:                   3.74220549488
55
56
57 EC
58 ######
```

59
 60 Stiffness (RY/BOHR3): 0.0250005 0.009711 0.0160292 0.0 0.0 0.0
 61 0.009711 0.0204815 0.0089411 0.0 0.0 0.0
 62 0.0160292 0.0089411 0.0250019 0.0 0.0 0.0
 63 0.0 0.0 0.0 0.0127715 0.0 0.0
 64 0.0 0.0 0.0 0.0 0.0182966 0.0
 65 0.0 0.0 0.0 0.0 0.0 0.0127778
 66
 67 Stiffness (GPA): 364.598330 141.621966 233.764448 0.0 0.0 0.0
 68 141.621966 298.695202 130.394083 0.0 0.0 0.0
 69 233.764448 130.394083 364.619131 0.0 0.0 0.0
 70 0.0 0.0 0.0 186.254349 0.0 0.0
 71 0.0 0.0 0.0 0.0 266.830684 0.0
 72 0.0 0.0 0.0 0.0 0.0 186.346168
 73
 74 Compliance (1/GPA): 0.004966 -0.0011431 -0.002775 0.0 0.0 0.0
 75 -0.0011431 0.0042304 -0.00078 0.0 0.0 0.0
 76 -0.002775 -0.00078 0.0048006 0.0 0.0 0.0
 77 0.0 0.0 0.0 0.005369 0.0 0.0
 78 0.0 0.0 0.0 0.0 0.0037477 0.0
 79 0.0 0.0 0.0 0.0 0.0 0.0053664
 80
 81 Stability:
 82 C11: 364.598330439 (Stable)
 83 C11C22 - C12C12: 88846.9908456 (Stable)
 84 C11*C22*C33+2*C12*C13*C23-C11*C23*C23-C33*C12*C12: 34829887.0518 (Stable)
 85 C44: 186.254349116 (Stable)
 86 C55: 266.830684551 (Stable)
 87 C66: 186.346168623 (Stable)
 88
 89 Bulk Modulus B (GPA): 221.980853006
 90 BR (GPA): 217.35352142
 91 BV (GPA): 226.608184591
 92
 93 Shear Modulus G (GPA): 144.783544731
 94 GR: 126.872037967
 95 GV: 162.695051494
 96
 97 Young's Modulus E (GPA): 356.782113599
 98
 99 Poisson's Ratio v: 0.232122456533
 100
 101
 102 L Elastic Wave V: 0.217447522685
 103 T Elastic Wave V: 0.128433002219
 104 M Elastic Wave V: 0.142291445915
 105
 106 Debye Temperature: 0.0009653163572
 107
 108 Melting Point: 2528.0K
 109
 110
 111
 112
 113
 114
 115
 116
 117 References
 118 =====
 119

120 First Principles Calculations of Elastic Properties of Metals
121 M. J. Mehl, B. M. Klein, D. A. Papaconstantopoulos
122 1994
123
124 Ab Initio Study of the Elastic and Mechanical Properties of B19 TiAl
125 Y. Wen, L. Wang, H. Liu and L. Song
126 Crystals
127 2017
128
129 Density functional theory for calculation of elastic properties of orthorhombic crystals - applications to TiSi₂
130 P. Ravindran, Lars Fast, P. A. Korzhavyi, B. Johansson
131 Journal of Applied Physics
132 1998

O.6 FCC Palladium

O.6.1 Major DFT Settings

Ecutwfc: 71

Ecutrho: 430

Smearing: 0.04

K-points: 9 9 9 1 1 1

Nspin: 1 (non-polarized calculation)

Pseudopotential: Pd 106.42 Pd.pbe-spn-kjpaw_psl.1.0.0.UPF

O.6.2 Equation of State and Elastic Constants

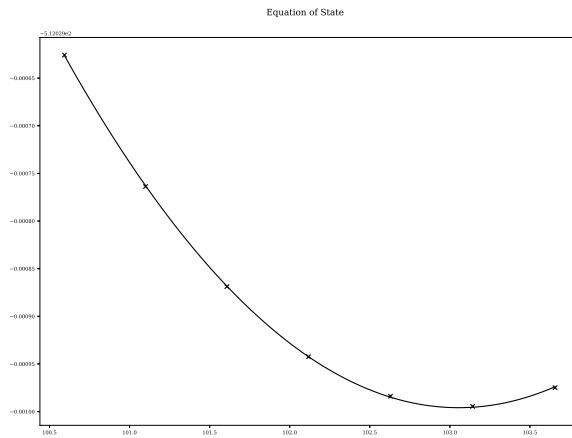


Figure O.9: Palladium FCC equation of state plot

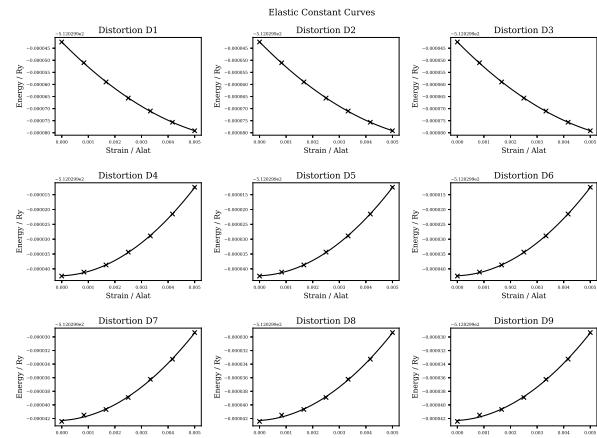


Figure O.10: Palladium FCC elastic constants plot

O.6.3 Palladium FCC Results File

Listing O.5: Palladium FCC DFT Calculated Properties

```
1 ######
2 #          RESULTS          #
3 ######
4
5 INPUT SETTINGS
6 #####
7
8 Structure:          fcc
9 Size:               2
10 AO:                3.89
11 AO Units:          ang
12 AO (Bohr):         14.7042
13 AO (Angs):          7.7785218
14
15 Ecutfc:             71
16 Ecutrho:            430
17 Degauss:            0.04
18 Kpoints:             9 9 9 1 1 1
19 Kpoints Type:        automatic
20 EOS Points:          7
21 EOS Strain:          0.005
22 EC Points:            7
23 EC Strain:            0.005
24
25
26 RELAXED
27 #####
28
29 a0 (Bohr):          14.8394198232
30 CP:                 1.0   0.0   0.0
31                   0.0   1.0   0.0
32                   0.0   0.0   1.0
33 Volume (Bohr^3/unit cell): 3267.76460963
34 Density KG/m^3:        11689.7296625
35 AMU per crystal unit: 3405.4400000000014
36 Atoms per crystal unit: 32
37 Total Energy/Ry:       -16384.95810846
38 Energy per Atom/Ry:    -512.029940889375
39 Total Force Ry/Bohr:   0.000188
40 Force per atom Ry/Bohr: 5.875e-06
41
42
43 a0 primitive (Bohr):    7.4197099116
44 a0 primitive (Ang):      3.9250265432364
45
46
47 EOS
48 #####
49
50 V0 (Bohr^3 / atom):    103.055133214
51 E0 (Ry / atom):         -512.029995887
52 B0 (RY/BOHR3):          0.012538146479
53 B0 (GPA):               184.442403779
54 BOP:                  4.7014295893
55
56
57 EC
58 ######
```

59
 60 Stiffness (RY/BOHR3): 0.0149806 0.010383 0.0103812 0.0 0.0 0.0
 61 0.010383 0.0149801 0.0103722 0.0 0.0 0.0
 62 0.0103812 0.0103722 0.0149778 0.0 0.0 0.0
 63 0.0 0.0 0.0 0.0055032 0.0 0.0
 64 0.0 0.0 0.0 0.0 0.0055036 0.0
 65 0.0 0.0 0.0 0.0 0.0 0.0055024
 66
 67 Stiffness (GPA): 218.471929 151.422406 151.395627 0.0 0.0 0.0
 68 151.422406 218.464281 151.264808 0.0 0.0 0.0
 69 151.395627 151.264808 218.430622 0.0 0.0 0.0
 70 0.0 0.0 0.0 80.2564574 0.0 0.0
 71 0.0 0.0 0.0 0.0 80.2629605 0.0
 72 0.0 0.0 0.0 0.0 0.0 80.2449828
 73
 74 Compliance (1/GPA): 0.0105867 -0.0043366 -0.0043346 0.0 0.0 0.0
 75 -0.0043366 0.0105705 -0.0043144 0.0 0.0 0.0
 76 -0.0043346 -0.0043144 0.0105702 0.0 0.0 0.0
 77 0.0 0.0 0.0 0.0124601 0.0 0.0
 78 0.0 0.0 0.0 0.0 0.012459 0.0
 79 0.0 0.0 0.0 0.0 0.0 0.0124618
 80
 81 Stability:
 82 C11: 218.471929458 (Stable)
 83 C11C22 - C12C12: 24799.5680651 (Stable)
 84 C11*C22*C33+2*C12*C13*C23-C11*C23*C23-C33*C12*C12: 7353517.38556 (Stable)
 85 C44: 80.256457365 (Stable)
 86 C55: 80.262960524 (Stable)
 87 C66: 80.244982752 (Stable)
 88
 89 Bulk Modulus B (GPA): 173.725818131
 90 BR (GPA): 173.725800894
 91 BV (GPA): 173.725835369
 92
 93 Shear Modulus G (GPA): 56.5595512391
 94 GR: 51.5472895795
 95 GV: 61.5718128987
 96
 97 Young's Modulus E (GPA): 153.067378318
 98
 99 Poisson's Ratio v: 0.35315234089
 100
 101
 102 L Elastic Wave V: 0.145988361768
 103 T Elastic Wave V: 0.0695585857774
 104 M Elastic Wave V: 0.0782389075958
 105
 106 Debye Temperature: 0.000471028127273
 107
 108 Melting Point: 1910.0K
 109
 110
 111
 112
 113
 114
 115
 116
 117 References
 118 =====
 119

120 First Principles Calculations of Elastic Properties of Metals
121 M. J. Mehl, B. M. Klein, D. A. Papaconstantopoulos
122 1994
123
124 Ab Initio Study of the Elastic and Mechanical Properties of B19 TiAl
125 Y. Wen, L. Wang, H. Liu and L. Song
126 Crystals
127 2017
128
129 Density functional theory for calculation of elastic properties of orthorhombic crystals - applications to TiSi₂
130 P. Ravindran, Lars Fast, P. A. Korzhavyi, B. Johansson
131 Journal of Applied Physics
132 1998

O.7 FCC Ruthenium

O.7.1 Major DFT Settings

Ecutwfc: 71

Ecutrho: 430

Smearing: 0.04

K-points: 9 9 9 1 1 1

Nspin: 1 (non-polarized calculation)

Pseudopotential: Ru 101.07 Ru.pbe-spn-kjpaw-psl.1.0.0.UPF

O.7.2 Equation of State and Elastic Constants

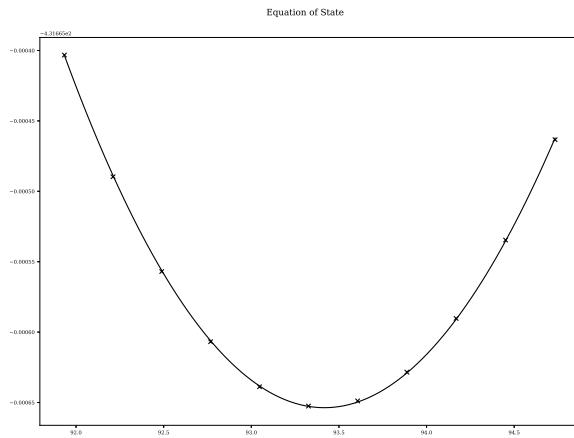


Figure O.11: Ruthenium FCC equation of state plot

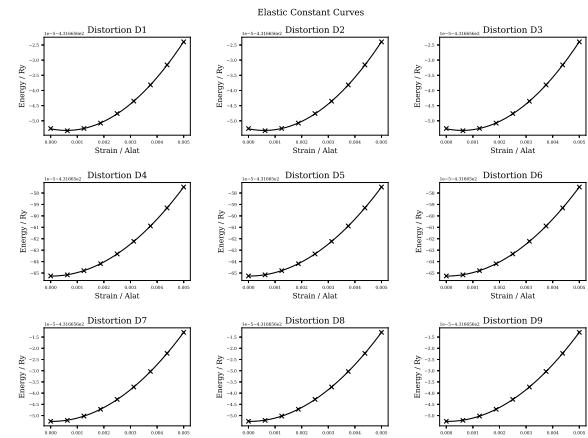


Figure O.12: Ruthenium FCC elastic constants plot

O.7.3 Ruthenium FCC Results File

Listing O.6: Ruthenium FCC DFT Calculated Properties

```
1 ######
2 #          RESULTS          #
3 #####
4 Fri, 26 Nov 2021 04:44:37 +0000
5
6 INPUT SETTINGS
7 #####
8
9 Structure Type:      fcc
10 Size:                2
11 AO:                  4.0
12 AO Units:           angs
13 AO (Bohr):          15.12
14 AO (Angs):          7.99848
15
16 Ecutwfc:             71
17 Ecutrho:              430
18 Degauss:              0.04
19 Kpoints:              9 9 9 1 1 1
20 Kpoints Type:        automatic
21 EOS Points:          11
22 EOS Strain:           0.005
23 EC Points:            9
24 EC Strain:            0.005
25
26
27 RELAXED
28 #####
29
30 a0 (Bohr):           14.4007416
31 CP:                  1.0   0.0   0.0
32                      0.0   1.0   0.0
33                      0.0   0.0   1.0
34 Volume (Bohr^3/unit cell): 2986.445358287136
35 Density KG/m^3:        12147.856996566989
36 AMU per crystal unit: 3234.240000000001
37 Atoms per crystal unit: 32
38 Total Energy/Ry:       -13813.30088106
39 Energy per Atom/Ry:    -431.665652533125
40 Total Force Ry/Bohr:   0.000137
41 Force per atom Ry/Bohr: 4.28125e-06
42
43
44 a0 primitive (Bohr):    7.2003708
45 a0 primitive (Ang):     3.8089961532000003
46
47
48 EOS
49 #####
50
51 V0 (Bohr^3 / atom):    93.41718637560044
52 E0 (Ry / atom):         -431.66565367812325
53 B0 (RY/BOHR3):          0.020926676273491183
54 B0 (GPA):               307.84187132119206
55 BOP:                   2.2239400402684173
56
57
58 EC
```

```

59 #####
60
61 Stiffness (RY/BOHR3):   0.0323332 0.0150236 0.0150295 0.0      0.0      0.0
62                      0.0150236 0.032334 0.0150304 0.0      0.0      0.0
63                      0.0150295 0.0150304 0.0323458 0.0      0.0      0.0
64                      0.0      0.0      0.0      0.0167987 0.0      0.0
65                      0.0      0.0      0.0      0.0      0.0167978 0.0
66                      0.0      0.0      0.0      0.0      0.0      0.0167975
67
68 Stiffness (GPA):      471.535040 219.099143 219.185184 0.0      0.0      0.0
69                      219.099143 471.546672 219.197356 0.0      0.0      0.0
70                      219.185184 219.197356 471.719577 0.0      0.0      0.0
71                      0.0      0.0      0.0      244.986869 0.0      0.0
72                      0.0      0.0      0.0      0.0      244.973541 0.0
73                      0.0      0.0      0.0      0.0      0.0      244.969279
74
75 Compliance (1/GPA):  0.0030075 -0.0009538 -0.0009542 0.0      0.0      0.0
76                      -0.0009538 0.0030075 -0.0009543 0.0      0.0      0.0
77                      -0.0009542 -0.0009543 0.0030067 0.0      0.0      0.0
78                      0.0      0.0      0.0      0.0040819 0.0      0.0
79                      0.0      0.0      0.0      0.0      0.0040821 0.0
80                      0.0      0.0      0.0      0.0      0.0      0.0040821
81
82 Stability:
83 C11:                  471.535040254 (Stable)
84 C11C22 - C12C12:     174346.3443625905 (Stable)
85 C11*C22*C33+2*C12*C13*C23-C11*C23*C23-C33*C12*C12: 80639667.71993023 (Stable)
86 C44:                  244.986869322 (Stable)
87 C55:                  244.97354179 (Stable)
88 C66:                  244.969279805 (Stable)
89
90 Bulk Modulus B (GPA): 303.30717357998026
91 BR (GPA):              303.3071627547384
92 BV (GPA):              303.3071844052222
93
94 Shear Modulus G (GPA): 187.73207685789976
95 GR:                   177.99024187226613
96 GV:                   197.47391184353336
97
98 Young's Modulus E (GPA): 466.87258322673495
99
100 Poisson's Ratio v:   0.24345447789438046
101
102
103 L Elastic Wave V:   0.21347878392203118
104 T Elastic Wave V:   0.12431381857023048
105 M Elastic Wave V:   0.1379068118794884
106
107 Debye Temperature:  0.0008555430077179703
108
109 Melting Point:      3072.0K
110
111
112
113
114
115
116
117
118 References
119 =====

```

120
121 First Principles Calculations of Elastic Properties of Metals
122 M. J. Mehl, B. M. Klein, D. A. Papaconstantopoulos
123 1994
124
125 Ab Initio Study of the Elastic and Mechanical Properties of B19 TiAl
126 Y. Wen, L. Wang, H. Liu and L. Song
127 Crystals
128 2017
129
130 Density functional theory for calculation of elastic properties of orthorhombic crystals - applications to TiSi₂
131 P. Ravindran, Lars Fast, P. A. Korzhavyi, B. Johansson
132 Journal of Applied Physics
133 1998

Appendix P

QEFORFIT

The full code is available at <https://github.com/BenPalmer1983/qeforfit>

The random Gaussian heating function is given below. It could be simplified, or moved into its own library, but it works as intended for now.

Listing P.1: Gaussian type random heating function

```
1 #!/bin/python3
2 #####
3
4 import os
5 import sys
6 import time
7 import hashlib
8 import math
9 import numpy
10 import matplotlib.pyplot as plt
11
12 class rand_dist:
13
14     def __init__(self, seed=None):
15         # RNG vars
16         if seed is None:
17             self.seed = 12791244
18         else:
19             self.seed = seed
20         self.xn = self.seed
21         self.m = 4294967296
22         self.a = 1103515245
23         self.c = 12345
24         #Dist Type vars
25         self.lower = 0.0e0
26         self.upper = 1.0e0
27         self.distType = "flat"
28         # make a new distribution table
29         self.table = RandDistTable()
30
31     def setSeed(self, seed):
32         self.seed = seed
33
34     def randomSeed(self):
35         currentTime = time.time()
36         myhash = hashlib.md5()
37         message = "randnum"+str(currentTime)
38         myhash.update(message.encode())
```

```

39     seed = myhash.hexdigest()
40     seedInt = int("0x"+seed[0:8],0)
41     self.xn = (self.xn + seedInt) % self.m
42
43 def setRange(self,lower,upper):
44     self.lower = lower
45     self.upper = upper
46
47 def flat(self):
48     self.distType = "flat"
49     self.table.set = False
50
51 def sqrt(self):
52     self.distType = "sqrt"
53     self.table.set = False
54
55 def gheat(self, lower=-1.0e0, upper=1.0e0, p1=1.0, p2=0.0, p3=1.0, p4=4.0):
56     self.distType = "gheat"
57     self.table.set = False
58     self.lower = lower
59     self.upper = upper
60     self.p1 = p1 # sigma
61     self.p2 = p2 # mu
62     self.p3 = p3 # factor
63     self.p4 = p4 # factor
64
65 def doubleGaussian(self, lower=-1.0e0, upper=1.0e0, p1=1.0, p2=0.0, p3=1.0, p4=4.0, p5=1.5, p6=0.8, p7=1.0, p8
66     =4.0):
67     self.distType = "doubleGaussian"
68     self.table.set = False
69     self.lower = lower
70     self.upper = upper
71     self.p1 = p1
72     self.p2 = p2
73     self.p3 = p3
74     self.p4 = p4
75     self.p5 = p5
76     self.p6 = p6
77     self.p7 = p7
78     self.p8 = p8
79
80 def getFloat(self):
81     # Get a random float between 0 and 1
82     self.xn = (self.a * self.xn + self.c) % self.m
83     randFloat = self.xn / self.m
84     randFloat = numpy.random.uniform(0.0, 1.0)
85     return randFloat
86
87 def rng(self):
88     # Choose function
89     if(self.distType=="flat"):
90         self.randFloat = self.getFloat()
91         randOut = self.lower + self.randFloat * (self.upper - self.lower)
92     # Square root
93     if(self.distType=="sqrt"):
94         self.randFloat = self.getFloat()
95         randOut = self.lower + self.randFloat * (self.upper - self.lower)
96         randOut = math.sqrt(abs(randOut))
97     # gheat
98     if(self.distType=="gheat"):
99         randOut = self.randDistF()

```

