The Simulation of a Nuclear Reactor

Jake Libbeter, Oscar Stewart-Tanner, Daniel Nutt, George Wareing, Will Forshaw, Jack McCrindle and Jay Wala.

Department of Physics: PHYS205
University of Liverpool

Project title: **PHYS205 Fission Reactor Simulator**
Directory of .ipynb notebook: **Final files/Fission Reactor.ipynb**

Abstract

Nuclear physics is a branch of physics that studies the properties, structure and behaviour of atomic nuclei. By exploring the fundamental forces and interactions that bind protons and neutrons together, it is possible to delve into the concepts of nuclear reactions, radioactive decay and nuclear energy production. Our project mathematically models the fission of ^{235}U inside a Boiling Water Reactor, calculating and summing the energy released by the chain reaction, using a Monte Carlo method to iterate the simulations. Using Pygame, an interactive model was constructed, allowing the user to change the properties of the reactor, for example the insertion depth of the control rods. This visualisation ties together with the nuclear calculations to provide a simulation of a nuclear fission reactor, generating the expected power output of approximately 1 GW of power at 20% control rod insertion.

1 Introduction

Nuclear fission refers to the process where a large, unstable nucleus divides into two lighter daughter nuclei and between two and four neutrons as a byproduct. This decay can occur spontaneously, or it can be induced by striking the nucleus with a neutron of sufficient speed. Uranium releases an average of 2.4 neutrons per fission event, often rounded to 2 for simplicity. Fission generates energy as the combined rest mass energies of the resulting daughter products are lower than that of the parent nuclei, releasing energy equivalent to change in net binding energies between the daughter and parent nuclei nuclei . This energy can be equated to a mass deficit using Einstein's Mass Energy Equivalence formula, which asserts that

$$E = mc^2 \tag{1}$$

where E is the rest energy of the system, m is the rest mass of the system and c is the speed of light. By utilizing our understanding of both the mass prior to and following the fission event, we define Q as the energy output per fission event

$$Q = (m_p - (m_{D1} + m_{D2} + 2m_n))c^2 \tag{2}$$

where m_p is the mass of the parent nuclei, m_{D1} and m_{D2} the masses of daughter nuclei, m_n is the mass of a neutron. With this central equation, the total energy output of fission reactions can be calculated. There are a range of daughter products due to the fission of ^{235}U . Each decay from a ^{235}U nucleus results in different daughter nuclei, and thus, a different mass deficit. Furthermore, each decay had subsequent β^- decays as the resulting daughter products may be unstable and decay in order to become stable, with stability defined achieved by lowering the N:Z ratio.

The neutrons that are released from the fission decays can then go on to collide with more ^{235}U nuclei and induce more fission reactions, producing further neutrons that can induce further fission. This is referred to as a chain reaction and the main goal of nuclear power is to moderate this chain reaction so that the reactor doesn't explode due to too much energy being released in a short time.

Particles of sufficiently large energy are said to be travelling at relativistic speeds, which means that classical mechanics can no longer be used to describe the motion of the particle. To determine whether a particle is travelling at relativistic speeds we can use the formula

$$\gamma = \frac{E_T}{E_0} = \frac{T + E_0}{E_0} \tag{3}$$

where γ is the Lorentz factor (**Eqn. 4**), E_T is the total energy of the particle, T is the kinetic energy of the particle and E_0 is the rest mass of the particle. The Lorentz factor can then be used to determine the velocity that the particle is travelling at from the definition

$$\gamma = \frac{1}{\sqrt{1 - \beta^2}} \tag{4}$$

where β is the fraction of light speed that the particle is moving at, $\beta = \frac{v}{c}$.

This can be rearranged for β in order to determine whether the neutrons travel at relativistic speeds.

$$\beta = \sqrt{1 - \frac{1}{\gamma^2}} \tag{5}$$

When it comes to talking about the interaction between particles, the probability of an event occurring needs to be mentioned. For particles, this is referred to as the cross section, σ, of an interaction and can be taken as a probability of an event occurring. The standard unit of a cross section is the barn, bn, coming from the phrase "as big as a barn door". $1\,\text{bn} = 1 \times 10^{-28}\,m^{-2}$ and the cross section for a nuclear fission reaction is on the order of $\sim \times 10^3$ bn.

It is necessary to consider something referred to as the mean free path, λ, as this is defined as the average distance between collisions inside the medium. The probability density function of a particle with a macroscopic cross section σ_T is defined as

$$P(x) = \sigma_T \cdot e^{-\sigma_T x} \tag{6}$$

The mean free path can then be found by integrating x multiplied by the probability density function between 0 and ∞

$$\lambda = \int_0^\infty x \cdot P(x)\, dx = \int_0^\infty \sigma_T x \cdot e^{-\sigma_T x}\, dx \tag{7}$$

When this integral is completed, it can be seen that the general result to this integral, and hence the mean free path, is given by

$$\lambda = \frac{1}{\sigma_T} \tag{8}$$

The neutrons that are released from ^{235}U fission reactions have a very large energy (~ 2 MeV), and in order to increase the cross section of induced fission, the neutrons must be slowed to thermal speeds (~ 25 meV). In order to determine the number of collisions needed to bring the neutrons to thermal speed, the logarithmic energy loss, ξ, must be considered, which is the average amount of energy lost per collision. This can be calculated by

$$\xi = \frac{(A-1)^2}{2A} \cdot \ln\left(\frac{A+1}{A-1}\right) \tag{9}$$

where A is the atomic mass number of the nucleus. This parameter is vital when it comes to selecting a good moderator, as better moderators have a higher value of ξ.

This allows us to quantify an estimate for the number of collisions required to bring the neutron to thermal speeds, n, given by

$$n = \frac{1}{\xi} \cdot \ln\left(\frac{E_{\text{start}}}{E_{\text{end}}}\right) \tag{10}$$

where E_{start} is the energy of the neutrons as they enter the moderator and E_{end} is the required energy of the neutrons when they leave the moderator. Better moderators have smaller values of n as they will require less material to bring the neutrons to thermal speeds.

