Determining neutron-nucleus potential through elastic neutron scattering

David R. Obee
(Dated: March 17, 2017)

In this work, we generate a variety of nuclear potentials. By comparing theoretical predictions using the partial wave scattering method, cross-sections of various nuclei with respect to the elastic scattering of neutrons are predicted and compared to experimental data. The Saxon-Woods potential is studied, along with more general potentials. We find that while potentials can be generated matching the cross-section data in some cases, this alone is not enough to say the generated potential is a valid model of the real potential.

I. INTRODUCTION

A. Scattering

Resonant scattering phenomena are of immense importance to particle physics research. In this study, elastic scattering of neutrons is considered.

First, we must establish what scattering is in particle physics. When we project a neutron towards a nucleus, we can expect that there may be some kind of interaction. The nucleus, being made up of protons and neutrons, has the potential to interact with the incoming neutron through the strong force. When this happens, the incoming neutron can be deflected, or as we will term it, scattered.

We can define a cross section σ for an interaction. When it comes to measuring this cross section, we need to look at how we detect a scattering event. If we set up a detector some distance away from the target, and in some direction θ, we can expect to only detect a fraction of any scattered particles. This defines the differential cross section $\frac{d\sigma}{d\Omega}$, where the number of detected particles is given by

$$D = nN\frac{d\sigma}{d\Omega}d\Omega, \tag{1}$$

where n is the number of targets per unit area, N is the number of particles per unit area per unit time travelling in the beam, and $d\Omega$ is the solid angle covered by the detector (in the limit of an infinitesimally small detector).

Hence, by integrating across all solid angles, we find $\sigma = \int \frac{d\sigma}{d\Omega}d\Omega$.

B. The partial wave method

We use the method of partial waves to predict the behaviour of our scattering system. This is a quantum mechanical treatment, given we are dealing with interactions between individual neutrons and nuclei. We will only be considering elastic scattering, i.e. processes where the incoming particles are the same as the outgoing particles.

The partial wave method begins with a potential $V(\mathbf{r})$, with $\mathbf{r}$ representing the vector from the target nucleus to a point in space. This describes the interaction between the incoming neutron and the target nucleus. In this case, it is a potential for strong force interactions.

We describe our incoming particles as non-localised plane waves, with the wave function

$$\psi \propto e^{i\mathbf{k}\cdot\mathbf{r}}. \tag{2}$$

Here, $\mathbf{k}$ represents the incoming neutron's wave number (essentially momentum). We can then describe the outgoing particles by the wave function [6]

$$\psi = \underbrace{e^{i\mathbf{k}\cdot\mathbf{r}}}_{\text{Plane wave}} + \underbrace{f(\hat{\mathbf{r}})\frac{e^{ikr}}{r}}_{\text{Modified spherical wave}}. \tag{3}$$

This description is composed of two parts, an outgoing plane wave, and a 'modified spherical wave'. This modified spherical wave is like a standard spherical wave emanating from the position of the target nucleus, however it is multiplied by a function $f(\hat{\mathbf{r}})$. This is a function of direction $\hat{\mathbf{r}}$ out from the position of the target nucleus, and it increases or decreases the intensity of the spherical wave in certain directions. This represents the fact that after a neutron is scattered, it is not equally likely to travel in all directions.

At this point, we can note an important symmetry in the solution to $f(\hat{\mathbf{r}})$. We assume that our potential is spherically symmetric (i.e. just a function of r). Consider we fire neutrons at our nucleus along a direction we define to be the positive-z direction. Using standard spherical coordinate notation with respect to the z-axis, we note that there is nothing here to favour any particular ϕ directions. Hence, the solutions to $f(\hat{\mathbf{r}})$ have to be azimuthally symmetric. As $f(\hat{\mathbf{r}})$ is also only a function of direction, this means that in general it is a function of (θ, ϕ), which in this symmetry becomes just a function of θ.

Following the method described in [6] we decompose this outgoing wave into a series of partial waves,

$$\psi = \sum_{l=0}^{\infty} R_l P_l\big(\cos(\theta)\big). \tag{4}$$

We can then solve the Schrödinger equation

$$\left[\frac{d^2}{dr^2} + \frac{2}{r}\frac{d}{dr} - \underbrace{\left(\frac{l(l+1)}{r^2} + V(r)\right)}_{\text{Effective potential [7]}} + k^2\right] R_l = 0, \tag{5}$$

where E, the incoming neutron's energy, is given by $\frac{\hbar^2 k^2}{2m}$.