```

99     # doubleGaussian
100    if(self.distType=="doubleGaussian"):
101        randOut = self.randDistF()
102    # store and return output
103    self.randOut = randOut
104    return randOut
105
106 def randDistF(self):
107     # If a table hasn't been built for the function, build it now
108     if(self.table.set==False):
109         self.rand_MakeTable()
110         #self.table.display()
111
112     # Loop attempts
113     loopTrials = True
114     i = 0
115     while(loopTrials and i<10000):
116         i = i + 1
117         randA = self.getFloat() # Block
118         randB = self.getFloat() # xVal
119         randC = self.getFloat() # yVal
120         if(randA==0):
121             randBlock = 0
122         else:
123             randBlock = math.ceil(randA * self.table.size)-1
124         xA = self.table.points_xA[randBlock]
125         x_m = self.table.points_x_m[randBlock]
126         xB = self.table.points_xB[randBlock]
127         yA = self.table.points_yA[randBlock]
128         y_m = self.table.points_y_m[randBlock]
129         yB = self.table.points_yB[randBlock]
130         yMax = self.table.points_yMax[randBlock]
131
132         # set x,y coords
133         x = xA + randB * (xB - xA)
134         y = randC * yMax
135
136         # Set up list
137         interpPoints = [[0 for x in range(2)] for y in range(3)]
138         interpPoints[0][0] = xA
139         interpPoints[0][1] = yA
140         interpPoints[1][0] = x_m
141         interpPoints[1][1] = y_m
142         interpPoints[2][0] = xB
143         interpPoints[2][1] = yB
144
145         # Interp
146         fx = self.interp(x,interpPoints)
147         if(y<=fx):
148             loopTrials = False
149             randNumber = x
150
151             return randNumber
152
153     def rand_MakeTable(self):
154         # Calculate total area under function
155         self.table.initPoints(256)
156         randRange = self.upper - self.lower
157         xInc = randRange / (self.table.targetSize - 1)
158         xA = self.lower
159         xB = xA + xInc
160         area = 0.0
161         for i in range(0,self.table.targetSize):
162             ## Choose function

```

```

160 #####-----#
161     if(self.distType=="gheat"):
162         yA = self.rand_Gaussian(xA)
163         yB = self.rand_Gaussian(xB)
164     if(self.distType=="doubleGaussian"):
165         yA = self.rand_DoubleGaussian(xA)
166         yB = self.rand_DoubleGaussian(xB)
167 #####-----#
168     # Calc area
169     area = area + 0.5 * xInc * (yA + yB)
170     # Increment
171     xA = xB
172     xB = xA + xInc
173     # Area when split into blocks
174     blockArea = area / self.table.targetSize
175     xA = self.lower
176     runLoop = True
177     i = 0
178     while(runLoop):
179         i = i + 1
180     ### Choose function
181 #####-----#
182     if(self.distType=="gheat"):
183         yA = self.rand_Gaussian(xA)
184     if(self.distType=="doubleGaussian"):
185         yA = self.rand_DoubleGaussian(xA)
186 #####-----#
187     # Find xB
188     xInc = randRange * 0.0001
189     xB = xA
190     area = 0.0
191     yMax = yA
192     findArea = True
193     while(findArea):
194         xB = xB + xInc
195         yB = self.rand_Gaussian(xB)
196         if(yB>yA):
197             yMax = yB
198         area = area + 0.5 * xInc * yMax
199         if(area>blockArea):
200             findArea = False # Area has been found
201         xB = xA + blockArea / yMax
202         if(xB>self.upper):
203             xB = self.upper
204             yMax = area/(xB - xA)
205             runLoop = False
206             # set table size
207             self.table.size = i
208             #Run calculations
209     ### Choose function
210 #####-----#
211     x_m = 0.5 * (xA + xB)
212     if(self.distType=="gheat"):
213         yA = self.rand_Gaussian(xA)
214         y_m = self.rand_Gaussian(x_m)
215         yB = self.rand_Gaussian(xB)
216     if(self.distType=="doubleGaussian"):
217         yA = self.rand_DoubleGaussian(xA)
218         y_m = self.rand_DoubleGaussian(x_m)
219         yB = self.rand_DoubleGaussian(xB)
220 #####-----#

```

```

221     # Store data points
222     self.table.points_yMax.append(yMax)
223     self.table.points_xA.append(xA)
224     self.table.points_x_m.append(x_m)
225     self.table.points_xB.append(xB)
226     self.table.points_yA.append(yA)
227     self.table.points_y_m.append(y_m)
228     self.table.points_yB.append(yB)
229     # Set xA for next loop
230     xA = xB
231     # Set table to true
232     self.table.set = True
233
234 def rand_Gaussian(self, x):
235     fx = self.p3 * (0.398942 / self.p1)
236     expV = (-0.5/(self.p1*self.p1)) * (self.p4 * self.p4 * (x - self.p2) * (x - self.p2))
237     fx = fx * math.exp(expV)
238     return fx
239
240 def rand_DoubleGaussian(self, x):
241     fxA = self.p3 * (0.398942 / self.p1)
242     expV = (-0.5/(self.p1*self.p1)) * (self.p4 * self.p4 * (x - self.p2) * (x - self.p2))
243     fxA = fxA * math.exp(expV)
244     fxB = self.p7 * (0.398942 / self.p5)
245     expV = (-0.5/(self.p5*self.p5)) * (self.p8 * self.p8 * (x - self.p6) * (x - self.p6))
246     fxB = fxB * math.exp(expV)
247     fx = fxA + fxB
248     return fx
249
250 def makeTally(self, tallySize=200, sampleSize=100000):
251     # Declare list
252     self.tally = []
253     self.tallyX = []
254     self.tallySize = tallySize
255     sampleSize = 1000000
256     self.sampleSize = sampleSize
257     halfIncrement = (self.upper - self.lower) / (2 * tallySize)
258     # Loop through
259     for i in range(0,tallySize):
260         self.tally.append(0)
261         xVal = self.lower + (2 * i + 1) * halfIncrement
262         self.tallyX.append(xVal)
263         for i in range(0,self.sampleSize):
264             randNum = self.rng()
265             binNum = randNum - self.lower
266             binNum = binNum / (self.upper - self.lower)
267             binNum = math.floor(binNum * tallySize)
268             # Force to be in range
269             if(binNum>=tallySize):
270                 binNum = tallySize - 1
271             if(binNum<0):
272                 binNum = 0
273             self.tally[binNum] = self.tally[binNum] + 1
274
275 def displayTally(self):
276     print("====")
277     print("    Tally: ",self.distType)
278     print("====")
279     for i in range(0,self.tallySize):
280         #print(i," ",self.tally[i]," ",self.tallyX[i]," ",self.tally[i])
281         print('{0:3G},{1:8G},{2:5f},{3:8G}'.format(i,self.tally[i],self.tallyX[i],self.tally[i]))

```

```

282     print("====")
283     print()
284
285     plt.figure(figsize=(12,8))
286     plt.plot(self.tallyX[:,], self.tally[:,], '+')
287     plt.xlabel('Random Number')
288     plt.ylabel('Count per bin (0.01 width)')
289     plt.title('G-Heat 1,000,000 random numbers')
290     plt.grid(True)
291     plt.savefig('gheat-distribution', format='eps')
292     plt.show()
293     plt.close('all')
294
295
296 @staticmethod
297 def interp(x, points):
298     # Lagrange interpolation
299     output = 0.0
300     count = len(points)
301     coefficients = []
302     # Make coefficients
303     for i in range(0, count):
304         numerator = 1.0
305         denominator = 1.0
306         for j in range(0, count):
307             if(i!=j):
308                 numerator = numerator * (x - points[j][0])
309                 denominator = denominator * (points[i][0] - points[j][0])
310             coefficients.append(numerator/denominator)
311     # Calculate y
312     y = 0
313     for i in range(0, count):
314         y = y + points[i][1] * coefficients[i]
315     return y

```

Appendix Q

EAMPA Code

The full code is available at https://github.com/BenPalmer1983/eampa_v3

Several of the important functions and subroutines that make up the code are listed below. In some cases they will require Fortran module globals or Python classes in order to run.

Q.1 Energy, Force, and Stress Subroutine

The subroutine that calculates energy, stress and forces is given in listing ???. It uses OpenMP to split the calculations for individual configurations so each is computed by an individual process, meaning multiple configurations may be calculated in parallel. It handles one or multiple density/embedding groups where either standard EAM, 2BEAM or more bands are required.

Listing Q.1: Energy, Force and Stress subroutine (Fortran)

```
1 SUBROUTINE energy_force_stress()
2 !#####
3 INTEGER(KIND=StandardInteger) :: cn
4 !#####
5
6 CALL start_t()
7 !$OMP PARALLEL
8 !$OMP DO
9 DO cn = 1, cc
10   CALL energy_inner(cn)
11   CALL energy_force_stress_inner(cn)
12 END DO
13 !$OMP END DO
14 !$OMP END PARALLEL
15 CALL end_t(efs_timer)
16 efs_timer_sum = efs_timer_sum + efs_timer
17 END SUBROUTINE energy_force_stress
18
19
20
21 SUBROUTINE energy_force_stress_inner(cn)
22 !#####
23 INTEGER(KIND=StandardInteger) :: cn
24 !#####
25 CALL energy_force_stress_calc(cn, &
26     nlist_l(key(cn, 5):key(cn, 6),:), &
27     nlist_r(key(cn, 5):key(cn, 6),:), &
28     labels(key(cn, 1):key(cn, 2)), &
```

```

29         coords(key(cn, 1):key(cn, 2),:) &
30     )
31 END SUBROUTINE energy_force_stress_inner
32
33
34 SUBROUTINE energy_force_stress_calc(cn, nl_l, nl_r, c_l, c_r)
35 !#####
36 INTEGER(KIND=StandardInteger) :: cn
37 INTEGER(KIND=StandardInteger) :: nl_l(:, :)
38 REAL(kind=DoubleReal) :: nl_r(:, :)
39 INTEGER(KIND=StandardInteger) :: c_l(:)
40 REAL(kind=DoubleReal) :: c_r(:, :)
41 !#####
42 INTEGER(KIND=StandardInteger) :: i, j, n, f, fn, dn
43 INTEGER(KIND=StandardInteger) :: cc, nc
44 REAL(kind=DoubleReal) :: fx, dfwdx
45 REAL(kind=DoubleReal) :: fx_arr(1:2)
46 REAL(kind=DoubleReal) :: fv(1:3)
47 REAL(kind=DoubleReal), ALLOCATABLE :: density(:, :)
48 REAL(kind=DoubleReal), ALLOCATABLE :: density_grad_ab(:, :)
49 REAL(kind=DoubleReal), ALLOCATABLE :: density_grad_ba(:, :)
50 REAL(kind=DoubleReal), ALLOCATABLE :: nl_force(:, :)
51 !REAL(kind=DoubleReal) :: s(1:3, 1:3)
52 REAL(kind=DoubleReal) :: epA, epB, dpAB, dpBA
53 !#####
54 ! nlist_l(nc, 1) = ids(an)
55 ! nlist_l(nc, 2) = labels(an)
56 ! nlist_l(nc, 3) = ghostids(gn)
57 ! nlist_l(nc, 4) = ghostlabels(gn)
58 ! nlist_r(nc, 1) = r_mag
59 ! nlist_r(nc, 2:4) = r(1:3)/r_mag
60
61 cc = SIZE(c_l, 1)
62 nc = SIZE(nl_l, 1)
63 ALLOCATE(density(1:cc, 1:fgroup_max))
64 ALLOCATE(density_grad_ab(1:nc, 1:fgroup_max))
65 ALLOCATE(density_grad_ba(1:nc, 1:fgroup_max))
66 ALLOCATE(nl_force(1:nc, 1:3))
67 config_energy(cn, :) = 0.0D0
68 config_forces(cn, 1:key(cn, 20), 1:3) = 0.0D0
69 density = 0.0D0
70 density_grad_ab = 0.0D0
71 density_grad_ba = 0.0D0
72
73 DO n = 1, nc
74     ! PAIR ENERGY
75     CALL search_f(1, nl_l(n, 2), nl_l(n, 4), 0, nl_r(n, 1), fx)
76     config_energy(cn, 1) = config_energy(cn, 1) + fx
77     ! PAIR FORCE
78     CALL search_grad(1, nl_l(n, 2), nl_l(n, 4), 0, nl_r(n, 1), dfwdx)
79     fv(:) = dfwdx * nl_r(n, 2:4)
80     config_forces(cn, nl_l(n, 1), :) = config_forces(cn, nl_l(n, 1), :) - fv(:)
81     config_forces(cn, nl_l(n, 3), :) = config_forces(cn, nl_l(n, 3), :) + fv(:)
82     nl_force(n,:) = - fv(:)
83
84     ! LOOP THROUGH DENSITY GROUPS
85     DO fn = 1, fgroup_max
86         ! DENSITY OF ATOM B AT A (B with any atom)
87         CALL search_f(2, nl_l(n, 2), 0, fn, nl_r(n, 1), fx)
88         CALL search_grad(2, nl_l(n, 2), 0, fn, nl_r(n, 1), dfwdx)
89         density(nl_l(n, 1), fn) = density(nl_l(n, 1), fn) + fx

```

```

90     density_grad_ab(n, fn) = density_grad_ab(n, fn) + dfwdx
91     ! DENSITY OF ATOM B AT A (B with only atom A)
92     CALL search_f(2, nl_l(n, 2), nl_l(n, 4), fn, nl_r(n, 1), fx)
93     CALL search_grad(2, nl_l(n, 2), nl_l(n, 4), fn, nl_r(n, 1), dfwdx)
94     density(nl_l(n, 1), fn) = density(nl_l(n, 1), fn) + fx
95     density_grad_ab(n, fn) = density_grad_ab(n, fn) + dfwdx
96     ! DENSITY OF ATOM A AT B (A with any atom)
97     CALL search_f(2, nl_l(n, 4), 0, fn, nl_r(n, 1), fx)
98     CALL search_grad(2, nl_l(n, 4), 0, fn, nl_r(n, 1), dfwdx)
99     density(nl_l(n, 3), fn) = density(nl_l(n, 3), fn) + fx
100    density_grad_ba(n, fn) = density_grad_ba(n, fn) + dfwdx
101    ! DENSITY OF ATOM A AT B (A with only atom B)
102    CALL search_f(2, nl_l(n, 4), nl_l(n, 2), fn, nl_r(n, 1), fx)
103    CALL search_grad(2, nl_l(n, 4), nl_l(n, 2), fn, nl_r(n, 1), dfwdx)
104    density(nl_l(n, 3), fn) = density(nl_l(n, 3), fn) + fx
105    density_grad_ba(n, fn) = density_grad_ba(n, fn) + dfwdx
106  END DO
107 END DO
108
109 ! EMBED ENERGY
110 DO n = 1, cc
111   ! LOOP THROUGH DENSITY GROUPS
112   DO fn = 1, fgroup_max
113     CALL search_f(3, nl_l(n, 4), 0, fn, density(n, fn), fx)
114     config_energy(cn, 2) = config_energy(cn, 2) + fx
115   END DO
116 END DO
117
118 DO n = 1, nc
119   DO fn = 1, fgroup_max
120     ! @Fi(p)/@p Gradient of embedding energy for atom A
121     CALL search_grad(3, nl_l(n, 2), 0, fn, density(nl_l(n, 1), fn), dfwdx)
122     epA = dfwdx
123
124     ! @Fi(p)/@p Gradient of embedding energy for atom A
125     CALL search_grad(3, nl_l(n, 4), 0, fn, density(nl_l(n, 3), fn), dfwdx)
126     epB = dfwdx
127
128     ! @Pij(r)/@r Gradient of density function at A due to atom B
129     dpAB = density_grad_ab(n,fn)
130
131     ! @Pji(r)/@r Gradient of density function at B due to atom A
132     dpBA = density_grad_ba(n,fn)
133
134     fv(:) = (epA * dpBA + epB * dpAB) * nl_r(n, 2:4)
135     config_forces(cn, nl_l(n, 1), :) = config_forces(cn, nl_l(n, 1), :) - fv(:)
136     config_forces(cn, nl_l(n, 3), :) = config_forces(cn, nl_l(n, 3), :) + fv(:)
137     nl_force(n,:)= nl_force(n,:)- fv(:)
138   END DO
139 END DO
140
141 ! STRESS
142 config_stresses(cn, 1:3, 1:3) = 0.0D0
143 DO n = 1, nc
144   IF(nlisthalo(n))THEN
145     DO i = 1,3
146       DO j = 1,3
147         config_stresses(cn, i,j) = config_stresses(cn, i,j) + (nl_r(n, 1+i) * nl_force(n,j))
148       END DO
149     END DO
150   END IF

```

```

151 END DO
152 config_stresses(cn, 1:3, 1:3) = config_stresses(cn, 1:3, 1:3) / (2.0D0 * volume(cn))
153 stresses_calculated(cn) = 1
154
155 max_density = MAX(max_density, MAXVAL(density))
156
157 DEALLOCATE(density)
158 DEALLOCATE(density_grad_ab)
159 DEALLOCATE(density_grad_ba)
160 DEALLOCATE(nl_force)
161 config_energy(cn, 3) = config_energy(cn, 1) + config_energy(cn, 2)
162 config_energy(cn, 4:6) = config_energy(cn, 1:3) / cc
163
164 END SUBROUTINE energy_force_stress_calc

```

Q.2 Bulk Properties: Add Configuration Subroutine

In this subroutine, the structures used to compute bulk properties are created. The atom positions are set, depending on the crystal type, and then distortions are applied to the basis vector for each configuration in order to compute the equation of state and elastic constants (listing ??).

Listing Q.2: Bulk Properties - add configuration subroutine (Fortran)

```

1 SUBROUTINE add_bp_config(rcut_in, alat_in, uv_in, label_in, crystal_type_in, expansion_in, bp_id)
2 !#####
3 REAL(kind=DoubleReal), INTENT(IN) :: rcut_in
4 REAL(kind=DoubleReal), INTENT(IN) :: alat_in
5 REAL(kind=DoubleReal), INTENT(IN) :: uv_in(1:3,1:3)
6 INTEGER(KIND=StandardInteger), INTENT(IN) :: label_in
7 INTEGER(KIND=StandardInteger), INTENT(IN) :: crystal_type_in
8 INTEGER(KIND=StandardInteger), INTENT(IN) :: expansion_in
9 INTEGER(KIND=StandardInteger), INTENT(OUT) :: bp_id
10 !#####
11 INTEGER(KIND=StandardInteger) :: primitive_labels(1:20)
12 REAL(kind=DoubleReal) :: primitive_coords(1:20,1:3)
13 INTEGER(KIND=StandardInteger) :: atoms_per_crystal
14 !#####
15
16 ! SC = 1, BCC = 2, FCC = 3, ZB = 4
17 !#####
18
19 ! SC
20 IF(crystal_type_in .EQ. 1)THEN
21   atoms_per_crystal = 1
22   primitive_labels(1) = label_in
23   primitive_coords(1,1) = 0.0D0
24   primitive_coords(1,2) = 0.0D0
25   primitive_coords(1,3) = 0.0D0
26   CALL add_inner(rcut_in, alat_in, uv_in, label_in, crystal_type_in, expansion_in, atoms_per_crystal, &
27                 primitive_labels(1:1), primitive_coords(1:1,1:3), bp_id)
28 ! BCC
29 ELSE IF(crystal_type_in .EQ. 2)THEN
30
31   atoms_per_crystal = 2
32
33   primitive_labels(1) = label_in
34   primitive_labels(2) = label_in
35
36   primitive_coords(1,1) = 0.0D0

```