A parameter can be defined called the slowing down power, η, which is the capacity of a moderator to slow down neutrons to thermal speeds, and is given by

$$\eta = \xi \cdot \sigma_s \tag{11}$$

where σ_s is the macroscopic scattering cross section of the material. The scattering cross section, σ_s, is the probability that the neutron will be scattered by the moderator. Conversely, absorption cross section, σ_a, is the probability that the moderator will absorb a neutron when it comes into contact with it, this should be as low as possible to prevent the neutrons being absorbed by the moderator as this prevents subsequent fission events from occurring.

The moderating ratio, ζ, is a value that defines how good of a moderator a material will be, taking into account not only just the slowing down power, but also the absorption cross section. It is given by

$$\zeta = \frac{\eta}{\sigma_a} = \xi \cdot \frac{\sigma_s}{\sigma_a} \tag{12}$$

Nuclear reactors are incredibly complex machines designed to convert the energy released from nuclear reactions to useful energy. Safety is paramount when ensuring that a sustained and controlled reaction occurs, as otherwise a chain reaction can get out of control and cause an explosion. The key components of a nuclear reactor include the fuel assemblies, containing enriched fuel, commonly uranium or plutonium, a moderator, which will slow down the fast energy neutrons to thermal speeds, and a coolant system to absorb the heat generated during the fission reactions. There are several types of fission reactors, one of the most common is a boiling water reactor (**Fig. 1**). This type of reactor uses water as both a coolant and a moderator and the steam that is generated from boiling the water is used to turn a turbine.

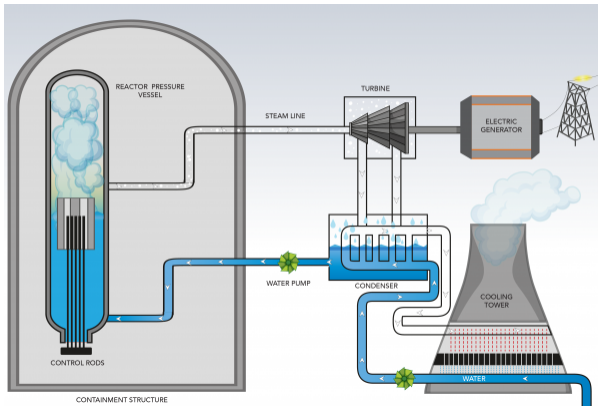


Figure 1: Shows the basic schematic of a boiling water reactor (BWR).[1]

The coolant must be selected carefully as energy will be lost heating the coolant and then boiling it so that the steam can turn the turbine. This energy lost can be quantified using the specific heat capacity, c, and the latent heat of vaporisation L_v.

The specific heat capacity, c, is defined as the amount of energy, Q, required to raise the temperature, T, of a 1 kg mass, m, by 1 K

$$c = \frac{1}{m\Delta T} \cdot Q \tag{13}$$

and the latent heat of vaporisation, L_v is the energy required, Q, to change the state of a mass, m, of

1 kg from liquid to gas [2]

$$L_v = \frac{1}{m} \cdot Q \tag{14}$$

If nuclear reactors fail to be properly moderated and controlled, it can lead to devastating consequences. For example, the Chernobyl disaster lead to the meltdown of reactor 4, causing a core explosion, resulting in open fires and extensive radiation fallout. The effects of this catastrophe are still evident today, with Pripyat, the nearest town, being uninhabitable. It will be more than 3,000 years before it becomes safe for humans to inhabit the region again [3]. Whilst the initial impact of the explosion resulted in the death of only 28 individuals, according to the World Health Organization, the Chernobyl disaster is linked to over 300,000 deaths, attributed to radiation poisoning, burns, elevated cancer rates due to exposure and DNA damage. It's estimated that more than 7 million people were exposed to radiation as a result of this tragedy [4].

Figure 2: This figure shows the aftermath of reactor 4 and the collateral damage it caused.[5]

After the disaster, when scientists simulate the real-world use of a nuclear reactor, they must ensure that they can effectively control it through all required moderation and control mechanisms within the reactor.

2 Method

The goal of this project was to design a tool that would allow the modelling of a nuclear fission reactor. Furthermore Pygame will be used to visualise and make the program interactive, using Object-Oriented Programming to implement the control rods and moderators for the visualisation.

In order to complete this, it was necessary to split into three groups. One of the groups would complete

the research required for the project, one group would code the nuclear calculations using Numpy and Pandas, and then the final group would put together the visualisation tool. It was decided to break the problem down into smaller tasks and complete them at the same time so that the project could be completed in the time available.

Given this, the following approach will be taken. First the research carried out will be spoken about, followed by the nuclear calculations and modelling, and then finally the visualisation will be discussed. This is so that each section can have the appropriate depth and amount of discussion about how it was carried out.

Initially we wanted to test if the neutrons travelled at relativistic speeds, we calculated the Lorentz Factor for the first collision (**Eqn. 5**). If the calculated velocity of the neutrons is larger than about 80% the speed of light then it is important to consider the effects of special relativity for the path of the neutrons, subsequently neutrons may also be relativistic, but if the first neutron has a velocity less than this, special relativity will not need to be considered.

The next section was to determine the thicknesses of materials required in the fission reactor, such as the moderator and the fuel rods. It is important to consider both the mean free path (**Eqn. 8**) and the number of collisions (**Eqn. 10**), as this will allow for a best estimate of the required thickness of each component to be considered. The motion of the neutrons has been assumed to be in 1 dimension as over the high number of reactions, the net trajectory has little affect. The number of collisions can be estimated for a material and then multiplying this by the mean free path gives the average distance travelled inside the medium. For example, in the moderator this would be the average distance for the neutron to reach thermal speeds [6]. Following these calculations, we looked at the instability of the daughter products produced. As a consequence of their high N:Z ratio, subsequent high energy β^- decays occurred. The decay paths of each and every daughter product was found [7] until they lie on the line of N:Z stability.