```

37 primitive_coords(1,2) = 0.0D0
38 primitive_coords(1,3) = 0.0D0
39 primitive_coords(2,1) = 0.5D0
40 primitive_coords(2,2) = 0.5D0
41 primitive_coords(2,3) = 0.5D0
42
43 CALL add_inner(rcut_in, alat_in, uv_in, label_in, crystal_type_in, expansion_in, atoms_per_crystal, &
44 primitive_labels(1:2), primitive_coords(1:2,1:3), bp_id)
45
46
47
48 ! FCC
49 ELSE IF(crystal_type_in .EQ. 3)THEN
50
51 atoms_per_crystal = 4
52
53 primitive_labels(1) = label_in
54 primitive_labels(2) = label_in
55 primitive_labels(3) = label_in
56 primitive_labels(4) = label_in
57
58 primitive_coords(1,1) = 0.0D0
59 primitive_coords(1,2) = 0.0D0
60 primitive_coords(1,3) = 0.0D0
61 primitive_coords(2,1) = 0.5D0
62 primitive_coords(2,2) = 0.5D0
63 primitive_coords(2,3) = 0.0D0
64 primitive_coords(3,1) = 0.5D0
65 primitive_coords(3,2) = 0.0D0
66 primitive_coords(3,3) = 0.5D0
67 primitive_coords(4,1) = 0.0D0
68 primitive_coords(4,2) = 0.5D0
69 primitive_coords(4,3) = 0.5D0
70
71 CALL add_inner(rcut_in, alat_in, uv_in, label_in, crystal_type_in, expansion_in, atoms_per_crystal, &
72 primitive_labels(1:4), primitive_coords(1:4,1:3), bp_id)
73
74 ! ZB
75 ELSE IF(crystal_type_in .EQ. 4)THEN
76
77 atoms_per_crystal = 8
78
79 primitive_labels(1) = label_in
80 primitive_labels(2) = label_in
81 primitive_labels(3) = label_in
82 primitive_labels(4) = label_in
83 primitive_labels(5) = label_in
84 primitive_labels(6) = label_in
85 primitive_labels(7) = label_in
86 primitive_labels(8) = label_in
87
88 primitive_coords(1,1) = 0.0D0
89 primitive_coords(1,2) = 0.0D0
90 primitive_coords(1,3) = 0.0D0
91 primitive_coords(2,1) = 0.5D0
92 primitive_coords(2,2) = 0.5D0
93 primitive_coords(2,3) = 0.0D0
94 primitive_coords(3,1) = 0.5D0
95 primitive_coords(3,2) = 0.0D0
96 primitive_coords(3,3) = 0.5D0
97 primitive_coords(4,1) = 0.0D0

```

```

98     primitive_coords(4,2) = 0.5D0
99     primitive_coords(4,3) = 0.5D0
100    primitive_coords(5,1) = 0.25D0
101    primitive_coords(5,2) = 0.25D0
102    primitive_coords(5,3) = 0.25D0
103    primitive_coords(6,1) = 0.75D0
104    primitive_coords(6,2) = 0.75D0
105    primitive_coords(6,3) = 0.25D0
106    primitive_coords(7,1) = 0.25D0
107    primitive_coords(7,2) = 0.75D0
108    primitive_coords(7,3) = 0.75D0
109    primitive_coords(8,1) = 0.75D0
110    primitive_coords(8,2) = 0.25D0
111    primitive_coords(8,3) = 0.75D0
112
113    CALL add_inner(rcut_in, alat_in, uv_in, label_in, crystal_type_in, expansion_in, atoms_per_crystal, &
114                  primitive_labels(1:8), primitive_coords(1:8,1:3), bp_id)
115
116
117 END IF
118
119 END SUBROUTINE add_bp_config
120
121
122
123
124
125 SUBROUTINE add_inner(rcut_in, alat_in, uv_in, label, crystal_type, expansion, atoms_per_crystal, &
126                      primitive_labels, primitive_coords, bp_id)
127 !#####
128 REAL(kind=DoubleReal), INTENT(IN) :: rcut_in
129 REAL(kind=DoubleReal), INTENT(IN) :: alat_in
130 REAL(kind=DoubleReal), INTENT(IN) :: uv_in(1:3,1:3)
131 INTEGER(KIND=StandardInteger), INTENT(IN) :: label
132 INTEGER(KIND=StandardInteger), INTENT(IN) :: crystal_type
133 INTEGER(KIND=StandardInteger), INTENT(IN) :: expansion
134 INTEGER(KIND=StandardInteger), INTENT(IN) :: atoms_per_crystal
135 INTEGER(KIND=StandardInteger), INTENT(IN) :: primitive_labels(:)
136 REAL(kind=DoubleReal), INTENT(IN) :: primitive_coords(:, :)
137 INTEGER(KIND=StandardInteger), INTENT(OUT) :: bp_id
138 !#####
139 INTEGER(KIND=StandardInteger) :: expanded_labels(1:1024)
140 REAL(kind=DoubleReal) :: expanded_coords(1:1024,1:3)
141 REAL(kind=DoubleReal) :: strain(1:3,1:3)
142 REAL(kind=DoubleReal) :: alat_temp
143 REAL(kind=DoubleReal) :: uv_temp(1:3,1:3)
144 REAL(kind=DoubleReal) :: uv_config(1:3,1:3)
145 REAL(kind=DoubleReal) :: vol_cell(1:3,1:3)
146 REAL(kind=DoubleReal) :: vol
147 REAL(kind=DoubleReal) :: sigma
148 INTEGER(KIND=StandardInteger) :: n, i, j, k, m, cn_counter, cc_start, cc_end, ec_n, rn, cc_counter
149 !#####
150
151 ! CHECK
152 n = 1
153 bp_id = 0
154 DO WHILE(bp_keys_i(n, 1) .NE. -1)
155   IF(bp_keys_i(n, 1) .EQ. label .AND. &
156      bp_keys_i(n, 2) .EQ. crystal_type .AND. &
157      bp_keys_i(n, 3) .EQ. expansion .AND. &
158      bp_keys_r(n,1) .EQ. rcut_in .AND. &

```

```

159     bp_keys_r(n,2) .EQ. alat_in) THEN
160
161     ! LOAD ID
162     bp_id = n
163
164 END IF
165 n = n + 1
166 END DO
167 calc_count = max(calc_count, bp_id)
168
169 IF(bp_id .EQ. 0)THEN
170
171     alat_temp = alat_in * maxval(uv_in)
172     uv_temp(1:3,1:3) = uv_in / maxval(uv_in)
173
174     ! Calculate number of eos points
175     eos_points = 2 * eos_size + 1
176
177     !# GET BPID
178     bp_configs_count = bp_configs_count + 1
179     bp_id = bp_configs_count
180
181     !# STORE UNIQUE DETAILS - LABEL, TYPE, EXPANSION, RCUT, ALAT
182     bp_keys_i(bp_id, 1) = label
183     bp_keys_i(bp_id, 2) = crystal_type
184     bp_keys_i(bp_id, 3) = expansion
185     bp_keys_r(bp_id,1) = rcut_in
186     bp_keys_r(bp_id,2) = alat_temp
187
188     !#
189     known_atoms_per_crystal(bp_id) = atoms_per_crystal
190     known_expansion(bp_id) = expansion
191
192     !# CONFIG COUNTER FOR THIS BPID
193     cn_counter = 0
194
195     m = 0
196     DO i=1,expansion
197         DO j=1,expansion
198             DO k=1,expansion
199                 DO n=1,SIZE(primitive_labels, 1)
200                     m = m + 1
201                     expanded_labels(m) = primitive_labels(n)
202                     expanded_coords(m,1) = (i - 1 + primitive_coords(n,1)) / expansion
203                     expanded_coords(m,2) = (j - 1 + primitive_coords(n,2)) / expansion
204                     expanded_coords(m,3) = (k - 1 + primitive_coords(n,3)) / expansion
205             END DO
206         END DO
207     END DO
208
209
210
211 !#####
212 ! EQUATION OF STATE
213 !#####
214
215 cc_start = cc + 1
216 cc_end = cc + eos_points
217 cc_log(bp_id, 1) = cc_start
218 cc_log(bp_id, 2) = cc_end
219
```

```

220
221 cc_counter = 1
222 rn = 1
223 bp_cn(bp_id, rn, 1) = cc_counter
224
225 DO n=1,eos_points
226   cc_counter = cc_counter + 1
227
228   ! Expand input uv
229   uv_config = expansion * uv_temp
230
231   ! Calculate strain
232   sigma = -(eos_size + 1) * (eos_strain / eos_size)
233   strain(:,:) = 0.0D0
234   strain(1,1) = 1.0D0 + sigma
235   strain(2,2) = 1.0D0 + sigma
236   strain(3,3) = 1.0D0 + sigma
237
238   ! Apply strain
239   uv_config = matmul(strain, uv_config)
240
241   ! Calc volume
242   vol_cell(:,:) = alat_temp * uv_config(:,:)
243   vol = TripleProduct(vol_cell(1,:), vol_cell(2,:), vol_cell(3,:))
244
245 CALL add_config(rcut_in, alat_temp, uv_config, expanded_labels(1:m), expanded_coords(1:m,1:3)) ! Increments cc
246   by 1
247 cn_counter = cn_counter + 1
248 bp_configs(bp_id, cn_counter) = cc
249 CALL make_nl(cc)
250
251 !print *, bp_id, n, cc, rcut_in, alat_temp, m
252
253 ! RECORD DATA
254 calc_strains(bp_id, 1, n) = sigma
255 calc_volumes(bp_id, 1, n) = vol/m
256
257 ! REALS
258 bp_data_r(cc, 1) = rcut_in      ! rcut
259 bp_data_r(cc, 2) = alat_temp    ! alat
260 bp_data_r(cc, 3) = sigma        ! strain
261 bp_data_r(cc, 4) = vol          ! vol
262 bp_data_r(cc, 5) = vol/m       ! vol/atom
263 !bp_data_r(cc, 11) =           ! epa
264 !bp_data_r(cc, 12) =           ! epa_eos
265
266 ! INTS
267 bp_data_i(cc, 1) = n            ! eos number
268 bp_data_i(cc, 2) = cc_start     ! bp start
269 bp_data_i(cc, 3) = cc_end       ! bp end
270 bp_data_i(cc, 4) = m            ! atom count
271 !print *, key(cc, 1), key(cc, 2), key(cc, 3), key(cc, 4), key(cc, 5), key(cc, 6)
272
273 total_atoms = total_atoms + m
274 END DO
275 bp_cn(bp_id, rn, 2) = cc_counter - 1
276
277 !#####
278 ! ELASTIC CONSTANTS
279

```

```

280 !#####
281
282
283 ! Calculate number of eos points
284 ec_points = ec_size
285
286 DO ec_n = 1, 9
287   cc_start = cc + 1
288   cc_end = cc + ec_points
289   cc_log(bp_id, 2*ec_n+1) = cc_start
290   cc_log(bp_id, 2*ec_n+2) = cc_end
291
292   rn = rn + 1
293   bp_cn(bp_id, rn, 1) = cc_counter
294
295 DO n=1,ec_points
296   cc_counter = cc_counter + 1
297
298 ! Expand input uv
299 uv_config = expansion * uv_temp
300
301 ! Calculate strain
302 sigma = (n - 1) * (eos_strain / ec_points)
303 strain(:,:) = 0.0D0
304
305 IF(ec_n .EQ. 1)THEN
306   strain(1,1) = 1.0D0 + sigma
307   strain(2,2) = 1.0D0
308   strain(3,3) = 1.0D0
309 ELSE IF (ec_n .EQ. 2)THEN
310   strain(1,1) = 1.0D0
311   strain(2,2) = 1.0D0 + sigma
312   strain(3,3) = 1.0D0
313 ELSE IF (ec_n .EQ. 3)THEN
314   strain(1,1) = 1.0D0
315   strain(2,2) = 1.0D0
316   strain(3,3) = 1.0D0 + sigma
317 ELSE IF (ec_n .EQ. 4)THEN
318   strain(1,1) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
319   strain(2,2) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
320   strain(3,3) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
321   strain(2,3) = sigma / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
322   strain(3,2) = sigma / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
323 ELSE IF (ec_n .EQ. 5)THEN
324   strain(1,1) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
325   strain(2,2) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
326   strain(3,3) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
327   strain(1,3) = sigma / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
328   strain(3,1) = sigma / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
329 ELSE IF (ec_n .EQ. 6)THEN
330   strain(1,1) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
331   strain(2,2) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
332   strain(3,3) = 1.0D0 / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
333   strain(1,2) = sigma / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
334   strain(2,1) = sigma / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
335 ELSE IF (ec_n .EQ. 7)THEN
336   strain(1,1) = (1.0D0 + sigma) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
337   strain(2,2) = (1.0D0 - sigma) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
338   strain(3,3) = (1.0D0) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
339 ELSE IF (ec_n .EQ. 8)THEN
340   strain(1,1) = (1.0D0 + sigma) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))

```

```

341     strain(2,2) = (1.0D0) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
342     strain(3,3) = (1.0D0 - sigma) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
343 ELSE IF (ec_n .EQ. 9)THEN
344     strain(1,1) = (1.0D0) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
345     strain(2,2) = (1.0D0 + sigma) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
346     strain(3,3) = (1.0D0 - sigma) / ((1.0D0 - sigma**2)**(1.0D0/3.0D0))
347 END IF
348
349
350 ! Apply strain
351 uv_config = matmul(strain, uv_config)
352
353 ! Calc volume
354 vol_cell(:,:) = alat_temp * uv_config(:,:)
355 vol = TripleProduct(vol_cell(1,:), vol_cell(2,:), vol_cell(3,:))
356
357 CALL add_config(rcut_in, alat_temp, uv_config, expanded_labels(1:m), expanded_coords(1:m,1:3)) ! Increments
358             cc by 1
359 cn_counter = cn_counter + 1
360 bp_configs(bp_id, cn_counter) = cc
361 CALL make_nl(cc)
362
363 ! RECORD DATA
364 calc_strains(bp_id, ec_n + 1, n) = sigma
365 calc_volumes(bp_id, ec_n + 1, n) = vol/m
366
367 ! RECORD DATA
368 bp_data_r(cc, 1) = rcut_in           ! rcut
369 bp_data_r(cc, 2) = alat_temp         ! alat
370 bp_data_r(cc, 3) = sigma             ! strain
371 bp_data_r(cc, 4) = vol               ! vol
372 bp_data_r(cc, 5) = vol/m            ! vol/atom
373 !bp_data_r(cc, 11) =                 ! epa
374 !bp_data_r(cc, 12) =                 ! epa_eos
375
376 bp_data_i(cc, 1) = n                ! eos number
377 bp_data_i(cc, 2) = cc_start         ! bp start
378 bp_data_i(cc, 3) = cc_end           ! bp end
379 bp_data_i(cc, 4) = m                ! atom count
380 !print *, key(cc, 1), key(cc, 2), key(cc, 3), key(cc, 4), key(cc, 5), key(cc, 6)
381
382 total_atoms = total_atoms + m
383 END DO
384 bp_cn(bp_id, rn, 2) = cc_counter - 1
385
386 END DO
387
388 END IF
389
390 END SUBROUTINE add_inner

```

Q.3 Main fitting class

The fitting class is in Python and is given in listing ???. It loops over the required fitting settings provided by the user and attempts to optimise the parameters for the potentials.

Listing Q.3: Potential fitting - main class (Python)

```
1 #####  
2 # Ben Palmer University of Birmingham 2020  
3 # Free to use  
4 #####  
5  
6 import numpy  
7 from scipy.optimize import minimize  
8 from scipy.optimize import basinhopping  
9 from potential import potential  
10 from gd import gd  
11 from rescale_density import rescale_density  
12 from pot_fit_start import pf_start  
13 from pot_fit_steps import pf_steps  
14 from pot_fit_genetic import pf_genetic  
15 from pot_fit_data import pf_data  
16 from pot_fit_cycle import pf_cycle  
17 from pot_fit_potential import pf_potential  
18 from pot_fit_generation import pf_generation  
19 from pot_fit_extinction import pf_extinction  
20 from pot_fit_enhance import pf_enhance  
21 from pot_fit_parameters import pf_parameters  
22 from pot_fit_setop import pf_setop  
23 from pot_fit_gd import pf_gd  
24 from pot_fit_ng import pf_ng  
25 from pot_fit_sa import pf_sa  
26 from pot_fit_neldermead import pf_nm  
27 from pot_fit_bh import pf_bh  
28 from pot_fit_cg import pf_cg  
29 from pot_fit_bfgs import pf_bfgs  
30 from pot_fit_gsa import pf_gsa  
31 from pot_fit_saa import pf_saa  
32 from pot_fit_sps import pf_sps  
33 from pot_fit_saved import pf_saved  
34 from pot_fit_random import pf_random  
35 from pot_fit_top import pf_top  
36 from pot_fit_pgradient import pf_top  
37 from pot_fit_display import pf_display  
38 from pot_fit_display_simple import pf_display_simple  
39 from pot_fit_init import pf_init  
40 from pot_fit_save import pf_save  
41 from pot_fit_final import pf_final  
42 import matplotlib.pyplot as plt  
43 import time  
44 import random  
45 import hashlib  
46  
47  
48 class pf:  
49  
50     start_time = 0.0  
51     end_time = 0.0  
52     rss_plot_data = []  
53  
54     def run():  
55  
56         # Start Time  
57         pf.start_time = time.time()  
58  
59         pf.pbest_dir = g.dirs['wd'] + '/fitting/saved_potentials/0000_pbest/'  
60         std.make_dir(pf.pbest_dir)  
61
```

```

62     # Log
63     main.log_title("Potential Fit")
64
65     # Set up EFS and BP modules and g.pfdata
66     pf_init.run()
67
68     # Create Plot Dirs
69     std.make_dir(g.dirs['wd'] + '/fitting')
70     pf.fh = open(g.dirs['wd'] + '/fitting/fitting_summary.txt', 'w')
71
72
73     # Run once with initial parameters
74     g.pfdata['stage'] = 'START - INITIAL PARAMETERS'
75     g.pfdata['stage_brief'] = 'START'
76     rss = pf.get_rss()
77
78
79     for fit in g.fit:
80         pf.fit = fit
81         if(pf.fit['type'].lower() == "random"):
82             pf_random.run(fit)
83         elif(pf.fit['type'].lower() == "sa"):
84             pf_sa.run(fit)
85         elif(pf.fit['type'].lower() == "gsa"):
86             pf_gsa.run(fit)
87         elif(pf.fit['type'].lower() == "ga"):
88             pf_genetic.run(fit)
89         elif(pf.fit['type'].lower() == "ng"):
90             pf_ng.run(fit)
91         elif(pf.fit['type'].lower() == "sps"):
92             pf_sps.run(fit)
93         elif(pf.fit['type'].lower() == "nm"):
94             pf_neldermead.run(fit)
95         elif(pf.fit['type'].lower() == "bfgs"):
96             pf_bfgs.run(fit)
97         elif(pf.fit['type'].lower() == "bh"):
98             pf_bh.run(fit)
99         elif(pf.fit['type'].lower() == "cg"):
100            pf_cg.run(fit)
101
102     pf_save.top("BEST_PARAMETERS")
103
104     potential.update(g.pfdata['p']['best'])
105     rss = pf.get_rss()
106
107     pf_top.save(g.dirs['wd'] + '/save', 'top_parameters.txt')
108
109     # Output
110     #potential_output.full()
111
112     # End Time
113     pf.end_time = time.time()
114
115     # Log fit time
116     main.log("Fit time: ", str(pf.end_time - pf.start_time))
117
118     pf_save.rss_plot_full()
119
120 #####
121     # SET UP - initialise EFS, BP and any other modules

```