This leads us to sufficiently modelling nuclear fission which requires a number of equations, and as a result, we had to choose the factors that contributed the most to the outputted energy. We chose to include calculation for the logarithmic energy loss, the number of collisions to thermalise, slowing down power, moderating ratio as well as normalised probability calculations for each decay, Q-value calculations and a consideration for the efficiency of both the reactor and the generator.

The moderating ratio represents a value that summarises the aptness of a moderator. By this we mean the ability of a moderator to decrease neutrons to thermal speeds, without completely absorbing them, thus controlling the chain reaction. We investigated the three most common moderators: Water (H_2O), Heavy Water (D_2O) and graphite [8].

We started by calculating the necessary parameters that contribute to the moderating ratio, starting with defining a function for the logarithmic energy loss (**Eqn. 9**). This represents the average energy loss per collision between neutrons and the moderator. We then subsequently defined a function for the slowing down power, which quantifies the ability of the moderator to slow down neutrons to thermal speeds (**Eqn. 11**). A third function was then defined for the moderating ratio, which combined the outputs from the previous two functions with experimentally validated values for both the absorption and scattering cross sections (**Eqn. 12**). Also it was required for us to calculate the number of collisions to bring the neutrons

to thermal speeds. For efficiency, it is more optimal to have fewer collisions because the chain reaction can continue at a higher rate (**Eqn. 10**).

Following all the functions created, the value of each parameter was calculated for the three moderators. Each output was saved to a Pandas data frame, alongside the absorption and scattering cross section.

Following these calculations this leads to a brief part of the code entailing values regarding coolant information and displaying them in a Pandas data frame. The function of a coolant is to transfer the heat energy produced by the nuclear decay, and transfer it to the generator which produces electricity. The three most common coolants are H_2O, D_2O and CO_2. The coolant information that was loaded into the Pandas data frame was the specific heat capacity and the latent heat of vaporisation for each coolant.

In order to connect the theory associated with beta decays to our simulation, we chose the 16 most probable decay paths of ^{235}U fission, the process of nuclear fission could be accurately simulated. Each decay had an associated decay equation, an example of which is shown below:

$$^{235}_{92}\mathrm{U} + ^{1}_{0}\mathrm{n} \rightarrow ^{95}_{40}\mathrm{Zr} + ^{138}_{52}\mathrm{Te} + 3\ ^{1}_{0}\mathrm{n} \implies P(\text{reaction}) = 8.14\% \tag{15}$$

To calculate the Q-value for each possible decay, we defined a function that calculates the Q value for each reaction (**Eqn. 2**). The atomic masses of each daughter nuclei were researched and loaded into a .csv file to be imported into the code. Creating a for loop allowed us to calculate the Q-value of each reaction by passing the Q-value function over the atomic masses for each daughter product pair. The results were then converted into Joules.

Each decay path had a corresponding probability, representing the likelihood of the fission releasing those specific daughter products. The probabilities were then normalised by finding a normalisation constant, which was done by calculating the inverse of the sum of the probabilities, then multiplying the probabilities by this constant. This allowed us to appropriately adjust the probabilities so that the total probabilities summed to one. This was required as we had only chosen the sixteen most probable reactions [9], so this total was not equal to one.

To arrive at a finalised energy output for the reactor, a fission chain reaction had to be simulated by randomly choosing a series of reactions and summing their calculated Q-values. This started by defining a set of variables and empty arrays. We chose a loop with depth of twelve as this was the maximum number of generations that CoCalc could manage without causing an absurdly long run time or the program to run out of memory. A random energy is generated for each induced fission in the decay chain, followed by summing the energy from each reaction to calculate the total energy released from a chain reaction of depth twelve ($\sum_{i=0}^{12} 2^i = 8191$ reactions). The total released energy is then multiplied by an energy scale factor, to account for CoCalc only being able to run twelve iterations in an appropriate length of time. The energy scale factor was determined by taking into account the energy required to heat and vaporise the coolant water and also the energy lost due to the efficiency of the turbine. The scale factor ensures that the remaining power in the simulation is of the correct order of magnitude, 1 GW[10], as the power produced by a nuclear fission reactor. An adjustment also had to be made considering the varying position of the control rods within the nuclear reactor. By analytically deducing a reciprocal model of $(x + 0.01) \cdot y = 0.01$, where x is the insertion depth and y is the percentage of energy loss, and the factor of 0.01 is added to x so that when there is zero insertion, the power output isn't an infinite value, we were

able to model the power output at different control rod insertion depths. For example, at an insertion depth of 50%, the power output dropped to 2% of the maximum power output. Consequently, maximum power output occurred at an insertion depth of less than 20%. The specific insertion depth can be altered in the user interface in the Pygame visualiser.

To begin the Pygame, we had to code a start screen. The vision was to allow the user to select a moderator from a list of predetermined moderators. It was decided to give the choice of three possible moderators: H_2O, D_2O and Graphite. The buttons were created by defining a surface of the size that we wanted the button, and then getting the associated rectangle so that we could check for collisions and change the colour, with all required colours being defined as constants at the start of the program. Collisions are checked by first checking if the left mouse button has been selected, if it has the mouse coordinates are retrieved and then if the mouse is colliding with one of the rectangles belonging to a button then the corresponding event happens.

The four buttons that are on the start screen are 'Water', 'Heavy Water', 'Graphite' and 'Click to Start'. The buttons were implemented in such a way that the start button couldn't be pressed until an option for a moderator was chosen, which was accomplished by making a NoneType variable corresponding to the chosen moderator, and the start button could only be pressed once this variable was not a NoneType. Initially we were just going to make it so that the buttons were static and didn't change when clicked, but we later decided to make it so that the user could see which moderator they have currently selected by making the button be a light grey colour.

Once the start menu had been implemented, the next step was to create the main visualisation tool. This was decomposed into several smaller tasks so as to make it easier to complete. The first of these was to implement a help button. This would describe the process of nuclear fission to anyone who was unsure of what it meant, as well as giving a visual picture of what was happening at a sub-atomic level. There was also a back button that takes the place of the help button, whilst on the help screen, so that the user can return to the main tool. Another goal was to make it so that the user could press a return button and get back to the start screen in order to select a different moderator.

In order to allow the user to switch between windows, functions were created for each window the user can see. Each time a button is pressed that moves the user between screens, the corresponding function is called. For example, if the user is on the start screen and they press the start button, the Visualiser function will be called which will swap the game loop from the start screen to the visualisation tool.

The next step was to create an image that would include all non-moving parts of the fission reactor. This was so that it would be easier to code as it would be quite tricky to implement the design of a reactor using just Pygame due to the nature of how images are displayed on the screen. This leaves the parts that need to be implemented as the fuel rods, the control rods, the turbine, as well as all the text on the screen. Both the fuel and control rods were classes that inherited from the Pygame sprite class, and added to a sprite group so that they can be easily updated and be looped over.

The fuel rods are instantiated from the FuelRod class one at a time and added to the corresponding sprite group. The fuel rods are placed on the screen by having a coordinate for the top left of the first control rod, and then each subsequent fuel rod is placed at these coordinates plus an offset based on which control rod it is.

The control rods are instantiated from the ControlRod class and the same method is taken when placing the control rods on the screen as the fuel rods. The only difference is the ControlRod class contains two new methods that the FuelRod class does not. These are the methods responsible for moving the control rod up and down. When the user presses one of the buttons that change the insertion of the control rods, the first thing that is checked is whether the change in insertion will bring the control rods outside of the allowed range of 0-100%. If the move doesn't take it outside of the valid range, then the control rods are moved by the appropriate amount. The variable that is used to define the insertion percentage is the same as the variable used to calculate the reduction in energy due to them being inserted.

The turbine is currently modelled by having an image of a turbine [11]. It was set to have angle step that depends on the time period of 1 rotation as well as the frames per second, with this angle step added to the current angle of the turbine every frame to give the effect of a smooth continuous rotation. This angle step was multiplied a factor proportional to the insertion percentage so as to make the rotation slow down as the control rods are inserted more.

For placing the text on the screen, there were be three size fonts generated. Each of these fonts were used to create a surface containing the text, and then the rectangle is retrieved from the surface. When there is a text element that needs to have its text changed, such as the insertion percentage, an f-string is used to allow the variable to be added into the Pygame text. This was also done for the power output text, taking the power output from the nuclear calculations and displaying it on the screen.

3 Results and Discussion

After calculating the moderating ratios for our three potential moderators, we presented our findings in Table 1.

Moderator	σ_S (m^{-1})	σ_A (m^{-1})	ξ	No. Collisions	η	ζ
H_2O	165	2.2	0.9	19.6	153	70
D_2O	36	0.012	0.5	35.7	18.4	1530
Graphite	39	0.003	0.2	115.3	6.2	2051

Table 1: Shows the moderator data that was used to determine the moderator used in the nuclear reactor. σ_S is the scattering cross section, σ_A is the absorption cross section, ξ is the logarithmic energy loss, η is the slowing down power and ζ is the moderating ratio.

Although in the table it seems that graphite is the most appropriate moderator, from wider research the actual value of the moderating ratio is an order of 10 smaller than the value calculated. The calculations in the code were verified externally, and the reason for the discrepancy could not be ascertained, so it was decided that the excepted value for the moderating ratio of graphite would be used instead of the anomalous value we calculated. This informed our decision to use heavy water as a moderator, as it had the highest moderating ratio. Heavy water also had the fewest number of collisions for the neutrons to reach thermal speeds, further justifying it as the preferable moderator choice.

The coolant information was loaded into Pandas data frames and is displayed in Table 2.

Coolant	L_v ($\mathrm{kJ\,kg^{-1}}$)	c ($\mathrm{kJ\,kg^{-1}\,K^{-1}}$)
H_2O	2260	4.18
D_2O	2073	4.23
CO_2	571	0.85

Table 2: Shows the cooling data that was used to determine the coolant that will be used in the modelled reactor. L_v is the latent heat of vaporisation of the substance and c is the specific heat capacity.

In a boiling water reactor, the coolant and moderator are the same material. As coolants there is little difference in specific latent heat of vaporisation and specific heat capacity between heavy water (D_2O) and regular water (H_2O).However, this slight difference in suitability makes heavy water the clear winner as a dual-purpose moderator and coolant. Even though there is little difference, a high specific heat capacity and a low latent heat of vaporisation are both optimal characteristics for a coolant, which further supports the use of D_2O as the preferable coolant. There was some discussion about whether to consider real world complications, like cost or feasibility, but in the end we decided to ignore these, as we are not confined to budget constraints in this project.

The nuclear calculation simulation outputted our desired power of 1 GW, which is the average power output of a nuclear power plant. This power output is then stored as a variable and imported into the Pygame where it can be displayed in the simulation to show the power output of the nuclear reactor at different insertion depths of the control rods.

The Pygame section of our nuclear fission reactor simulation was relatively successful as we managed to visualise the process of our nuclear reactor whilst making it interactive and informative for the user. The simulation begins with a start screen (see **Fig. 4a**) that allows the user to choose a moderator for the simulation from a selection of 'Graphite', 'Heavy Water' and 'Water'. We were unable to implement the different moderators changing the result from the simulation. However, if given more time we could add this into the code to improve the simulation.