```

123     #         and run first calc
124 #####
125
126
127 def get_rss(save_in_top=True, quiet=False):
128     rss = rss_calc.run_calc(save_in_top)
129
130     g.pfdata['rss']['current'] = rss
131     g.pfdata['rss']['counter'] += 1
132
133     # If successful
134     if(rss is not None and not quiet):
135
136         # Store Details
137         g.pfdata['bp']['current'] = g.bp_results.copy()
138         g.pfdata['rss']['counter_successful'] += 1
139         g.pfdata['rss']['since_improvement'] += 1
140         if(g.pfdata['rss']['start'] == None):
141             g.pfdata['rss']['start'] = rss
142         if(g.pfdata['rss']['best'] == None or rss < g.pfdata['rss']['best']):
143             g.pfdata['p']['best'] = numpy.copy(potential.pot['p'])
144             g.pfdata['rss']['best'] = rss
145             g.pfdata['bp']['best'] = g.bp_results.copy()
146             g.pfdata['efs']['best'] = g.efs_results.copy()
147             g.pfdata['rss']['since_improvement'] = 0
148
149             potential.save(pf.pbest_dir, g.pfdata['p']['best'])
150             pf_save.save_summary(pf.pbest_dir, g.pfdata['p']['best'], g.pfdata['rss']['best'], g.pfdata['bp']['best'],
151                                 g.pfdata['bp']['known'], g.pfdata['efs']['best'], g.pfdata['efs']['known'])
152             potential.plot(pf.pbest_dir)
153
154         # Save in top
155         if(save_in_top):
156             list_len = g.fitting['top_parameters']
157
158             # COPY ARRAYS
159             p_now = numpy.copy(potential.pot['p'])
160             efs_results = g.efs_results.copy()
161             bp_results = g.bp_results.copy()
162             rss_details = g.rss.copy()
163
164             if(len(g.top_parameters) == 0):
165                 g.top_parameters.append([rss, p_now, efs_results, bp_results, bp.max_density, efs.max_density,
166                                         rss_details])
167             else:
168                 i = 0
169                 while(i<len(g.top_parameters)):
170                     if(rss < g.top_parameters[i][0]):
171                         break
172                     elif(rss == g.top_parameters[i][0] and g.top_parameters[i][1].all() == p_now.all()):
173                         i = list_len
174                         break
175                     else:
176                         i = i + 1
177                 if(i < list_len):
178                     g.top_parameters.insert(i, [rss, p_now, efs_results, bp_results, bp.max_density, efs.max_density,
179                                         rss_details])
180             if(len(g.top_parameters) > list_len):
181                 g.top_parameters.pop()
182
183         # DISPLAY

```

```

181     if(not quiet):
182         pf_display.output()
183 #for i in range(min(len(g.top_parameters),10)):
184 #    print(g.top_parameters[i][0])
185     return rss
186
187
188
189     def summary_line(opt_type, t_start, t_end, rss_start, rss_end):
190
191         t_taken = str('{:6.3f}'.format(t_end - t_start))
192         while(len(t_taken)<16):
193             t_taken = t_taken + " "
194
195         rss_start = str('{:12.4e}'.format(rss_start))
196         while(len(rss_start)<16):
197             rss_start = rss_start + " "
198
199         rss_end = str('{:12.4e}'.format(rss_end))
200         while(len(rss_end)<16):
201             rss_end = rss_end + " "
202
203         opt_type = str(opt_type)
204         while(len(opt_type)<30):
205             opt_type = opt_type + " "
206
207         pf.fh.write(opt_type + rss_start + "--> " + rss_end + " " + t_taken + "\n")
208
209
210     def save_rss_plot_data(t_start, name, data):
211         plot_data = numpy.asarray(data)
212         pf.rss_plot_data.append([t_start, name, data])

```

Q.4 Cubic Splines

Fortran computes the cubic spline in the fnc module and the subroutine for the Heaviside type spline is in listing ??.

Listing Q.4: A cubic spline, using the Heaviside step function to cutoff each summed polynomial

```

1 ! Two Band Modelling Fe-Cr Olsson, Wallenius
2 ! CUBIC SPLINE
3 ! sum (ai (r - ri)^3 H(ri - r)
4 ! SCALAR SUBROUTINE
5 SUBROUTINE cubic_spline(r, p, p_fixed, y)
6 !#####
7 ! p coefficients
8 ! pf r cutoffs
9 ! they must be the same size
10 IMPLICIT NONE
11 !#####
12 REAL(kind=DoubleReal), INTENT(IN) :: r
13 REAL(kind=DoubleReal), INTENT(IN) :: p(:)
14 REAL(kind=DoubleReal), INTENT(IN) :: p_fixed(:)
15 REAL(kind=DoubleReal), INTENT(OUT) :: y
16 !#####
17 REAL(kind=DoubleReal) :: H
18 INTEGER(kind=StandardInteger) :: n
19 !#####

```

```

20   y = 0.0DO
21   DO n = 1, SIZE(p,1)
22     ! Heaviside((p_fixed(n) - r)
23     IF((p_fixed(n) - r) < 0.0DO)THEN
24       H = 0.0DO
25     ELSE
26       H = 1.0DO
27     END IF
28     y = y + p(n) * (r - p_fixed(n))**3 * H
29   END DO
30   IF(SIZE(p,1) + 1 .EQ. SIZE(p_fixed,1))THEN
31     IF(r .GE. p_fixed(SIZE(p_fixed,1)))THEN
32       y = 0.0DO
33     ELSE
34       y = y * (r - p_fixed(SIZE(p_fixed,1)))**3
35     END IF
36   END IF
37   END SUBROUTINE cubic_spline
38
39 ! VECTOR SUBROUTINE
40 SUBROUTINE cubic_spline_v(r, p, p_fixed, y)
41 !#####
42 ! CUBIC SPLINE
43 IMPLICIT NONE
44 !#####
45 REAL(kind=DoubleReal), INTENT(IN) :: r(:)
46 REAL(kind=DoubleReal), INTENT(IN) :: p(:)
47 REAL(kind=DoubleReal), INTENT(IN) :: p_fixed(:)
48 REAL(kind=DoubleReal), INTENT(OUT) :: y(1:SIZE(r,1))
49 INTEGER(kind=StandardInteger) :: n
50 !#####
51 ! Loop through all the values in r(:), calculate and store in y(:)
52 DO n = 1, SIZE(r,1)
53   CALL cubic_spline(r(n), p, p_fixed, y(n))
54 END DO
55 END SUBROUTINE cubic_spline_v

```

Q.5 Spline Subroutine for the Cubic Knot Spline

The cubic knot spline is also computed in Fortran (listing ??) and uses the solve linear set (sls_solve) subroutine.

Listing Q.5: A cubic knot spline that could be extended to a quintic.

```

1 SUBROUTINE spline_ab(spline_type, node_a, node_b, coeffs)
2 !#####
3 IMPLICIT NONE
4 !#####
5 INTEGER(kind=StandardInteger), INTENT(IN) :: spline_type
6 REAL(kind=DoubleReal), INTENT(IN) :: node_a(:)
7 REAL(kind=DoubleReal), INTENT(IN) :: node_b(:)
8 REAL(kind=DoubleReal), INTENT(OUT) :: coeffs(1:SIZE(node_a) + SIZE(node_b) - 2)
9 !#####
10 coeffs = 0.0DO
11 ! POLY3
12 ! Each node/knot has 3 values x, y, y'
13 IF(spline_type .EQ. 1 .AND. SIZE(node_a) .EQ. 3 .AND. SIZE(node_b) .EQ. 3)THEN
14   CALL spline_ab_poly(node_a, node_b, coeffs)
15 END IF
16 ! POLY5
17 ! Each node/knot has 4 values x, y, y', y''
```

```

18 IF(spline_type .EQ. 2 .AND. SIZE(node_a) .EQ. 4 .AND. SIZE(node_b) .EQ. 4)THEN
19   CALL spline_ab_poly(node_a, node_b, coeffs)
20 END IF
21 END SUBROUTINE spline_ab
22
23
24 SUBROUTINE spline_ab_poly(node_a, node_b, coeffs)
25 !#####
26 IMPLICIT NONE
27 !#####
28 REAL(kind=DoubleReal), DIMENSION(:, :, :), INTENT(IN) :: node_a
29 REAL(kind=DoubleReal), DIMENSION(:, :, :), INTENT(IN) :: node_b
30 REAL(kind=DoubleReal), INTENT(OUT) :: coeffs(1:SIZE(node_a) + SIZE(node_b) - 2)
31 !#####
32 INTEGER(kind=StandardInteger) :: m_size = 0
33 INTEGER(kind=StandardInteger) :: m_half_size = 0
34 INTEGER(kind=StandardInteger) :: d_loop = 0
35 REAL(kind=DoubleReal) :: x(1:SIZE(node_a,1)+SIZE(node_b,1)-2,1:SIZE(node_a,1)+SIZE(node_b,1)-2)
36 REAL(kind=DoubleReal) :: y(1:SIZE(node_a,1)+SIZE(node_b,1)-2)
37 REAL(kind=DoubleReal) :: x_coeffs(1:(SIZE(node_a,1)+SIZE(node_b,1)-2))
38 REAL(kind=DoubleReal) :: x_exponent(1:(SIZE(node_a,1)+SIZE(node_b,1)-2))
39 INTEGER(kind=StandardInteger) :: n, row, col, a_row
40 !#####
41 ! Sizes
42 m_size = SIZE(node_a) + SIZE(node_b) - 2
43 m_half_size = m_size / 2
44 ! x exponents
45 DO n = 1, m_size
46   x_exponent(n) = 1.0D0 * (n - 1)
47 END DO
48 ! Coeffs
49 x_coeffs = 1
50 ! Make y matrix
51 row = 1
52 DO n=2,SIZE(node_a)
53   y(row) = node_a(n)
54   row = row + 1
55   y(row) = node_b(n)
56   row = row + 1
57 END DO
58 ! Make x matrix
59 row = 0
60 DO a_row = 1, m_half_size
61   ! NODE A
62   row = row + 1
63   DO col = 1, a_row -1
64     x(row, col) = 0.0D0
65   END DO
66   DO col = a_row, m_size
67     x(row, col) = x_coeffs(col) * node_a(1)**x_exponent(col)
68   END DO
69   ! NODE B
70   row = row + 1
71   DO col = 1, a_row -1
72     x(row, col) = 0.0D0
73   END DO
74   DO col = a_row, m_size
75     x(row, col) = x_coeffs(col) * node_b(1)**x_exponent(col)
76   END DO
77   ! UPDATE COEFFS AND EXPONENT
78   DO col = 1, m_size

```

```

79      x_coeffs(col) = x_coeffs(col) * x_exponent(col)
80  END DO
81  DO col = 1, m_size
82    IF(x_exponent(col) .GT. 0.0D0)THEN
83      x_exponent(col) = x_exponent(col) - 1.0D0
84    END IF
85  END DO
86 END DO
87 ! SOLVE
88 CALL s1s_solve(x, y, coeffs)
89 !#####
90 END SUBROUTINE spline_ab_poly

```

Q.6 ZBL with Cubic Knot Spline

A further set of subroutines are used to compute the cubic spline summed to the ZBL function (listing ??).

Listing Q.6: ZBL with cubic knot spline, variable x, y(x) and y'(x) parameters

```

1 SUBROUTINE cubic_knot_spline_5(r, p, p_fixed, y)
2 !#####
3 ! p coefficients
4 ! pf r cutoffs
5 ! they must be the same size
6 IMPLICIT NONE
7 !#####
8 REAL(kind=DoubleReal), INTENT(IN) :: r
9 REAL(kind=DoubleReal), INTENT(IN) :: p(:)
10 REAL(kind=DoubleReal), INTENT(IN) :: p_fixed(:)
11 REAL(kind=DoubleReal), INTENT(OUT) :: y
12 !#####
13 INTEGER(kind=StandardInteger) :: n
14 REAL(kind=DoubleReal) :: nodes(1:(size(p,1)-2) / 3 + 2,1:4)
15 REAL(kind=DoubleReal) :: qa, qb, zbl_mult
16 INTEGER(kind=StandardInteger) :: psize, pfszie, node_count, pstride
17 LOGICAL :: complete
18 !#####
19 psize = SIZE(p, 1)
20 pfszie = SIZE(p_fixed, 1)
21 pstride = (size(p,1)-2) / 3
22 node_count = pstride + 2
23 nodes(:, :) = 0.0D0
24 nodes(2:node_count-1, 1) = p(1:pstride)
25 nodes(2:node_count-1, 2) = p(pstride+1:2*pstride)
26 nodes(2:node_count-1, 3) = p(2*pstride+1:3*pstride)
27
28 qa = p_fixed(1)
29 qb = p_fixed(2)
30 zbl_mult = p_fixed(3)
31 nodes(1, 1) = p_fixed(4)
32 nodes(1, 2) = p(3*pstride+1)
33 nodes(1, 3) = p(3*pstride+2)
34 nodes(node_count, 1) = p_fixed(5)
35 nodes(node_count, 2) = p_fixed(6)
36 nodes(node_count, 3) = p_fixed(7)
37
38 IF(r .LT. nodes(1, 1))THEN
39   y = nodes(1, 2)
40 ELSE IF(r .GT. nodes(node_count, 1))THEN
41   y = nodes(node_count, 2)

```

```

42 ELSE
43 ! Add ZBL
44 y = y + 1.0D0 * f_zbl_ackson_mendelev(r, qa, qb)
45
46 ! IF EXACTLY A NODE
47 complete = .FALSE.
48 DO n = 1, node_count
49   IF(r .EQ. nodes(n,1))THEN
50     complete = .TRUE.
51     y = y + nodes(n,2)
52   END IF
53 END DO
54
55 IF(complete .EQV. .FALSE.)THEN
56   y = y + f_cubic_knot_spline(r, nodes)
57 END IF
58 END IF
59
60
61 END SUBROUTINE cubic_knot_spline_5
62
63
64 ! VECTOR SUBROUTINE
65 SUBROUTINE cubic_knot_spline_5_v(r, p, p_fixed, y)
66 !#####
67 ! PAIR SPLINE
68 IMPLICIT NONE
69 !#####
70 REAL(kind=DoubleReal), INTENT(IN) :: r(:)
71 REAL(kind=DoubleReal), INTENT(IN) :: p(:)
72 REAL(kind=DoubleReal), INTENT(IN) :: p_fixed(:)
73 REAL(kind=DoubleReal), INTENT(OUT) :: y(1:SIZE(r,1))
74 INTEGER(kind=StandardInteger) :: n
75 !#####
76 ! Loop through all the values in r(:), calculate and store in y(:)
77 DO n = 1, SIZE(r,1)
78   CALL cubic_knot_spline_5(r(n), p, p_fixed, y(n))
79 END DO
80 END SUBROUTINE cubic_knot_spline_5_v
81
82
83
84 FUNCTION f_cubic_knot_spline(r, nodes) RESULT (y)
85 !#####
86 REAL(kind=DoubleReal) :: r
87 REAL(kind=DoubleReal) :: nodes(:, :)
88 REAL(kind=DoubleReal) :: y
89 !#####
90 INTEGER(kind=StandardInteger) :: n, j
91 !#####
92
93 ! FIND n, n+1 TO SPLINE BETWEEN
94 IF(r .LT. nodes(1,1))THEN
95   n = 1
96 ELSE IF(r .GT. nodes(SIZE(nodes,1), 1))THEN
97   n = SIZE(nodes,1) - 1
98 ELSE
99   DO n = 1, SIZE(nodes,1) - 1
100     IF(r .GE. nodes(n, 1) .AND. r .LT. nodes(n+1, 1))THEN
101       EXIT
102     END IF

```

```

103     END DO
104   END IF
105   y = f_cubic_knot_ab(r, nodes(n, 1), nodes(n, 2), nodes(n, 3), nodes(n+1, 1), nodes(n+1, 2), nodes(n+1, 3))
106 END FUNCTION f_cubic_knot_spline
107
108
109 FUNCTION f_cubic_knot_ab(r, xa, ya, ypa, xb, yb, ypb) RESULT (y)
110 ! CUBIC KNOT
111 IMPLICIT NONE
112 !#####
113 REAL(kind=DoubleReal) :: r
114 REAL(kind=DoubleReal) :: xa, ya, ypa, xb, yb, ypb
115 REAL(kind=DoubleReal) :: y
116 !#####
117 REAL(kind=DoubleReal) :: xmat(1:4,1:4)
118 REAL(kind=DoubleReal) :: xmat_inv(1:4,1:4)
119 REAL(kind=DoubleReal) :: ymat(1:4)
120 REAL(kind=DoubleReal) :: c(1:4)
121 !#####
122 xmat(1,1) = 1.0
123 xmat(1,2) = xa
124 xmat(1,3) = xa**2
125 xmat(1,4) = xa**3
126 xmat(2,1) = 1.0
127 xmat(2,2) = xb
128 xmat(2,3) = xb**2
129 xmat(2,4) = xb**3
130 xmat(3,1) = 0.0
131 xmat(3,2) = 1.0D0
132 xmat(3,3) = 2.0D0 * xa
133 xmat(3,4) = 3.0D0 * xa**2
134 xmat(4,1) = 0.0
135 xmat(4,2) = 1.0D0
136 xmat(4,3) = 2.0D0 * xb
137 xmat(4,4) = 3.0D0 * xb**2
138
139 ymat(1) = ya
140 ymat(2) = yb
141 ymat(3) = ypa
142 ymat(4) = ypb
143
144 !CALL sls_solve(xmat, ymat, c)
145 CALL minverse(xmat, xmat_inv, 4)
146 c = MATMUL(xmat_inv, ymat)
147 y = c(1) + c(2) * r + c(3) * r**2 + c(4) * r**3
148
149 END FUNCTION f_cubic_knot_ab
150
151
152 FUNCTION f_zbl_ackland_mendelev(r, q1, q2) RESULT (y)
153 !#####
154 ! ZBL Development of an interatomic potential for phosphorus impurities in -iron 2004
155 IMPLICIT NONE
156 !#####
157 REAL(kind=DoubleReal) :: q1, q2, r
158 REAL(kind=DoubleReal) :: y
159 !#####
160 REAL(kind=DoubleReal) :: rs, x, e
161 !#####
162 rs = 0.4683766D0 / (q1**((2.0D0/3.0D0) + q2**((2.0D0/3.0D0)))
163 x = (r / rs)

```

```

164 e = 0.1818D0 * exp(-3.2 * x) + 0.5099D0 * exp(-0.9423 * x) + 0.2802D0 * exp(-0.4029 * x) + 0.02817D0 * exp
165 (-0.2016 * x)
166 y = (q1 * q2 * e) / r
END FUNCTION f_zbl_ackland_mendelev

```

Q.7 Simulated Annealing

The simulated annealing algorithm is used in Python and is one of the global optimization algorithms (listing 5.3).

Listing Q.7: Simulated Annealing global optimization function

```

1 """
2 Simple simulated annealing
3 No direction
4 Temperature loop from start to end temperature
5 Step size decreases with temperature
6
7 """
8
9 class pf_sa:
10
11     t = None
12     count = 0
13
14     def run(fit):
15         for repeat in range(fit['repeat']):
16             pf_sa.run_sa(fit)
17
18     def run_sa(fit):
19         pf_sa.count = pf_sa.count + 1
20         t_start = time.time()
21         start_best_rss = g.pfdata['rss']['best']
22
23         if(pf.fit['sa_loops_t'] == 0 or pf.fit['sa_loops_i'] == 0):
24             return 0
25
26         g.pfdata['stage'] = 'Simulated Annealing ' + str(pf_sa.count)
27         g.pfdata['stage_brief'] = 'SA' + str(pf_sa.count)
28
29         # Start - use best
30         p = g.pfdata['p']['best']
31         potential.update(p)
32         rss = pf.get_rss(False)
33         p_count = len(p)
34
35         # Store Best Parameters
36         p_best = numpy.copy(p)
37         rss_best = rss
38
39         rss_plot = []
40         rss_plot.append([time.time()-t_start, rss_best])
41
42         for tn in range(pf.fit['sa_loops_t']):
43             pv = pf.fit['sa_var'] * pf.fit['sa_var_factor']**tn
44             if(pf.fit['sa_loops_t'] == 1):
45                 pf_sa.t = pf.fit['sa_temp_start']
46             else:

```

```

48     pf_sa.t = numpy.round(pf.fit['sa_temp_start'] - tn * (pf.fit['sa_temp_start'] - pf.fit['sa_temp_end']) / (
49         pf.fit['sa_loops_t'] - 1), 6)
50     g.pfdata['stage'] = 'Simulated Annealing ' + str(pf_sa.count) + ' Loop ' + str(tn + 1) + ' Temp: ' + str(
51         pf_sa.t) + ' Pvar: ' + str(pv)
52     rss_start_loop = rss_best
53     for n in range(pf.fit['sa_loops_i']):
54         loop = True
55         while(loop):
56             p_new = potential.random(p, pv, False)
57             rss_new = pf.get_rss(False)
58             if(rss_new is not None):
59                 loop = False
60             if(rss_new is not None):
61                 if(rss_new < rss or numpy.random.uniform() < numpy.exp((rss-rss_new) / pf_sa.t)):
62                     p = numpy.copy(p_new)
63                     rss = rss_new
64                     if(rss_new < rss_best):
65                         p_best = numpy.copy(p_new)
66                         rss_best = rss_new
67                         rss_plot.append([time.time()-t_start, rss_best])
68                     if(rss_best < rss_start_loop):
69                         potential.update(p_best)
70                         pf.get_rss(True)
71                         p = numpy.copy(p_best)
72                         rss_plot.append([time.time()-t_start, rss_best])
73
74             pf.summary_line("SIMULATED ANNEALING " + str(pf_sa.count), t_start, time.time(), start_best_rss, g.pfdata['rss
75                 '][])
76             pf_save.top("SIMULATED_ANNEALING_" + str(pf_sa.count))
77             pf_save.rss_plot("SIMULATED_ANNEALING_" + str(pf_sa.count), rss_plot)
78             pf.save_rss_plot_data(t_start, "SIMULATED_ANNEALING_" + str(pf_sa.count), rss_plot)

```

Q.8 Genetic Algorithm

Another global optimization algorithm used in the Python section of the program is the genetic algorithm (listing ??).

Listing Q.8: Genetic Algorithm global optimization function

```

1 #####
2 # Ben Palmer University of Birmingham 2020
3 # Free to use
4 #####
5
6 import numpy
7 from potential import potential
8 from display import display
9 from gd import gd
10 from rescale_density import rescale_density
11 from pot_fit_data import pf_data
12 from pot_fit_cycle import pf_cycle
13 from pot_fit_potential import pf_potential
14 from pot_fit_extinction import pf_extinction
15 from pot_fit_enhance import pf_enhance
16 from pot_fit_parameters import pf_parameters
17 import matplotlib.pyplot as plt
18 import time
19 import random

```

```

20 import hashlib
21
22
23 class pf_genetic:
24
25     count = 0
26     start_time = 0
27     variation_factor = 1.0
28
29     def run(fit):
30         if(fit['gens'] == 0):
31             return 0
32
33         #
34         t_start = time.time()
35         start_best_rss = g.pfdata['rss']['best']
36         pf_genetic.start_time = time.time()
37         pf_genetic.count = pf_genetic.count + 1
38         pf_genetic.rss_plot = []
39         pf_genetic.rss_plot.append([time.time() - pf_genetic.start_time, start_best_rss])
40         pf_genetic.rss_best = start_best_rss
41
42         g.pfdata['stage'] = 'Genetic Algorithm ' + str(pf_genetic.count)
43         g.pfdata['stage_brief'] = 'GA' + str(pf_genetic.count)
44
45         main.log_title("Genetic Fit")
46
47         # Load from input
48         pf_genetic.width = g.pfdata['psize']
49         pf_genetic.pop_size = fit['pop_size']
50         pf_genetic.fresh_size = fit['fresh_size']
51         pf_genetic.generations = fit['gens']
52         pf_genetic.no_clone_var = fit['no_clone_var']
53         pf_genetic.gen_variation_multiplier = fit['gen_variation_multiplier']
54
55
56
57         # Force to be even
58         if(pf_genetic.pop_size % 2 != 0):
59             pf_genetic.pop_size = pf_genetic.pop_size + 1
60         if(pf_genetic.fresh_size % 2 != 0):
61             pf_genetic.fresh_size = pf_genetic.fresh_size + 1
62
63         # Calculate other sizes
64         pf_genetic.children_size = pf_genetic.pop_size + 2 * pf_genetic.fresh_size
65         pf_genetic.merge_size = 2 * pf_genetic.pop_size + 3 * pf_genetic.fresh_size
66
67         # Set Up Pop + Fresh arrays
68         pf_genetic.pop = numpy.zeros((pf_genetic.pop_size, pf_genetic.width,),)
69         pf_genetic.pop_rss = numpy.zeros((pf_genetic.pop_size,),)
70         pf_genetic.fresh = numpy.zeros((pf_genetic.fresh_size, pf_genetic.width,),)
71         pf_genetic.fresh_rss = numpy.zeros((pf_genetic.fresh_size,),)
72         pf_genetic.children = numpy.zeros((pf_genetic.children_size, pf_genetic.width,),)
73         pf_genetic.children_rss = numpy.zeros((pf_genetic.children_size,),)
74         pf_genetic.pop_merged = numpy.zeros((pf_genetic.merge_size, pf_genetic.width,),)
75         pf_genetic.pop_merged_rss = numpy.zeros((pf_genetic.merge_size,),)
76
77         # Parents array
78         pf_genetic.parents = numpy.arange(pf_genetic.pop_size)
79
80         # Initialise population

```

```

81 g.pfdata['stage'] = 'Genetic Fit ' + str(pf_genetic.count) + ' Initialising'
82
83 # Copy out parameters from "top_parameters"
84 start_parameters = []
85 if(len(g.top_parameters) > 0):
86     for i in range(len(g.top_parameters)):
87         new_p = numpy.copy(g.top_parameters[i][1][:])
88         start_parameters.append(new_p)
89 else:
90     p = potential.get_start_parameters()
91     new_p = numpy.copy(p)
92     start_parameters.append(new_p)
93
94 # Load start parameters into pop, and fill remaining with random parameters
95 n = 0
96 for pn in range(pf_genetic.pop_size):
97     loop = True
98     while(loop):
99         n = n + 1
100        if(n <= len(start_parameters)):
101            pf_genetic.pop[pn, :] = start_parameters[n-1] [:]
102        else:
103            pf_genetic.pop[pn, :] = potential.random(g.pfdata['p']['best'], 1.0, False)
104        # Try - if it fails or rss == None, try next
105        potential.update(pf_genetic.pop[pn, :])
106        try:
107            # Update
108            rss = pf.get_rss()
109            if(rss is not None):
110                loop = False
111                pf_genetic.pop_rss[pn] = rss
112            except:
113                pass
114            if(rss < pf_genetic.rss_best):
115                pf_genetic.rss_best = rss
116                pf_genetic.rss_plot.append([time.time() - pf_genetic.start_time, pf_genetic.rss_best])
117
118 ######
119 # LOOP THROUGH GENERATIONS
120 #####
121
122 pf_genetic.generation = 0
123 for gen in range(pf_genetic.generations):
124     pf_genetic.generation = pf_genetic.generation + 1
125     pf_genetic.run_gen()
126
127 pf_genetic.rss_plot.append([time.time() - pf_genetic.start_time, pf_genetic.rss_best])
128
129 # End
130 pf.summary_line("GENETIC ALGORITHM " + str(pf_genetic.count), t_start, time.time(), start_best_rss, g.pfdata['rss']['best'])
131 pf_save.top("GENETIC_ALGORITHM_" + str(pf_genetic.count))
132 pf_save.rss_plot("GENETIC_ALGORITHM_" + str(pf_genetic.count), pf_genetic.rss_plot)
133 pf.save_rss_plot_data(t_start, "GENETIC_ALGORITHM_" + str(pf_genetic.count), pf_genetic.rss_plot)
134
135
136
137 def run_gen():
138
139     ss = 'Genetic Fit ' + str(pf_genetic.count) + ' Gen ' + str(pf_genetic.generation) + ' '

```

```

141 # Shuffle parents array
142 numpy.random.shuffle(pf_genetic.parents)
143
144 c = 0
145 # Breed Population
146 g.pfdata['stage'] = ss + 'Breed Population'
147 for p in range(pf_genetic.pop_size // 2):
148     loop = True
149     while(loop):
150         loop = pf_genetic.breed_event(p, c, 'p+p')
151         c = c + 2
152
153 # Make Fresh
154 g.pfdata['stage'] = ss + 'Initialising Fresh Population'
155 pf_genetic.make_fresh()
156
157 # Shuffle parents array
158 numpy.random.shuffle(pf_genetic.parents)
159
160 # Breed Population-Fresh
161 g.pfdata['stage'] = ss + 'Breed With Fresh Population'
162 for p in range(pf_genetic.fresh_size):
163     loop = True
164     while(loop):
165         loop = pf_genetic.breed_event(p, c, 'p+f')
166         c = c + 2
167
168 # Merge populations and select top to form next generation
169 g.pfdata['stage'] = ss + 'Merge Populations'
170 pf_genetic.merge()
171
172
173 # Load best back into population
174 pf_genetic.pop[:, :] = pf_genetic.pop_merged[0:pf_genetic.pop_size, :]
175 pf_genetic.pop_rss[:] = pf_genetic.pop_merged_rss[0:pf_genetic.pop_size]
176
177
178 # Change variation for next generation
179 pf_genetic.variation_factor = pf_genetic.variation_factor * pf_genetic.gen_variation_multiplier
180
181
182
183
184 def breed_event(p, c, opt):
185     pc = pf_genetic.width
186
187     pa = pf_genetic.parents[p]
188     if(opt == 'p+p'):
189         pb = pf_genetic.parents[p + pf_genetic.pop_size // 2]
190         pb_array = 'pop'
191     if(opt == 'p+f'):
192         pb = p
193         pb_array = 'fresh'
194
195     ca = c
196     cb = c + 1
197
198     # Breed
199     state = pf_genetic.get_state()
200     for i in range(pc):

```

```

202     state = pf_genetic.get_state(state)
203     if(state):
204         pf_genetic.children[ca, i] = pf_genetic.pop[pa, i]
205         if(pb_array == 'pop'):
206             pf_genetic.children[cb, i] = pf_genetic.pop[pb, i]
207         elif(pb_array == 'fresh'):
208             pf_genetic.children[cb, i] = pf_genetic.fresh[pb, i]
209     else:
210         if(pb_array == 'pop'):
211             pf_genetic.children[ca, i] = pf_genetic.pop[pb, i]
212         elif(pb_array == 'fresh'):
213             pf_genetic.children[ca, i] = pf_genetic.fresh[pb, i]
214     pf_genetic.children[cb, i] = pf_genetic.pop[pa, i]
215
216
217 # No clones - Child A
218 for pn in range(pf_genetic.pop_size):
219     if((pf_genetic.children[ca, :].all() == pf_genetic.pop[pn,:]).all()):
220         r = numpy.random.rand(pc)
221         pf_genetic.children[ca, :] = pf_parameters(pf_genetic.children[ca, :], pf_genetic.no_clone_var)
222         break
223
224 # No clones - Child B
225 for pn in range(pf_genetic.pop_size):
226     if((pf_genetic.children[cb, :].all() == pf_genetic.pop[pn,:]).all()):
227         r = numpy.random.rand(pc)
228         pf_genetic.children[cb, :] = pf_parameters(pf_genetic.children[cb, :], pf_genetic.no_clone_var)
229         break
230
231 # Check the children actually give valid parameters
232 loop = False
233 potential.update(pf_genetic.children[ca, :])
234 rss_a = pf.get_rss()
235 if(rss_a is None):
236     loop = True
237
238 potential.update(pf_genetic.children[cb, :])
239 rss_b = pf.get_rss()
240 if(rss_b is None):
241     loop = True
242
243 if(loop == False):
244     pf_genetic.children_rss[ca] = rss_a
245     pf_genetic.children_rss[cb] = rss_b
246
247 if(rss_a != None and rss_b != None):
248     if(rss_a < pf_genetic.rss_best):
249         pf_genetic.rss_best = rss_a
250         pf_genetic.rss_plot.append([time.time() - pf_genetic.start_time, pf_genetic.rss_best])
251     if(rss_b < pf_genetic.rss_best):
252         pf_genetic.rss_best = rss_b
253         pf_genetic.rss_plot.append([time.time() - pf_genetic.start_time, pf_genetic.rss_best])
254
255
256 # Loop again or not
257 return loop
258
259
260
261 def get_state(state = None):
262     if(state == None):

```

```

263     if(random.uniform(0.0, 100.0) > 50.0):
264         return True
265     return False
266 else:
267     if(random.uniform(0.0, 100.0) > 50.0):
268         if(state):
269             return False
270         return True
271     return state
272
273
274
275 def make_fresh():
276     for pn in range(pf_genetic.fresh_size):
277         loop = True
278         pbest = numpy.copy(g.top_parameters[0][1])
279         while(loop):
280             try:
281                 rn = int(min(len(g.top_parameters) - 1, numpy.floor((len(g.top_parameters) + 1) * random.uniform(0.0,
282                                         1.0)**3)))
283                 p_trial = numpy.copy(g.top_parameters[rn][1])
284                 p_new = potential.random(p_trial, pf_genetic.variation_factor, False)
285                 rss = pf.get_rss()
286                 if(rss is not None):
287                     loop = False
288             except:
289                 pass
290             pf_genetic.fresh[pn, :] = p_new
291             pf_genetic.fresh_rss[pn] = rss
292
293             if(rss < pf_genetic.rss_best):
294                 pf_genetic.rss_best = rss
295                 pf_genetic.rss_plot.append([time.time() - pf_genetic.start_time, pf_genetic.rss_best])
296
297 # Used to merge parents, fresh and all children into ordered array
298 def merge():
299     ps = pf_genetic.pop_size
300     fs = pf_genetic.fresh_size
301     cs = pf_genetic.pop_size + 2 * pf_genetic.fresh_size
302
303     # Reset array
304     pf_genetic.pop_merged[:, :] = 0.0
305     pf_genetic.pop_merged[0:ps, :] = pf_genetic.pop[:, :]
306     pf_genetic.pop_merged[ps:ps+fs, :] = pf_genetic.fresh[:, :]
307     pf_genetic.pop_merged[ps+fs:ps+fs+cs, :] = pf_genetic.children[:, :]
308     pf_genetic.pop_merged_rss[:] = 0.0
309     pf_genetic.pop_merged_rss[0:ps] = pf_genetic.pop_rss[:]
310     pf_genetic.pop_merged_rss[ps:ps+fs] = pf_genetic.fresh_rss[:]
311     pf_genetic.pop_merged_rss[ps+fs:ps+fs+cs] = pf_genetic.children_rss[:]
312
313     # Sort
314     pf_genetic.sort_merged()
315
316 def sort_merged():
317     sort.sort_1d_dp_asc(pf_genetic.pop_merged_rss[:])
318     pf_genetic.pop_merged_rss = sort.apply_keytable_1d_dp(pf_genetic.pop_merged_rss)
319     pf_genetic.pop_merged = sort.apply_keytable_2d_dp(pf_genetic.pop_merged)

```

Q.9 Gradient Descent

Gradient descent using a line search is one option to refine the genetic algorithm (listing ??), but on reflection it may have been better to use the algorithms in SciPy to enhance members of the genetic algorithm population.

Listing Q.9: Gradient descent pseudocode

```
1 // Simulated Annealing
2 subroutine simulated_annealing(f, p, p_best)
3
4 // f is the function being optimised
5 // p is an array of the parameters
6
7 p_search = True
8 rss_best = calc_rss(f, p)
9
10 do while(p_search)
11
12     // Some function to calculate the
13     df(:) = gradient(f,p(:))
14
15     // Back track line search
16     alpha = 1.0
17     alpha_search = True
18     last_rss = -1.0
19     do while(alpha_search)
20         rss = calc_rss(f, p(:) - alpha * df(:))
21
22         // If it's the first step, store rss into last_rss
23         if(last_rss == -1.0)then
24             last_rss = rss
25         end if
26
27         // If the new one is worse, stop searching and step alpha back
28         if(rss > ls_rss)then
29             alpha_search = False
30             alpha = 2.0 * alpha
31             // If it's better, reduce alpha and keep searching
32             else
33                 alpha = 0.5 * alpha
34                 last_rss = rss
35             end if
36         end do
37
38     // Check new parameters
39     rss = calc_rss(f, p(:) - alpha * df(:))
40
41     // If it's better, save and keep searching
42     if(rss < rss_best)then
43         p(:) = p(:) - alpha * df(:)
44         rss_best = rss
45
46     // Break out
47     else
48         p_search = False
49     end if
50 end do
51
52
53 end subroutine
```

Q.10 Interpolation

Lagrange polynomial interpolation was programmed in Fortran and was used in a number of places throughout the program (listing ??).

Listing Q.10: Lagrange polynomial interpolation

```
1 // lagrange polynomial interpolation
2 function lpinterp_y(x, data)
3 // x - point value being interpolated at to calc f(x)
4 // data - four x,y data points (n by 2 array)
5 n = len(data, 1)
6 y = 0.0
7 for i = 1,n
8     l = 1.0
9     for j = 1,n
10        if (i != j) then
11            l = l * (x - data[j][1]) / (data[i][1] - data[j][1])
12        end if
13    next j
14    y = y + l * data[i][2]
15 next i
16 end function lpinterp_y
```

Q.11 Interpolation (Gradient)

A modified version of the Lagrange polynomial interpolation was programmed in Fortran to interpolate the derivative of a set of data points at a given position between (and including) the start and end point (listing ??).

Listing Q.11: Lagrange polynomial interpolation (Gradients)

```
1 function lpinterp_dydx(x, data)
2 // x - point value being interpolated at fo calc f'(x)
3 // data - four x,y data points (n by 2 array)
4 n = len(data, 1)
5 dydx = 0.0
6 for i = 1,n
7     fx = 1.0
8     gx = 1.0
9     for j = 1, n
10        if (i != j) then
11            fx = fx / (data[i][1] - data[j][1])
12            psum = 1.0
13            for k = 1, n
14                if (i != k and j != k) then
15                    psum = psum * (x - data[k][1])
16                end if
17            next k
18            gx = gx + psum
19        end if
20    next j
21    dydx = dydx + fx * gx * data[i][2]
22 next i
23 end function lpinterp_dydx
```

Appendix R

Potential Functions

R.1 Introduction

This section of the appendix covers the types of potential function and common choices of function used and details on how to use these in the potential fitting code.

R.2 Functions

R.2.1 Ackland-Mendelev Pair

$$v(r) = \sum_{i=1}^N (a_i(r - r_i)^3 H(r_i - r)) \quad (\text{R.1})$$

Listing R.1: Ackland Mendelev Pair

```
1 #TYPE ackland_mendelev_pair
2 #P 1.0
3 #PF 0.0
```

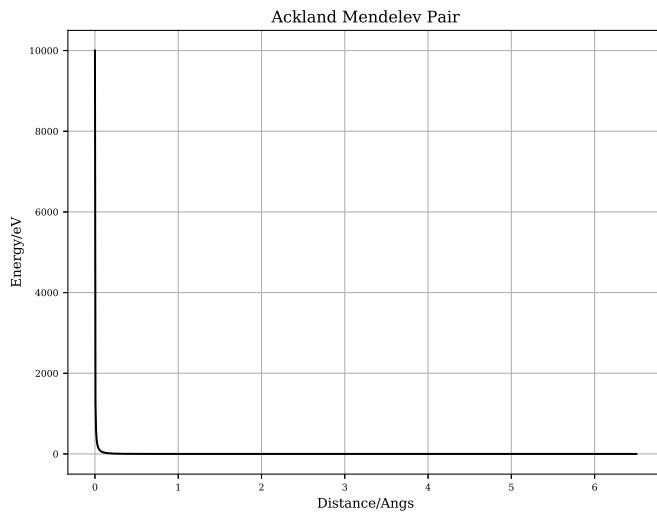


Figure R.1: Ackland Mendelev Pair

R.2.2 Buckingham

$$V(r) = A \times \exp(-B \times r) - \frac{C}{r^6} \quad (\text{R.2})$$

Listing R.2: Buckingham

```
1 #TYPE buckingham
2 #P 6.0 0.5 12.0
```

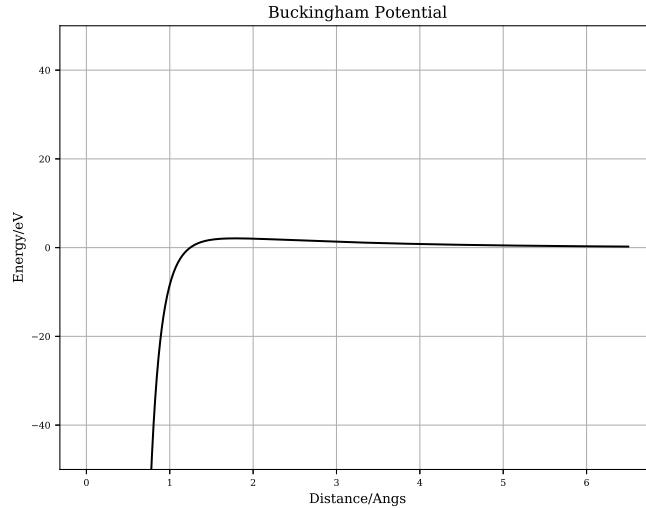


Figure R.2: Buckingham Potential

R.2.3 Cubic Knot Spline

$$V(r) = \sum_i^N a_i (r - r_i)^3 H(r_i - r)$$

where

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \quad (\text{R.3})$$

Listing R.3: Cubic Knot Spline