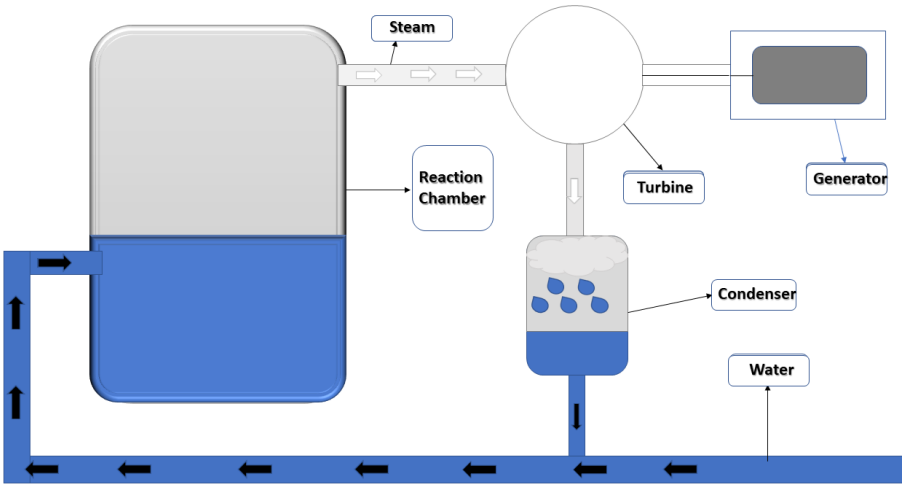
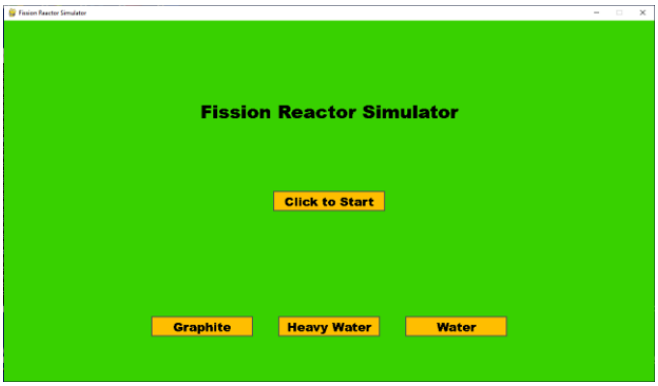


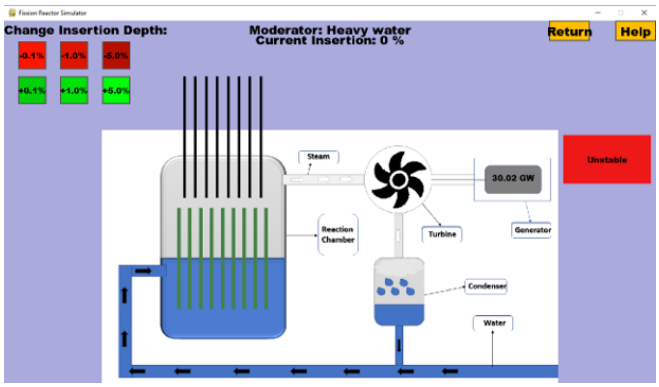
Figure 3: Shows a screenshot of the nuclear fission reactor template that was used in our Pygame simulation.

The template that we used for the reactor simulation (**Fig. 3**) was based on a Boiling Water Reactor (BWR) (**Fig. 1**) and was made in PowerPoint with the relevant labels applied in order to inform a user, who may know nothing about nuclear reactors, what each component is. The help screen was very successful and allows the user to press the help button and try and understand what a fission reactor does as well as what the reaction looks like. If this was to be redone it might be better to summarise the text into bullet points or less text as it is currently just a large block of text.

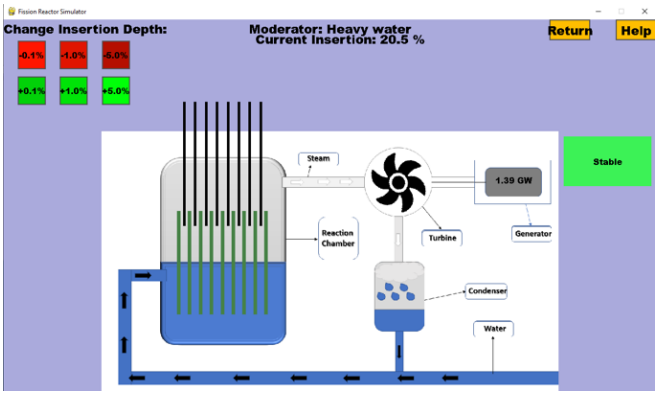
The most successful interactive portion of the simulation is the control rods (**Fig. 4b, 4c, 4d**). The functionality is exactly what was required to be implemented, other than the fact that if the control rods are inserted to any percentage, and the return button is pressed, when the start button is pressed again, even if the insertion percentage at the top of the screen says that the rods are not inserted, the sprites are still at the insertion depth they were before the return button was pressed. This is only a visual thing as the power output is based on the insertion variable not the physical location of the control rods but it could be fixed by refactoring how the code works with respect to drawing the control rods.



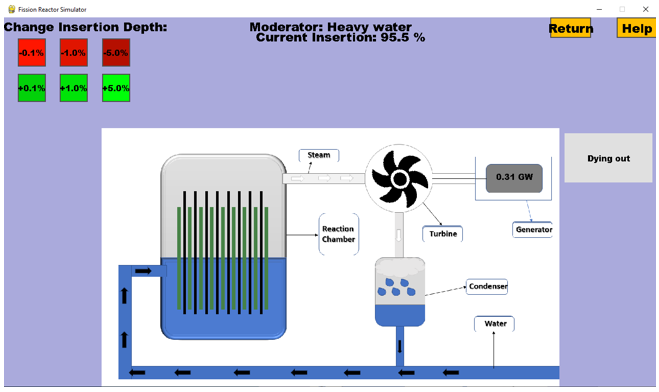
(a) Shows a screenshot of the start screen of our nuclear fission reactor simulation.



(b) Shows a screenshot of our nuclear fission reactor simulation with control rods at 0% insertion. The corresponding power output is 30.02 GW.



(c) Shows a screenshot of our nuclear fission reactor simulation with control rods at 20.5% insertion. The corresponding power output is 1.39 GW.



(d) Shows a screenshot of our nuclear fission reactor simulation with control rods at 95.5% insertion. The corresponding power output is 0.31 GW

Figure 4

As the power output doesn't increase over time with the control rods removed, due to the way that we have had to model the power output, a visual indicator has been added which shows the user whether the chain reaction is stable, unstable or dying out. This tells the user what would be happening to the chain reaction if a real nuclear reactor was to have the control rods inserted to this depth.