```
1 #TYPE cubic_knot_spline
2 #P 0.0 1.0 0.02 -0.01 0.002 0.0
3 #PF 0.0 1.0 2.0 3.0 4.0 5.0
```

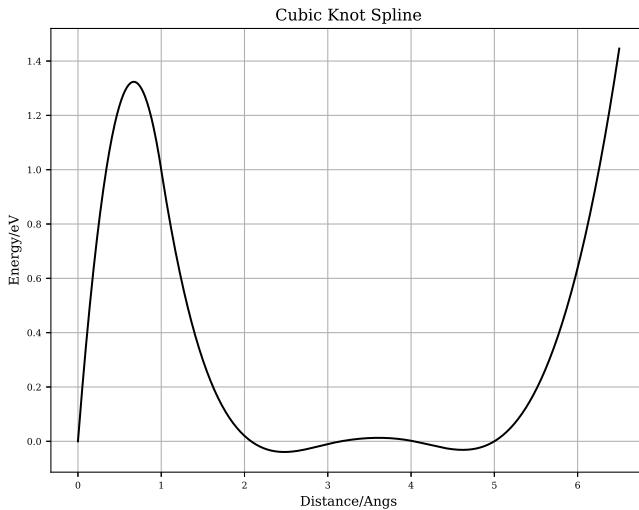


Figure R.3: Cubic Knot Spline

R.2.4 Cubic Knot Spline Fixed End

$$V(r) = \sum_i^N a_i (r - r_i)^3 H(r_i - r)$$

where

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \quad (\text{R.4})$$

Listing R.4: Cubic Knot Spline Fixed End

```
1 #TYPE cubic_knot_spline_fixed_end
2 #P 0.0 1.0 0.02 -0.01 0.002 0.0
3 #PF 0.0 1.0 2.0 3.0 4.0 5.0 6.5 0.0 0.0
```

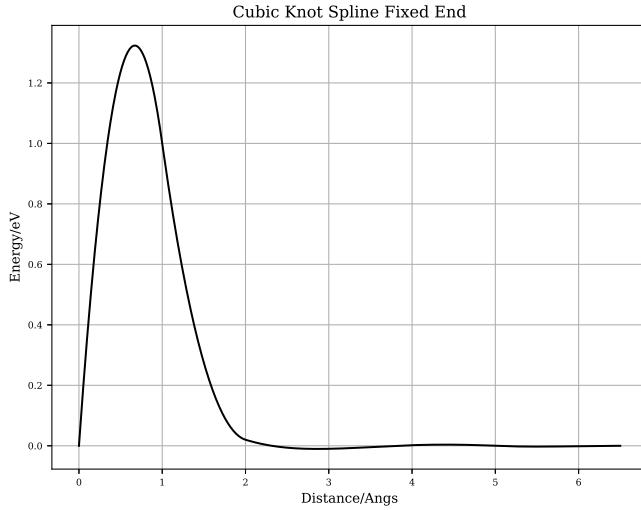


Figure R.4: Cubic Knot Spline Fixed End

R.2.5 Cubic Knot Spline Fixed End Pair

$$V(r) = \sum_i^N a_i (r - r_i)^3 H(r_i - r)$$

where

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \quad (R.5)$$

Listing R.5: Cubic Knot Spline Fixed End Pair

```

1 #TYPE cubic_knot_spline_fixed_end_pair
2 #P 0.5 0.007 -0.3 0.000001 0.002 0.0
3 #PF 2.0 2.5 3.0 3.5 4.0 5.0 26.0 26.0 1.0 6.5 0.0 0.0

```

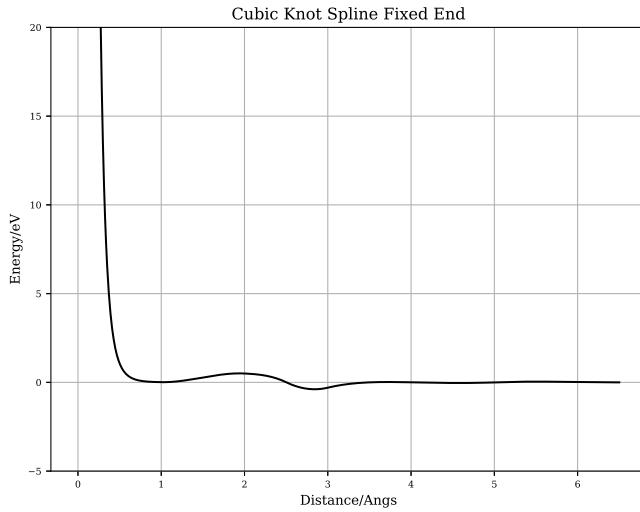


Figure R.5: Cubic Knot Spline Fixed End Pair

R.2.6 Cubic Splines

$$V(r) = \sum_i^N a_i (r - r_i)^3 H(r_i - r)$$

where

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \quad (\text{R.6})$$

This function requires two sets of parameters. P is a list of N coefficients and PF is a list of N cutoffs. The cubic polynomials are summed and individually scaled by the coefficients, and by virtue of its form and the heaviside step function, they cut off at the desired radius.

Listing R.6: Cubic Splines

```

1 #TYPE cubic_spline
2 #P -165.0 -78.5 -78.15 1.868
3 #PF 0.976 1.15 1.216 1.650

```

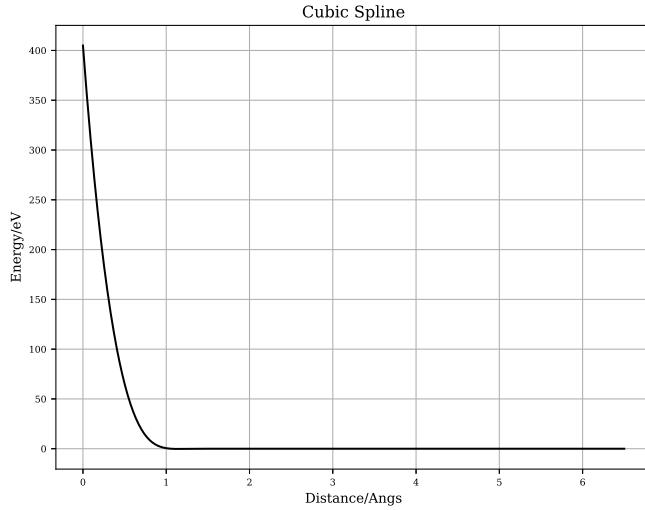


Figure R.6: Ackland Embedding

R.2.7 Cubic Spline Plus ZBL

$$V(r) = \begin{cases} z(r) & r < r_1 \\ a_0 + a_1r + a_2r^2 + a_3r^3 & r_1 \leq r \leq r_2 \text{ where} \\ \sum_i^N a_i(r - r_i)^3 H(r_i - r) & r > r_2 \end{cases} \quad (\text{R.7})$$

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

This function requires two sets of parameters. P is a list of N coefficients and PF is a list of N cutoffs. The cubic polynomials are summed and individually scaled by the coefficients, and by virtue of its form and the heaviside step function, they cut off at the desired radius.

Listing R.7: Cubic Splines Plus ZBL

```

1 #TYPE cubic_spline_zbl
2 #P 0.707937643181 1.00089008639 0.978759224081 -0.0460365116314 -0.00762125172653 -0.0209035950064
   0.00938859724754 0.000854708875904 0.0230647235533 0.011459696349 0.023419434017 0.0220932107774
   0.0325043051929 -0.0425828941427 -0.0147289840464 0.0138435254515 0.000880298224106 -0.0134658842396
   0.00102524605929
3 #PF 1.7 1.8 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 3.0 3.3 3.7 4.2 4.7 5.3 6.0 6.5 26.0 26.0 0.55 1.7 1.0

```

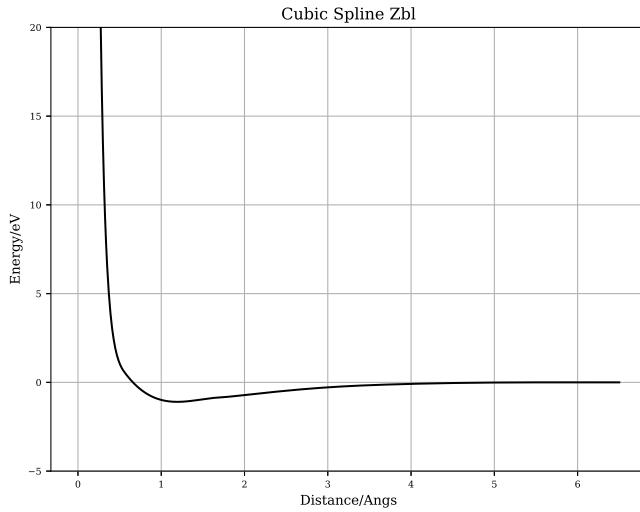


Figure R.7: Ackland Embedding

Listing R.8: Cubic Splines Plus ZBL

```

1 #TYPE cubic_spline_zbl
2 #P 1.00048374974 0.999953842306 0.817335736705 -0.0469506463728 -0.118042905174 0.00429308591859 0.024332744974
   0.0454260983505 0.0205392979832 0.415064378029 0.649740384042 0.0433627017609 0.047890256076 0.026097144285
   0.0106186840232 0.0138597059993 -0.0172453611171 -0.00144350314717 -0.00170969310596
3 #PF 1.7 1.8 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 3.0 3.3 3.7 4.2 4.7 5.3 6.0 6.5 26.0 26.0 0.6 1.6 1.0

```

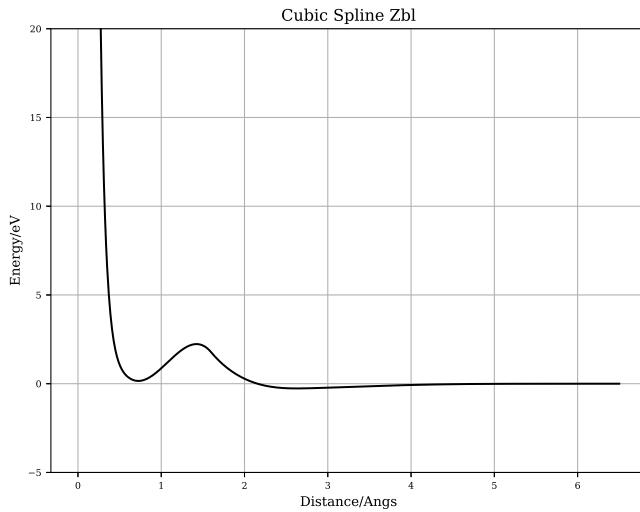


Figure R.8: Ackland Embedding

R.2.8 Embedding A - Finnis Sinclair

$$F(\rho) = -a\sqrt{\rho} \quad (\text{R.8})$$

This function requires one parameter[95].

Listing R.9: Embedding A

```
1 #TYPE embedding_a
2 #P 0.1
```

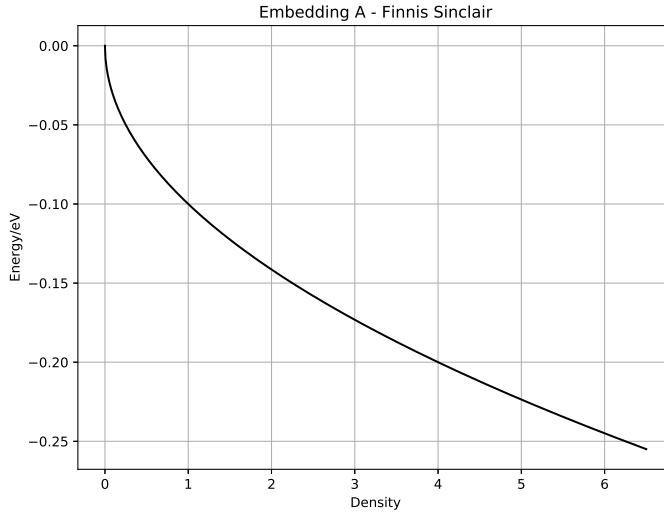


Figure R.9: Embedding A

R.2.9 Embedding B - Mendelev

$$F(\rho) = -\sqrt{\rho} + a\rho^2 \quad (\text{R.9})$$

This function requires one parameter. Development of new interatomic potentials appropriate for crystalline and liquid iron, 2003.

Listing R.10: Embedding B

```
1 #TYPE embedding_b
2 #P 0.1
```

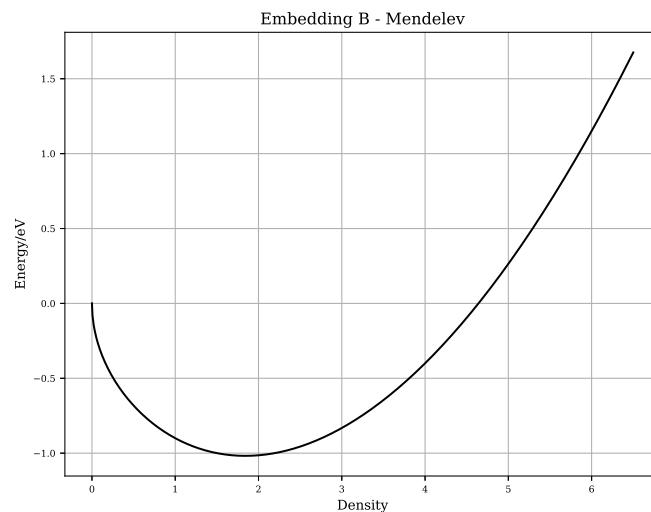


Figure R.10: Embedding B

R.2.10 Embedding C

$$F(\rho) = a\sqrt{\rho} + b\rho^2 \quad (\text{R.10})$$

Listing R.11: Embedding C

```
1 #TYPE embedding_c
2 #P 0.1 0.1
```

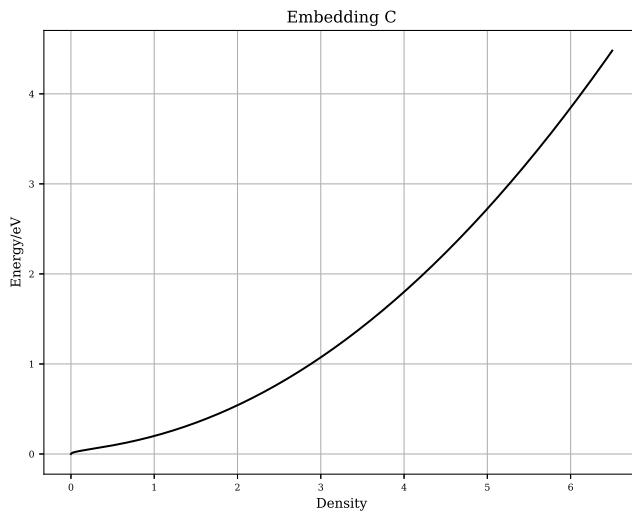


Figure R.11: Embedding C

R.2.11 Embedding D - Ackland-Mendelev

$$F(\rho) = -\sqrt{(\rho + a\rho^2 + b\rho^4)} \quad (\text{R.11})$$

This function requires one parameter. Development of an interatomic potential for phosphorus impurities in -iron, 2004.

Listing R.12: Embedding D

```
1 #TYPE embedding_d
2 #P 0.05 -0.00001
```

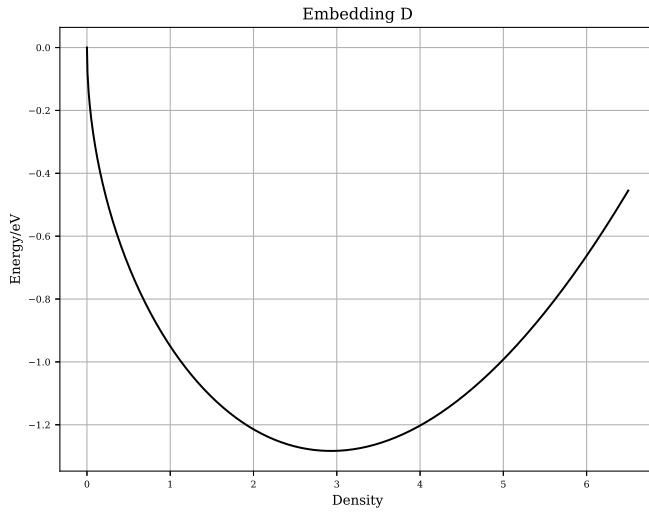


Figure R.12: Embedding D

R.2.12 Embedding E

$$F(\rho) = a\sqrt{\rho} + b\rho^2 + c\rho^4 \quad (\text{R.12})$$

Listing R.13: Embedding E

```
1 #TYPE embedding_e
2 #P -0.8 0.05 -0.00001
```

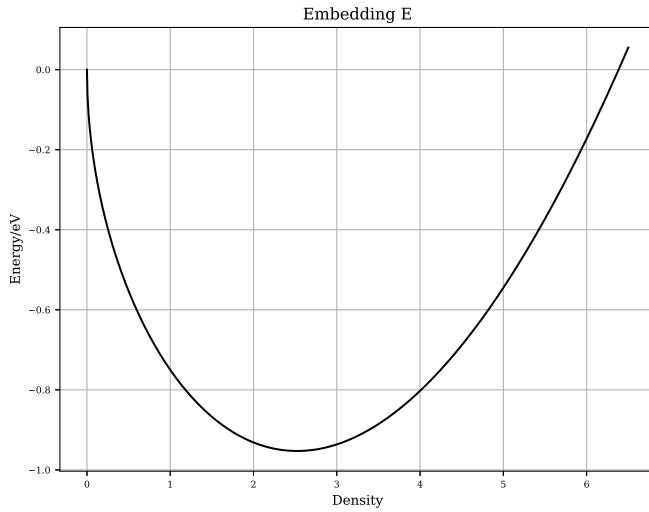


Figure R.13: Ackland Embedding

R.2.13 Embedding F

$$F(\rho) = a\sqrt{\rho} + b\rho^2 + \rho^4 \quad (\text{R.13})$$

Listing R.14: Embedding F

```
1 #TYPE embedding_f
2 #P -0.8 0.05
```

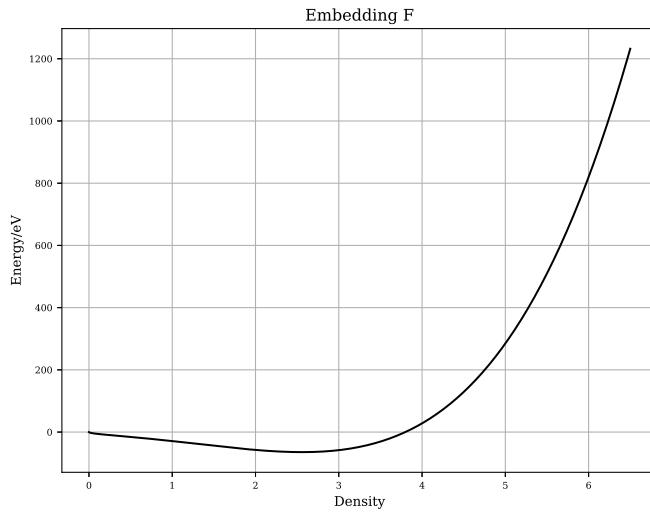


Figure R.14: Embedding F Function Plot

R.2.14 Embedding G

$$F(\rho) = a\sqrt{\rho} + b\rho^2 + f\rho^4 \quad (\text{R.14})$$

Listing R.15: Embedding F

```
1 #TYPE embedding_g
2 #P -9.15 5.57
3 #PF 0.01
```

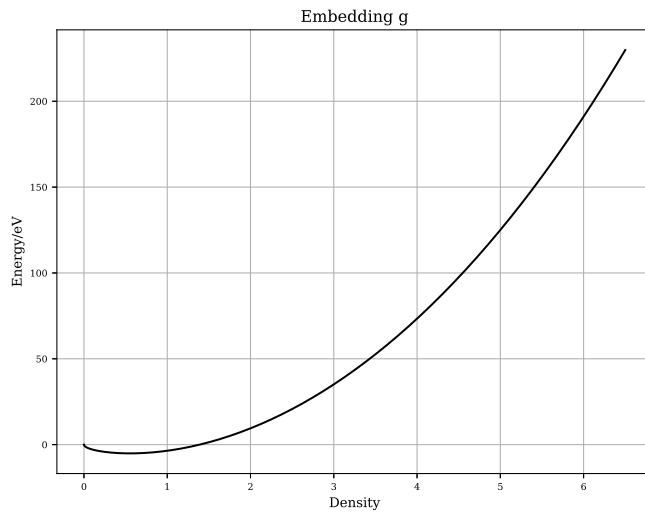


Figure R.15: Embedding G Function Plot

R.2.15 Lennard-Jones

$$V(r) = e \left(\left(\frac{r_m}{r} \right)^{12} - 2 \left(\frac{r_m}{r} \right)^6 \right) \quad (\text{R.15})$$

Listing R.16: Lennard-Jones

```
1 #TYPE lennard_jones
2 #P 2.3 3.5
```

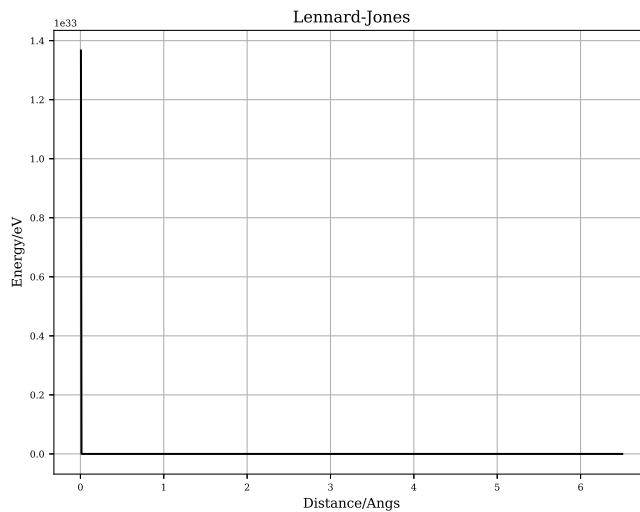


Figure R.16: Lennard-Jones

R.2.16 Morse

$$V(r) = \exp(-2a(r - r_e)) - 2\exp(-a \times (r - r_e)) \quad (\text{R.16})$$

Listing R.17: Morse

```
1 #TYPE morse
2 #P 4.669 1.256 2.8
```

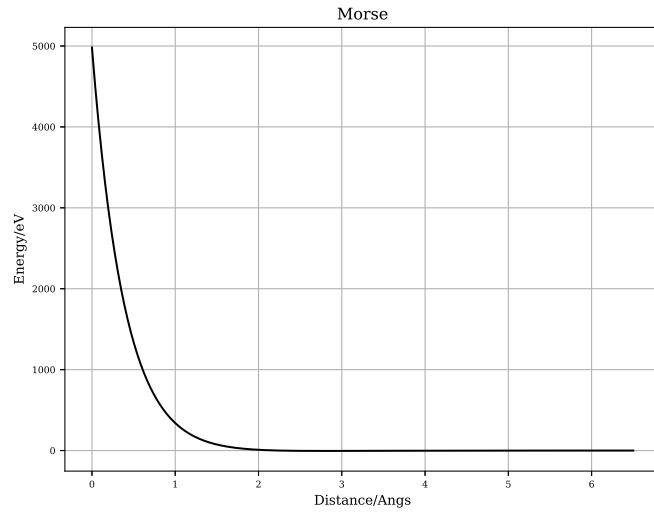


Figure R.17: Morse

R.2.17 Slater 4S (Squared)

$$\rho(r) = (N_s r^3 \exp(-\eta r))^2 \quad (\text{R.17})$$

Listing R.18: Slater 4S

```
1 #TYPE slater_4s
2 #P 5.0 1.323
```

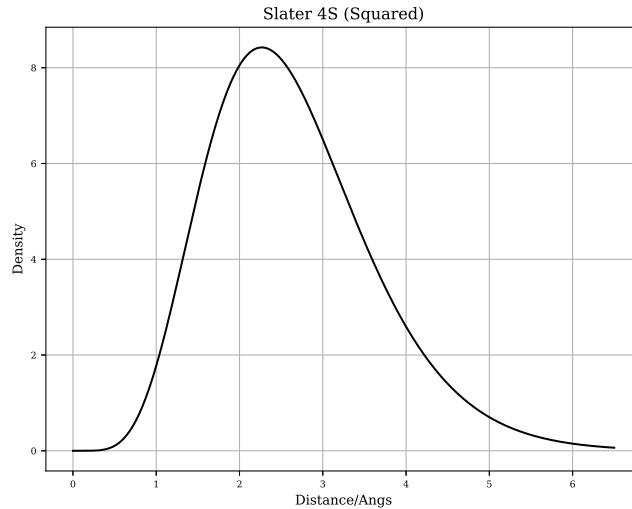


Figure R.18: Quadratic Density

R.2.18 ZBL

$$\phi(x) = 0.181e^{-3.2x} + 0.5099e^{-0.9423x} + 0.2802e^{-0.4029x} + 0.02817e^{-0.2016x}$$

$$\text{where } a_{ij} = \frac{0.8854a_0}{Z_i^{0.23} + Z_j^{0.23}} \quad (\text{R.18})$$

and $a_0 = 0.529$ angstrom

Listing R.19: ZBL

```
1 #TYPE zbl
2 #P 26.0 26.0
```

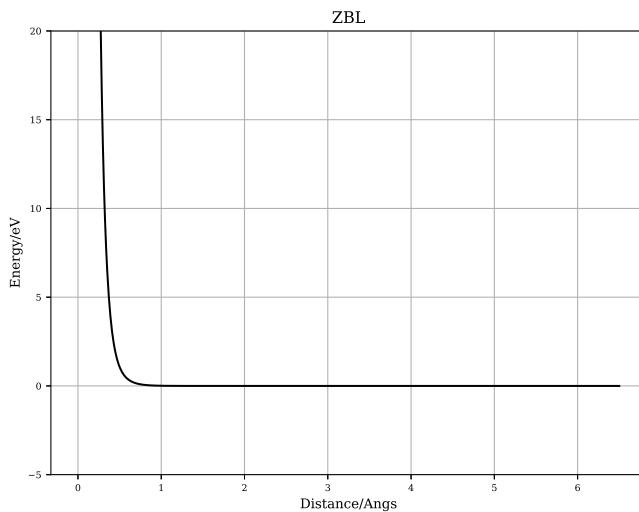


Figure R.19: ZBL

R.2.19 Zero

$$f(x) = 0 \quad (\text{R.19})$$

Listing R.20: Zero

```
1 #TYPE zero
2 #P 0.0
```

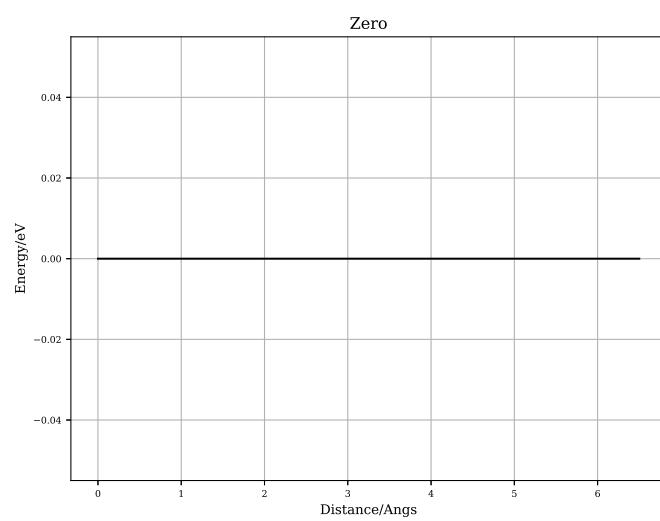


Figure R.20: Zero

Appendix S

FePd Fit: Version 1

This section of the appendix contains the full details of the first version of the Fe-Pd potential.

S.1 Potential Files

Listing S.1: Potential Index File fepd.pot

```
1 POTNAME fe_pot
2
3 START
4 F_ON true
5 FILE fe_pair.pot
6 LABEL FE FE
7 F_TYPE PAIR
8 END
9
10 START
11 F_ON true
12 FILE fe_dens.pot
13 LABEL FE
14 F_TYPE DENS
15 F_GROUP Group1
16 END
17
18 START
19 F_ON true
20 FILE fe_embe.pot
21 LABEL FE
22 F_TYPE EMBE
23 F_GROUP Group1
24 END
25
26 START
27 F_ON true
28 FILE pd_pair.pot
29 LABEL PD PD
30 F_TYPE PAIR
31 END
32
33 START
34 F_ON true
35 FILE pd_dens.pot
36 LABEL PD
```

```

37 F_TYPE DENS
38 F_GROUP Group1
39 END
40
41 START
42 F_ON true
43 FILE pd_embe.pot
44 LABEL PD
45 F_TYPE EMBE
46 F_GROUP Group1
47 END
48
49 START
50 F_ON true
51 FILE fepd_pair.pot
52 LABEL FE PD
53 F_TYPE PAIR
54 END

```

Listing S.2: fe_pair.pot

```

1 #TYPE cubic_knot_spline_fixed_end
2 #P 50089.4914436 1495.0173568 41.8077313048 2.62868971288 0.884035467244 0.518520725075 0.310525323002
   0.151317904619 0.0457231898041 -0.00199232081514 -0.0285815828726 -0.044141420039 -0.0826842455756
   -0.0800699021737 -0.0749035806431 -0.0483139551314 -0.0271622770015 -0.0244614853927 -0.0276516107094
   -0.0215826770276 -0.0100294465383 -0.00704320230389 -0.00983027553784 -0.00397339407663 -0.00187138774473
   0.000692137630225 0.00191634452235 0.00292127387636 0.00236805599954 0.000626427605252 -0.000652560615754
   -0.0011085461043 -0.0012092545989 -0.000773644125613 -0.000205249951821 -2.49686593374e-08 0.0
3 #PF 0.0 0.1472 1.0304 1.9328 2.2272 2.3744 2.5216 2.6688 2.816 2.9632 3.1104 3.2576 3.4048 3.4944 3.6416 3.7888
   3.936 4.0832 4.2304 4.3776 4.5248 4.672 4.8192 4.9664 5.1136 5.2608 5.408 5.4848 5.6128 5.7408 5.8688 5.9968
   6.0096 6.1376 6.2656 6.3936 6.4 6.5 0.0 0.0
4 #VR 1.0

```

Listing S.3: pd_pair.pot

```

1 #TYPE cubic_knot_spline_fixed_end
2 #P 57433.8196425 1246.54498259 372.86926226 186.530902261 90.9601078094 64.4722319917 44.7955309107 30.9422713797
   19.7661678393 22.7730631807 17.8712893446 20.5301963435 15.7532904958 8.14434737667 3.57750340968
   2.24283276474 1.29878073656 0.681897587232 0.3413406266 0.139633674104 0.0415441088092 -0.00157891985762
   -0.0265766263265 -0.0519228478909 -0.0713202575903 -0.0818422199207 -0.0763981760836 -0.0408735005309
   -0.0337434510853 -0.0340454413917 -0.0221603780251 -0.0140921270967 -0.0112860782378 -0.0075565994614
   -0.00650714812082 -0.00492928641989 -0.00224835545732 0.000551830630351 0.00190602741869 0.00239930313018
   0.00185508157381 0.000683532974593 -0.000618830916973 -0.00106264430655 -0.00107459874027 -0.000816500845506
   -0.000221711741725 -2.71267919382e-08 0.0
3 #PF 0.0 0.1472 0.2944 0.4416 0.5888 0.736 0.8832 1.0304 1.1776 1.1968 1.344 1.4912 1.6384 1.7856 1.9328 2.08
   2.2272 2.3744 2.5216 2.6688 2.816 2.9632 3.1104 3.2576 3.4048 3.4944 3.6416 3.7888 3.936 4.0832 4.2304 4.3776
   4.5248 4.672 4.8192 4.9664 5.1136 5.2608 5.408 5.4848 5.6128 5.7408 5.8688 5.9968 6.0096 6.1376 6.2656
   6.3936 6.4 6.5 0.0 0.0
4 #VR 0.2

```

Listing S.4: fepd_pair.pot

```

1 #TYPE cubic_knot_spline_fixed_end
2 #P 50509.5071771 1510.58202206 41.6181984732 2.73482804796 0.877876472889 0.515874627808 0.308018317521
   0.157372115281 0.0464758155843 -0.00201060627526 -0.0293229989679 -0.0438255499706 -0.0818154118837
   -0.079548485098 -0.0743015444599 -0.0497182011604 -0.027687237496 -0.0241247837683 -0.0288217312248
   -0.0218938512359 -0.00984505409127 -0.00702367568202 -0.00995935352427 -0.00396441140107 -0.00186009860295
   0.000684536237639 0.00191771085336 0.0028941005267 0.00229363524352 0.000624566932013 -0.000659755922205
   -0.00111343976897 -0.00119282373519 -0.000778770104568 -0.000201643446637 -2.50582829498e-08 0.0
3 #PF 0.0 0.1472 1.0304 1.9328 2.2272 2.3744 2.5216 2.6688 2.816 2.9632 3.1104 3.2576 3.4048 3.4944 3.6416 3.7888
   3.936 4.0832 4.2304 4.3776 4.5248 4.672 4.8192 4.9664 5.1136 5.2608 5.408 5.4848 5.6128 5.7408 5.8688 5.9968
   6.0096 6.1376 6.2656 6.3936 6.4 6.5 0.0 0.0
4 #VR 1.0

```

Listing S.5: fe.dens.pot

```
1 #TYPE cubic_knot_spline
2 #P 0.0 1.11185118593 1.07405655632 0.501301076432 0.157569861684 0.0558612376686 0.0331469688897 0.0185595436281
   0.0100054223074 0.00599263304398 0.0031448455987 0.00203980413638 0.00194732163608 0.00155166455135
   0.00107322938655 0.000574828582083 0.000328728725463 0.000104765166684 2.21153619775e-05 7.18575750428e-07
   0.0
3 #PF 0.0 1.184 1.1904 1.4912 1.792 2.0928 2.3936 2.6944 2.9952 3.296 3.5968 3.8976 4.1984 4.4992 4.8 5.1008 5.4016
   5.7024 6.0032 6.304 6.4
4 #VR 0.2
```

Listing S.6: pd.dens.pot

```
1 #TYPE cubic_knot_spline_fixed_end
2 #P 0.0 0.0575871210036 0.0349621319115 0.0175098696686 0.0102461350982 0.00571266625643 0.00297811468455
   0.00210702585433 0.00197274218575 0.00158627909383 0.00107976661552 0.0005622208493 0.000317467916465
   0.000111251939791 2.24374877178e-05 6.933985749e-07 0.0
3 #PF 0.0 2.0928 2.3936 2.6944 2.9952 3.296 3.5968 3.8976 4.1984 4.4992 4.8 5.1008 5.4016 5.7024 6.0032 6.304 6.4
   6.5 0.0 0.0
4 #VR 0.2
```

Listing S.7: fe_embe.pot

```
1 #TYPE embedding_g
2 #P -8.8990305293 -0.656860026472
3 #PF 0.1
4 #VR 2.0
```

Listing S.8: fe_embe.pot