One of the aspects we planned to incorporate into this project was the subsequent beta decays after the reaction. The daughter products of the beta decays release approximately 10% of the fission reactions energy. In total we calculated 38 subsequent beta decay and their daughter products. As an example two of the beta decays are shown:

$$\mathrm{U}_{92}^{235} + \mathrm{n}_{0}^{1} \rightarrow \mathrm{Zr}_{40}^{95} + \mathrm{Te}_{52}^{138} + 3\ \mathrm{n}_{0}^{1} \tag{16}$$

We then calculated the subsequent beta decays for the two daughter products seen above

$$\mathrm{Zr}_{40}^{95} \rightarrow \mathrm{Nb}_{41}^{95} + \beta^{-} \implies 95.9081u \text{ (Mass of daughter particle after decay [7])}$$

$$\mathrm{Te}_{52}^{138} \rightarrow \mathrm{I}_{53}^{138} + \beta^{-} \implies 137.9281u \text{ (Mass of daughter particle after decay [7])}$$

We also wanted to use Pygame to visualise the β^- decay paths of daughter products back down to the N Z stability line. However this was a low priority task and due to the complex restraints of the calculations involved with completing this task it couldn't be completed.

Another task we aimed to complete was to calculate the efficiency of the turbine generator from scratch. However due to the extreme conditions in the reactor, such as high pressure, it was much more difficult to calculate energy consumption in heating and vaporising the heavy water, as the specific heat capacities and latent heat of vaporisation vary considerably under these conditions. Instead we simplified this calculation with a scale factor in the code.

We initially aimed to simulate the trajectory of the fission neutrons, however with the dimensions of Pygame this would be difficult to get an accurate reflection for this. Firstly, we considered these trajectory's in 3D, however it quickly became apparent that it was really complex to construct the nuclear reactions in 3D. Therefore we planned to keep the calculation consistent with the visitation tool by considering them in 2D. We aimed to simulate their trajectory by randomly generate an angle between 0° and 180° and then calculating whether or not that neutron will be incident on any fuel rods. However this proved complex as we would have to convert the cross section into 2-Dimensions as well as other calculations. As an alternative we assumed that all neutrons are incident on the next fuel rod. This was a reasonable assumption because of the high volume of occurring reactions, meaning that over the course of the simulations, the trajectories would even out.

Using controlled chain fission has many positives for the world we live in today, however with so much energy comes a negative. The main issue with controlled nuclear fission is the disposal of radioactive waste. The daughter products of fission are especially unstable and need to be vitrified and sealed in concrete, then buried 200-1000 metres deep underground for around 50 years. The location of storage is massively important, it must be in a geologically stable area, so earthquakes don't crack open the concrete casing and also dry, so that water sources aren't irradiated. Our desired location would be in Nevada or Arizona, but in Nevada especially, the population of the state are understandably wary of nuclear waste being in the vicinity of their homes.

4 Potential Extensions

Simulating a nuclear fission reactor can be a very complex task without ignoring or disregarding certain factors to simplify the simulation. This, of course, means that the simulation itself will be less accurate. For our project, we didn't have the time to produce a 100% accurate simulation, so there is a number of things that we could add to our project in order to improve it.

For example, any subsequent decays after the initial fission were not considered. When the ^{235}U source decays by the absorption of the neutron, there are a multitude of different decay paths that it can take, each with different daughter products and Q-values, and each daughter product with its own decay path to the N:Z line. If we were less constrained by time, we could code these decays into the simulation,

providing a more accurate simulation and power output.

Additionally, we could add more visual effects to the Pygame, such as gauges that show the fission rate and turbine output. These dials would be semi-circular, with a green section and a red section. This would demonstrate to the user that if the rate of fission is too high, and hence in the red zone, then they must insert the control rods deeper into the reaction chamber in order to stabilise the reactor, thus moving the pointer into the green section. Both of these dials will depend on each other because if the rate of fission is too high then the turbine output will be too high for the reactor to deal with and will, therefore, be in it's own respective red zone.

To do this we would create the semi-circle backgrounds of the dial in PowerPoint, or find an image online if possible. We would then have to code the hand on top of the dial background in a similar way to which it was coded in the turbine. We would have to code the dial to be related to the quantity it is supposed to show, for instance, the turbine output dial would have to be coded as a calculation to give us an angle of rotation that is related to the power output of the turbine.

$$180° \cdot \left(\frac{\text{Current Power Output}}{\text{Maximum Power Output}} \right) = \text{Rotation Angle of Dial} \tag{17}$$

This equation demonstrates how we would codeg the rotation of the dial. The fraction of the power output is multiplied by 180°, because the Pygame rotation function works in degrees and we want our semi-circular dial to rotate across a region of 180°.

We also could make either the screen flash red or code lights that flash red when the reactor is in an unstable state for a longer period of time. Warning messages could be added along with this, in order to make the simulation more realistic and even more interactive for the user. To do this we could define a surface in Pygame, make it red and then make it 30% transparent, by changing it's alpha value, and then set it on a timer to flash over the simulation every 2 seconds.

Another extension to the project that would add another layer, is to include the premise that the mass of ^{235}U atoms in the fuel rods is going to decrease as a function of time as it gets used up. In our simulation, we have not given a starting mass in the fuel rods for ^{235}U . This was not added to our simulation as the fuel rods contain several tons of ^{235}U and in our simulation it would have to run for several years for the fuel to be extinguished. Our planning didn't allow a time scale for this. However, to make this simulation more realistic, this should be included to improve the model layer. To derive this we would have a differential equation that relates the starting mass of the fuel rods to time. This would allow a function to be created that subtracts the original mass and apply this to the game loop. This also relates to heavy water as the selected coolant, which was chosen due to its high specific heat capacity, despite ignoring real-world complications like cost. In future we would apply a feature in the game that allows the user to change the moderator and coolant to other variations discussed in this report. Not only would this add another layer to the model, it would provide a more applicable use to the real world as lower cost materials can be used and proven in our simulation.

5 Conclusion

The simulation successfully achieved the desired power output of 1 GW, representing the actual average power output of a nuclear reactor. Allowing the boron control rods insertion depth to be altered, meant the Pygame visualization of the fission reactor was interactive. This was done by creating an interactive tool, using Python 3.10, that allows the visualisation of a nuclear fission reactor, with the power output being changed by user input in increments of 0.1%, 1.0% and 5.0%. The tool was tested to ensure that it reproduced the power output seen in real fission reactors, and was deemed a partial success due to the power output being close to that of a real reactor. However, planned features like mean free path, beta decays, trajectory simulation, and turbine efficiency calculations were either partially implemented or simplified.

The choice of moderator was initially based on the calculations completed in the code, with graphite seeming the most optimal moderator. However, due to discrepancies in the graphite moderating ratio final value, heavy water (D_2O) was chosen as it has the highest moderating ratio and requires the fewest collisions for neutrons to reach thermal speeds. The values of latent heat of vaporisation and specific heat capacity were similar for both H_2O and D_2O, with the latter being slightly more preferable. This, alongside the moderator and coolant normally being the same material in a boiling water reactor, led to D_2O being the chosen coolant.

Drawbacks of nuclear fission reactors, such as radioactive waste disposal and reactor instability, were discussed. The report concludes by suggesting potential extensions to improve the simulation's accuracy, such as including subsequent decays, adding visual effects for reactor monitoring, and accounting for fuel depletion over time.

The simulation offers valuable insights into nuclear reactor dynamics, but further improvements are necessary for a more accurate representation of real-world scenarios. As previously mentioned this includes the beta decays following the initial fission reactions, to include an original fuel mass function that is time dependant representing that fuel decreases over time and, in turn, the rate of reaction will decrease.

References

[1] World Nuclear Association. Pressurized water reactor diagram, Accessed: 2024.

[2] University of liverpool. Online, 2022. Chapter 16 - An overview of heavy water reactors, ScienceDirect.

[3] Kristine Brennan. *The Chernobyl Nuclear Disaster*. Chelsea House Publishers, Philadelphia, PA, 2002.

[4] Andrew Blackwell. *Visit Sunny Chernobyl and Other Adventures in the World's Most Polluted Places*. Rodale, New York, 2012.

[5] CBS News. Horrifying photos of chernobyl and its aftermath, Accessed: 2024.

[6] Nuclear Power. Fission neutron, Accessed: 2024.

[7] PeriodicTable.com. Tellurium-138 isotope, Accessed: 2024.

[8] N/A. Phy135 - week 3 - radioactive decay processes - natural sources of radioactivity, Accessed: 2024.

[9] Fission product yields of ^{233}U, ^{235}U, ^{238}U and ^{239}Pu in fields of thermal neutrons, fission neutrons and 14.7-mev neutrons. page 2977, 2010.

[10] U.S. Department of Energy. Nuclear power: Most reliable energy source (and it's not even close). `https://www.energy.gov/ne/articles/nuclear-power-most-reliable-energy-source-and-its-not-even-close`. Accessed on: May 10, 2024.

[11] Turbine png.

[12] Pygame Community. *Pygame Documentation*. Pygame, 2022.

[13] Travis Oliphant. A guide to numpy. `https://numpy.org/doc/stable/user/absolute_beginners.html`, 2006.

[14] Eric Jones, Travis Oliphant, Pearu Peterson, et al. Scipy: Open source scientific tools for python. `https://www.scipy.org/`, 2019.

[15] Jeffrey Reback et al. pandas-dev/pandas: Pandas, 2020.

6 Appendix

A Modules Used

Please find below the modules that were used to create this program.

1. Pygame [12]
2. Numpy [13]
3. Scipy [14]
4. Pandas [15]

B Contributions

B.1 Jack

Overall in this project I was involved in the research and development of this project. In the early days of this project I made the first draft of the help screen in python using paint. The following weeks the group was more organised and I research the dimensions and cross section of a nuclear reactor, as well of the dimensions of the fuel and control rods. Also I researched the beta decays and calculated 36 subsequent beta decay and their masses. Towards the end of the project I took the lead in making the power point and the presentation alongside Jake who edited the video and audio. In the report, I contributed towards the abstract, introduction, results and discussion as well as the references.

B.2 Jake

I mostly took a back seat when it came to the research behind the theory of a nuclear reactor, although some stuff was still ran past me such as clarifying what an equation meant, but I made up for this when coding the Pygame and helping structure the report. I completed all of the Pygame code apart from the start screen, from coding the classes for the fuel and control rods to defining the constants for the colours. I also commented the whole code and added markdown cells explaining what was being done. I proofread the whole report and helped restructure sections such as the introduction so that they flowed in a logical order. I also helped Jack with the powerpoint and brought in my microphone and laptop so that we could record the audio on it, and then I edited the audio so that it flowed nicely and any stutters or “uhms” were removed.

B.3 Daniel

I was responsible for large parts of the initial research of theory, before focusing mainly on the aspects of the moderators, i.e. moderating ratio calculations, and coolants, i.e. specific heat capacity and latent

heat of vaporisation, and then adding this into our code. I then assisted Jake and Oscar in the rest of the code. Oscar and I created the section of the powerpoint that concerned our section of the code and talked about it in our presentation too. We also wrote the coding method, results and discussions sections of the report, apart from the Pygame sections. Also, contributing to large parts of the introduction and conclusion in the report and altering aspects of the report to ensure it flowed clearly and contained the key aspects of the project.

B.4 George

I was responsible for the theory behind the project, alongside the nuclear calculation for the Q values of the initial fission's and their normalised probability of occurring. This also led to the determination of all corresponding daughter products which were aimed to be used later in the calculation of the beta decay energy. I was also involved in the report writing, introduction and possible extensions.

B.5 Will

During this project my main responsibility was to develop the pygame code alongside Jake. I took a backseat when it came to the theoretical part of the project however I did use the search and replace function to rename a lot of the variables in Colac to make the code more readable. When it comes to the pygame I coded the start screen, made the background for the simulation with all the labels for the different parts of the reactor. I also made the help screen including the text on there describing the fission reactor process. In the video I was responsible for writing and presenting the 'Potential Extensions' section. I contributed to the report significantly mainly writing about sections which I had participated in during the project such as the potential extensions as well as the pygame sections.