```
1 #TYPE embedding_g
2 #P -9.50412589256 8.00275039263
3 #PF 0.1
4 #VR 0.5
```

S.2 Potential Plots

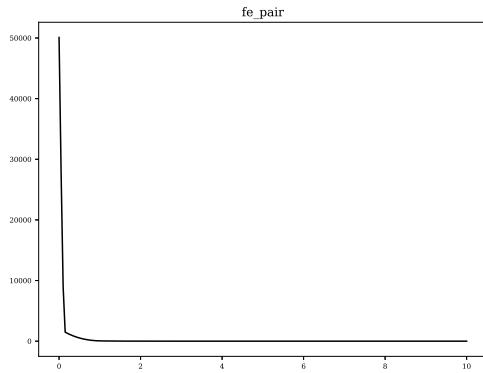


Figure S.1: Fe-Fe Pair Function

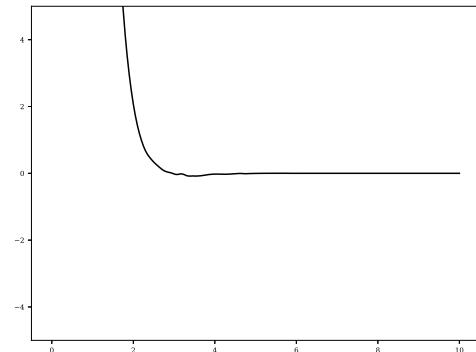


Figure S.2: Fe-Fe Pair Function (Zoomed In)

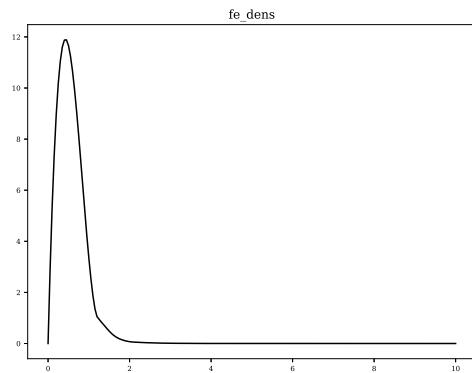


Figure S.3: Fe Density Function

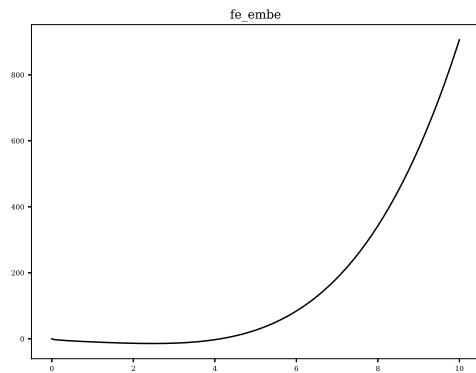


Figure S.4: Fe Embedding Function

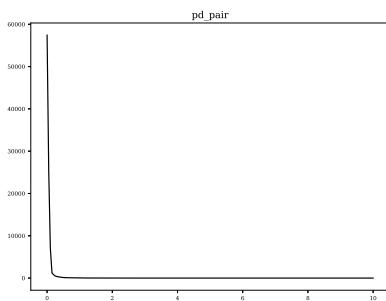


Figure S.5: Pd-Pd Pair Function

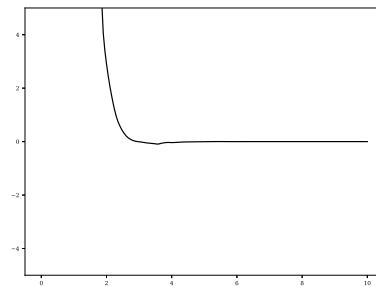


Figure S.6: Pd-Pd Pair Function (Zoomed In)

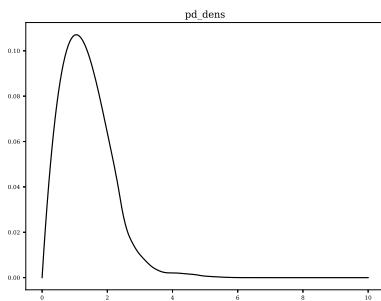


Figure S.7: Pd Density Function

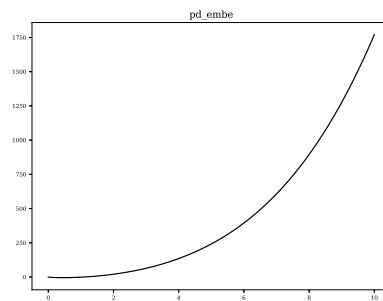


Figure S.8: Pd Embedding Function

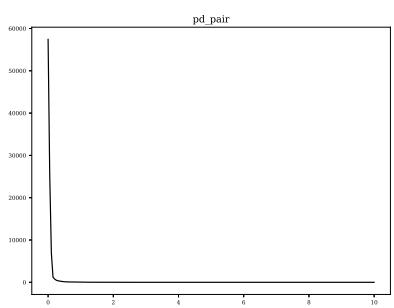


Figure S.9: Fe-Pd Pair Function

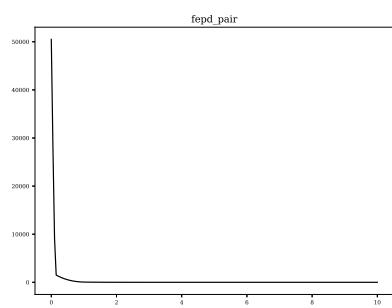


Figure S.10: Fe-Pd Pair Function (Zoomed In)

S.3 Equation of State and Elastic Constant (Distortion) Plots

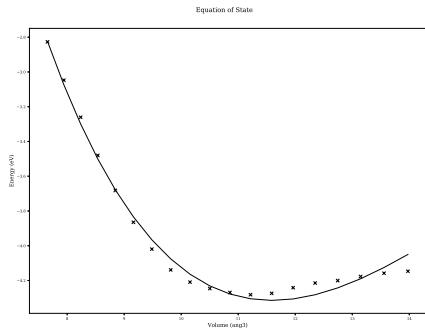


Figure S.11: Iron equation of state

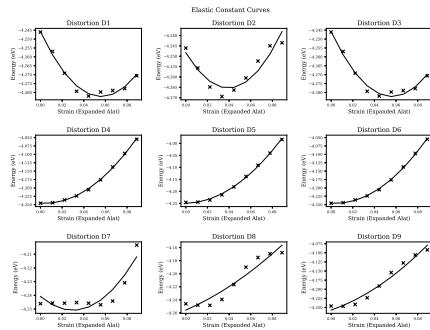


Figure S.12: Iron elastic constants

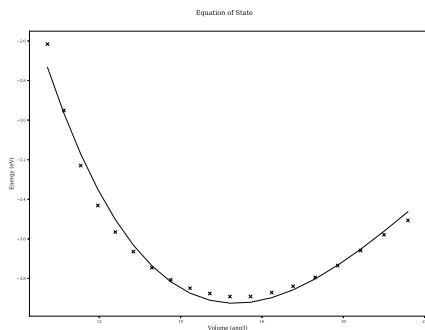


Figure S.13: Palladium equation of state

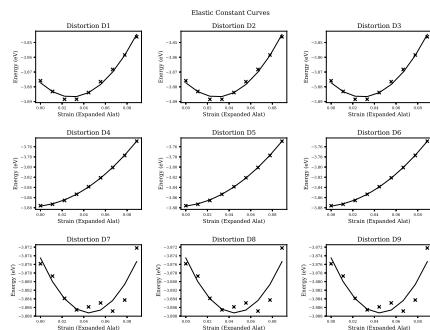


Figure S.14: Palladium elastic constants

S.4 Cohesive Energy Plots

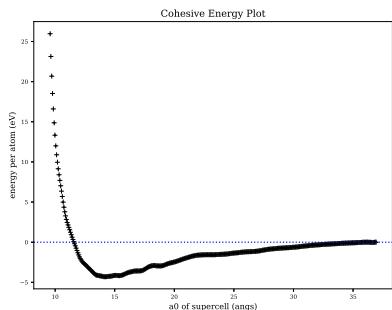


Figure S.15: Iron cohesive energy

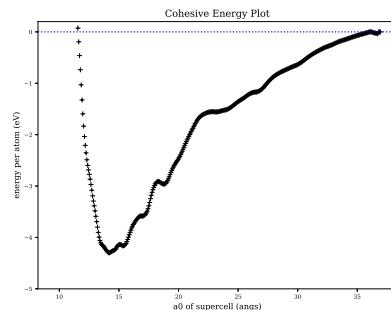


Figure S.16: Iron cohesive energy (zoomed in)

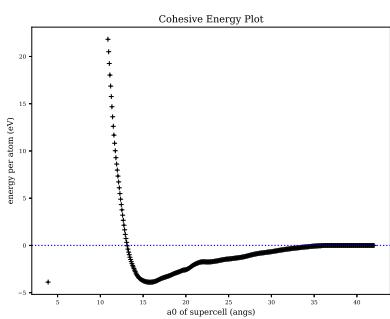


Figure S.17: Palladium cohesive energy

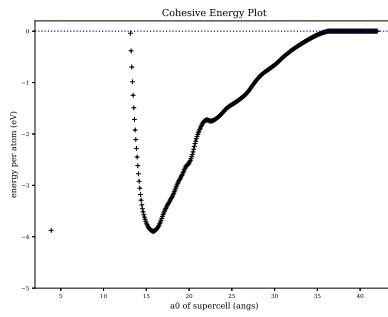


Figure S.18: Palladium cohesive energy (zoomed in)

S.5 Surface Energy Plots

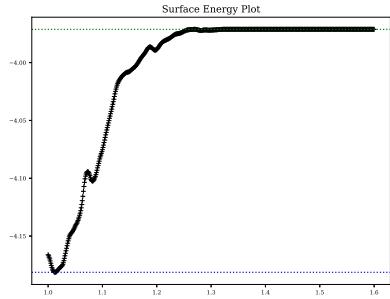


Figure S.19: Iron surface energy

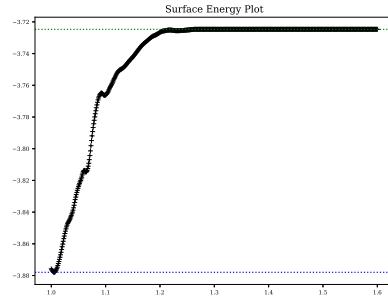


Figure S.20: Palladium surface energy

S.6 Fe-Pd V1 Potential Properties

Parameter	Experimental/DFT	This Potential
a_0	3.42	3.58
e_0	-4.27	-4.27
B_0	222.0	238.8
C_{11}	365.6	356.8
C_{22}	298.7	296.1
C_{33}	364.0	356.8
C_{44}	186.3	180.9
C_{55}	266.8	263.2
C_{66}	186.3	180.9
C_{12}	141.6	139.1
C_{13}	233.8	241.7
C_{23}	130.4	127.3

Table S.1: Iron potential properties (experimental vs this work v1)

Parameter	Experimental/DFT	This Potential
a_0	3.89	3.95
e_0	-3.91	-3.91
B_0	180.0	171.3
C_{11}	234.0	248.9
C_{12}	176.0	167.2
C_{44}	71.2	67.6

Table S.2: Palladium potential properties (experimental vs this work v1)

Appendix T

FePd Fit: Version 2

This section of the appendix contains the full details of the second version of the Fe-Pd potential.

T.1 Potential Files

Listing T.1: Potential Index File fepd.pot

```
1 POTNAME pdfe_pot
2
3 START
4 F_ON true
5 FILE fe_pair.pot
6 LABEL FE FE
7 F_TYPE PAIR
8 END
9
10 START
11 F_ON true
12 FILE pd_pair.pot
13 LABEL PD PD
14 F_TYPE PAIR
15 END
16
17 START
18 F_ON true
19 FILE fepd_pair.pot
20 LABEL FE PD
21 F_TYPE PAIR
22 END
23
24 START
25 F_ON true
26 FILE fe_dens_slater_4s.pot
27 LABEL FE
28 F_TYPE DENS
29 F_GROUP D1
30 END
31
32 START
33 F_ON true
34 FILE fe_embe_ackland.pot
35 LABEL FE
36 F_TYPE EMBE
```

```

37 F_GROUP D1
38 END
39
40 START
41 F_ON true
42 FILE pd_dens_slater_4s.pot
43 LABEL PD
44 F_TYPE DENS
45 F_GROUP D1
46 END
47
48 START
49 F_ON true
50 FILE pd_embe_ackland.pot
51 LABEL PD
52 F_TYPE EMBE
53 F_GROUP D1
54 END

```

Listing T.2: fe_pair.pot

```

1 #TYPE cubic_knot_spline_5
2 #P 0.534414176557 2.47158957018 3.62572960689 3.82910960636 5.19186569757 6.33755681101 -0.841594441898
   -0.0382741742048 -0.0294982066555 -0.128378635827 0.00520712319717 -0.00173340452865 0.0394106099537
   -0.0422374384814 -0.0212703972446 8.0576645813 0.0114178033157
3 #PF 26.0 26.0 1.0 0.0 6.5 0.0 0.0
4 #VR 0.01

```

Listing T.3: pd_pair.pot

```

1 #TYPE cubic_knot_spline_5
2 #P 0.621299299174 2.49026129701 3.70671760046 4.0464633078 5.12962452826 10.465167784 -0.531944848494
   0.00813465723736 0.017090970358 0.0234628754648 0.00109856040369 0.00122784782403 0.00107526425391
   0.00075123581898 0.000782865546721 38.7466665287 0.00147277281864
3 #PF 26.0 26.0 1.0 0.0 6.5 0.0 0.0
4 #VR 0.01

```

Listing T.4: fepd_pair.pot

```

1 #TYPE cubic_knot_spline_5
2 #P 0.602048570084 2.26359176313 3.43770789475 3.82473216113 6.2662084167 10.0200084349 -0.575534460106
   0.00701145112101 0.0163328650575 0.0209807243721 0.000935085295777 0.00104298283428 0.00113879012964
   0.00108097915755 0.000910022605946 47.310687291 0.00094153913992
3 #PF 26.0 26.0 1.0 0.0 6.5 0.0 0.0
4 #VR 0.01

```

Listing T.5: fe_dens.pot

```

1 #TYPE slater_4s
2 #P 0.112239220675 1.4062318087
3 #PF 0.0
4 #VR 0.01

```

Listing T.6: pd_dens.pot

```

1 #TYPE slater_4s
2 #P 0.841303167535 1.82754932548
3 #PF 0.0
4 #VR 0.01

```

Listing T.7: fe_embe.pot

```

1 #TYPE embedding_g
2 #P 17.3081279913 -42.0918942985

```

```
3 #PF 0.1  
4 #VR 0.01
```

Listing T.8: fe_embe.pot

```
1 #TYPE embedding_g  
2 #P -6.74702954151 41.2855552159  
3 #PF 0.1  
4 #VR 0.01
```

T.2 Potential Plots

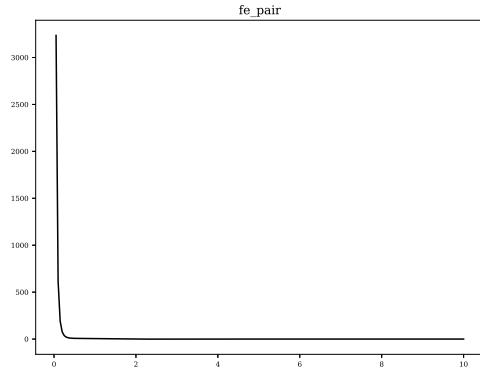


Figure T.1: Fe-Fe Pair Function

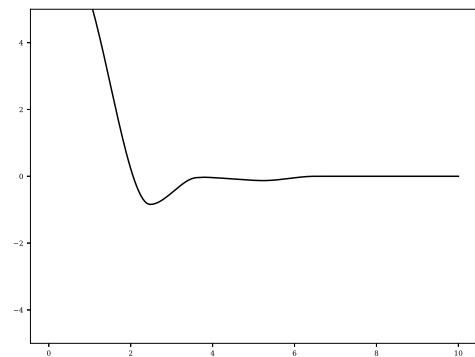


Figure T.2: Fe-Fe Pair Function (Zoomed In)

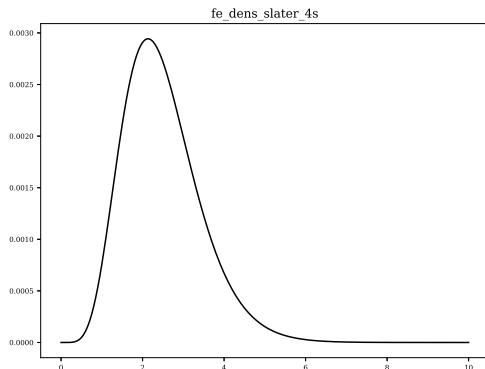


Figure T.3: Fe Density Function

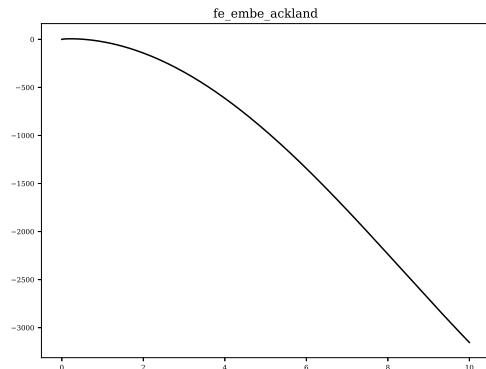


Figure T.4: Fe Embedding Function

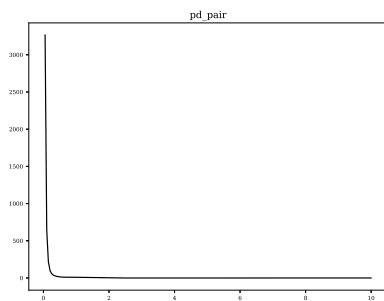


Figure T.5: Pd-Pd Pair Function

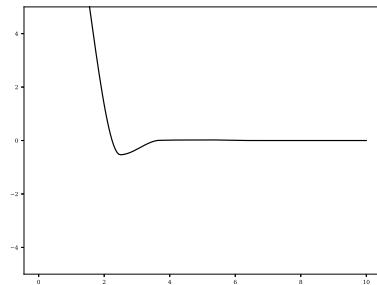


Figure T.6: Pd-Pd Pair Function (Zoomed In)

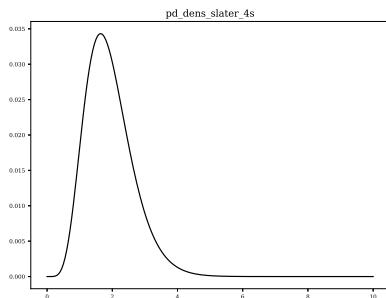


Figure T.7: Pd Density Function

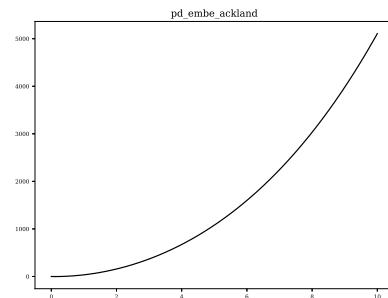


Figure T.8: Pd Embedding Function

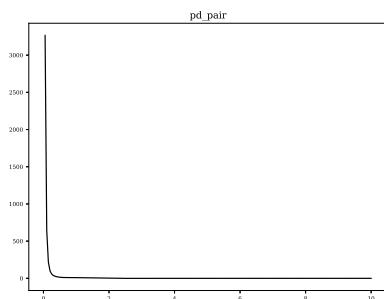


Figure T.9: Fe-Pd Pair Function

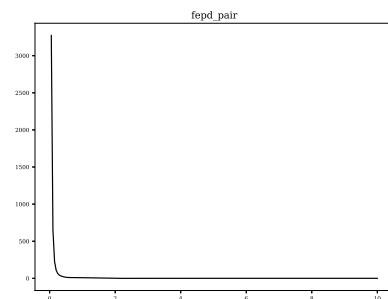


Figure T.10: Fe-Pd Pair Function (Zoomed In)

T.3 Equation of State and Elastic Constant (Distortion) Plots

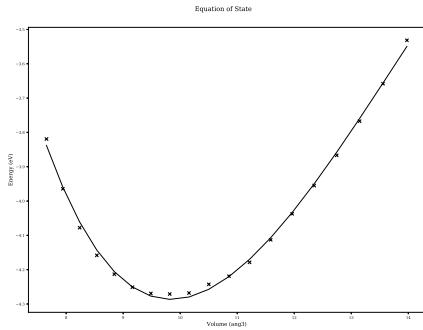


Figure T.11: Iron equation of state

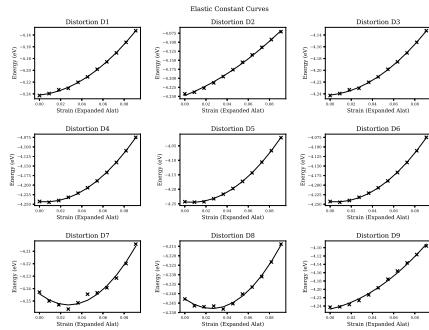


Figure T.12: Iron elastic constants

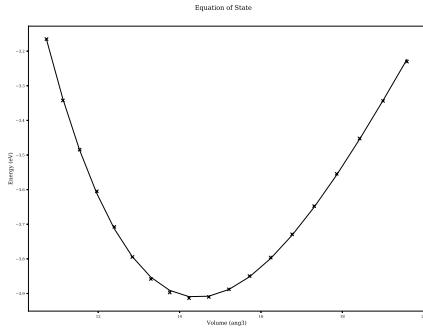


Figure T.13: Palladium equation of state

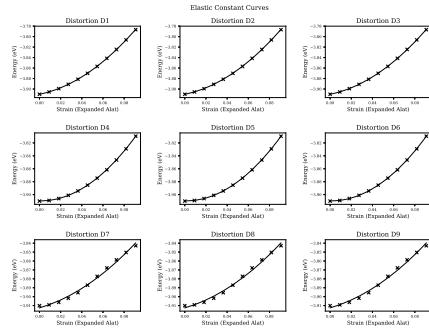


Figure T.14: Palladium elastic constants

T.4 Cohesive Energy Plots

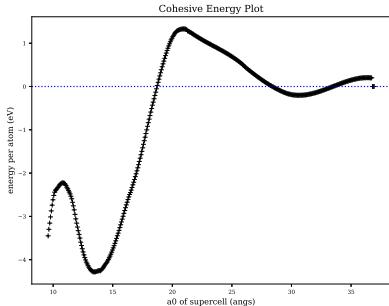


Figure T.15: Iron cohesive energy

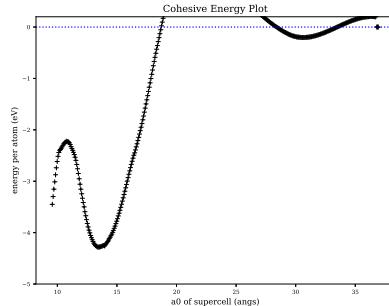


Figure T.16: Iron cohesive energy (zoomed in)

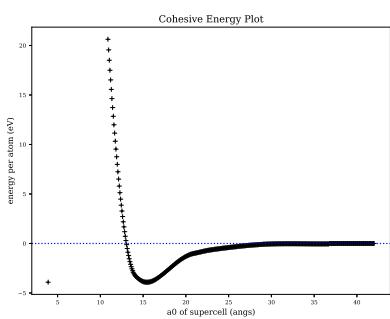


Figure T.17: Palladium cohesive energy

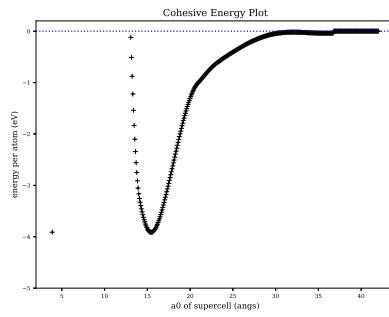


Figure T.18: Palladium cohesive energy (zoomed in)

T.5 Surface Energy Plots

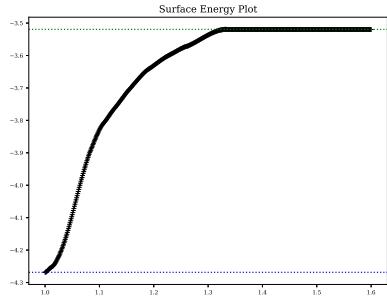


Figure T.19: Iron surface energy

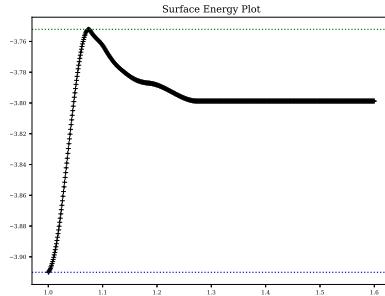


Figure T.20: Palladium surface energy

T.6 Fe-Pd V2 Potential Properties

Parameter	Experimental/DFT	This Potential
a_0	3.42	3.40
e_0	-4.27	-4.29
B_0	222.0	229.1
C_{11}	365.6	365.9
C_{22}	298.7	298.6
C_{33}	364.0	365.9
C_{44}	186.3	187.6
C_{55}	266.8	258.4
C_{66}	186.3	187.6
C_{12}	141.6	141.1
C_{13}	233.8	238.1
C_{23}	130.4	124.8

Table T.1: Iron potential properties (experimental vs this work v2)

Parameter	Experimental/DFT	This Potential
a_0	3.89	3.87
e_0	-3.91	-3.91
B_0	180.0	171.3
C_{11}	234.0	232.9
C_{12}	176.0	176.7
C_{44}	71.2	68.8

Table T.2: Palladium potential properties (experimental vs this work v2)