B.6 Oscar

My role mainly entailed coding the nuclear calculations necessary for the simulation. After being given a set of fission reactions and there associated probabilities, I had to research the mass of all 34 reactants and products, amalgamating the masses and probabilities into csv files so they could be analysed. I wrote the code for the simulation that took into accounts all the factors we considered and generated a desired power output. Subsequently, I contributed to the part of the report and video presentation that concerned with the code I had written, and Dan and I worked in conjunction throughout the project as our roles had significant cross-over.

B.7 Jay

Since superior coders were in the group and nuclear science is currently my main field of expertise in physics, I mainly contributed to theory and research of the reactor. This included finding the probabilities of fission products being produced in fission and also, using conservation of baryon number, the corresponding other daughter nuclei. In our attempt to make a to-scale reactor, I had to consider dimensions of the reactor and its components. This led to me researching calculations to find the thickness of moderator and control rods to ensure interactions occurred, so that if our 2D cross section of our reactor was scaled up, it would theoretically look similar to that of a real-life working reactor. I took a lead in the addressing and subsequent simplification of issues too complicated for us to consider, such as neutron trajectories and the use of macroscopic instead of microscopic cross sections. In the report, I consequently

wrote mostly about the above theory, as well as its relevance in the method and the discussion.

C Project Proposal

C.1 Goals

The project aims to simulate a fission reactor by combining a series of nuclear calculations, using Pygame to provide an interactive interface that allows the user to control elements of the reactor. The Pygame will be able to control the insertion depth of the control rods as well as the choice of moderator, fuel and coolant. Our project will allow us to apply various python techniques and modules that we have learnt over our last eighteen months at university, combined with our knowledge of nuclear physics, hopefully culminating in a successful and accurate simulation of a fission reactor.

C.2 Theory behind project

Nuclear fission is the main bit of theory that will underpin this project. We aim to consider the energy of fission via using Δm values as well as using the cross sections to model to-scale the moderation of neutrons, control rods' absorption of neutrons and of course the absorption of neutrons by the U-235 fuel rods. We will also spend the first couple of weeks researching more into nuclear reactors to fill any gaps in knowledge we have, and also to account for as many relevant variables as possible.

C.3 Nuclear Calculations Method

The primary goal of the code is to calculate as many factors that contribute towards the fission reactor, and then combine the results into one a single output that represents the power output of our reactor. We will also visualise some of our inputs using Pandas, collating necessary data into tables. To process the nuclear calculations, we will define a series of functions and loop them to provide a final result for the power output

C.4 Pygame Method

The Pygame will be split into several sections, having different screens for each section. The first section will be a start screen, containing buttons that will allow the user to select what material will be used for the moderators, coolants and fuel type. The second section will be the main visualisation screen. This will contain an image of a nuclear reactor and then Pygame sprites will be used to represent the control rods and fuel rods. The control rods will be able to have their insertion depth changed using buttons that will change the insertion in increments. There should also be a help screen that explains the process of nuclear fission as well as showing a visual representation of it.

C.5 Success Criteria

- Apply our understanding of nuclear fission to a real world reactor, taking into account a variety of theory.

- Aim to process the nuclear calculations in python using a variety of methods we have learnt so far at university.
- Produce an interactive, user friendly interface on Pygame that allows the user to change components of the reactor and observe how this changes the power output of the reactor.
- Work as a group to be adaptable and recognise that things will change throughout the project

D Help Screen

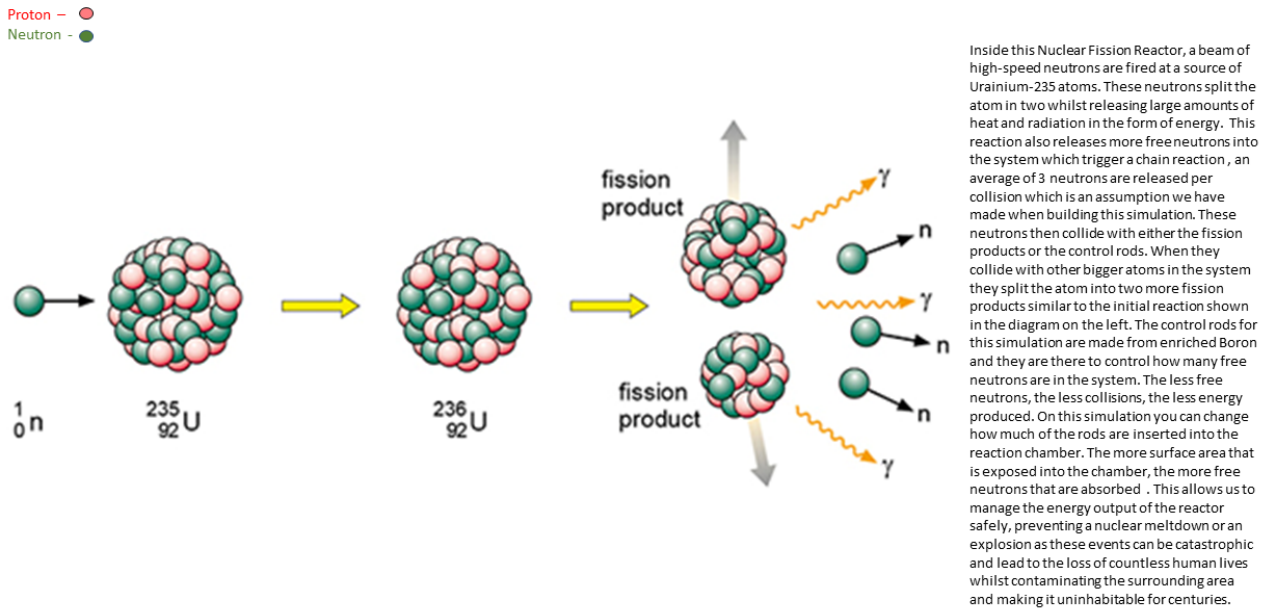


Figure 5: Shows the help screen that is shown to the user when they press the help button.