This is a second order differential equation. We can separate this in to two coupled first order differential equations, which can then be solved numerically. We then derive a term γ_l, the derivative of the radial wave function, which can then be used to solve for a series of phase shift terms δ_l. We then sum across these phase shift terms, using

$$\sigma = \frac{4\pi}{k^2} \sum_{l=0}^{\infty} (2l+1) \sin^2(\delta_l) \tag{6}$$

to find the total cross section σ.

C. Resonances

The scattering cross section of an interaction can depend on the kinetic energy of the incoming projectile. This allows us to plot the scattering cross section against this energy, as shown in the example in Fig. 3. Notice the pronounced peaks in this curve at certain energies. These are termed resonances.

D. Deducing the nuclear potential

The aim of this work is to use the method of partial waves to predict the scattering cross sections of various shapes of potentials, at various energies. By comparing with experimental measurements, this allows us to rule out certain potentials, and improve the accuracy of others.

We will use optimisation algorithms to take generic functions with variable parameters, and optimise them such that they produce results consistent with the experimental data.

II. METHODS

Optimisation is the process of improvement. By taking a system with variable parameters, and some metric for how 'good' the system is, we can work to find a configuration of parameters that improves that metric.

Optimisation algorithms are a method of systematically improving this metric, or 'fitness function'. There are a wide variety of optimisation algorithms, all with certain qualities that make them more effective at some problems, and less at others.

Two types of optimisation algorithm are explored here. The first is a *genetic algorithm*, which is based upon the ideas of natural selection and evolution. The second is *particle swarm optimisation*, which is loosely based on how flocks of animals (i.e. birds) look for food [2].

Both algorithms were tested on the problems presented in this paper. It was generally found that for a given number of fitness function evaluations (the time-limiting factor in all of these problems), the particle swarm algorithm performed best. This agrees with previous work comparing these algorithms [3]. Hence, all of the final results presented in this paper have been calculated using this algorithm.

A. Genetic algorithm

Genetic algorithms can vary significantly in their implementation, but they tend to obey the structure listed in Appendix B. For clarity, definitions of the terms *solution*, *fitness function*, *crossover function*, *mutation function* and *selection function* can be found in Appendix A.

Naturally, there are a wide variety of choices that can be made in the implementation of this algorithm.

1. Crossover function

The crossover function employed is simply an average of corresponding parameters. Given two parent solutions $\{a_n\}$ and $\{b_n\}$, the child will be given by

$$\left\{ \frac{a_n + b_n}{2} \right\}. \tag{7}$$

E.g. the parents $\{1, 2, 3\}$ and $\{5, 6, 7\}$ would produce the child $\{3, 4, 5\}$.

2. Mutation function

The mutation function employed uses a normal distribution around the original parameter values of the solution. Given a solution $\{a_n\}$, the mutated solution will be given by

$$\{\text{Normal}(a_n, \sigma_n)\}. \tag{8}$$

That is, each parameter in the original solution will be used as the mean for a normal distribution, which then produces the new value, with some standard deviation σ_n which is preset.

Boundary values are also set for each parameter. If the normal distribution produces a value that is outside these bounds, then the final mutated value is set to be the value at the bound that was violated.

3. Selection function

The selection function used is a variation on the *roulette wheel selection* method. Roulette wheel selection is a way of randomly selecting members from a population, while giving more weight to 'better' solutions, and less weight to 'worse' solutions. An outline of roulette wheel selection can be found in Appendix C.

This selection method requires solutions be given weights. Of course, they need to be some function of each members fitness, as this is our measure of how 'good' a solution is. Throughout this work, we always try to minimise our fitness function, hence we wish to give more weight to smaller fitness values.

The fitness to weight mapping chosen is a function of both the individual member's fitness, and the overall population's fitnesses. The weight is given by

$$w = \exp\left(-\beta \cdot \frac{f - f_{min}}{f_{max} - f_{min}} \right), \tag{9}$$

where w is the assigned weight, β is a chosen dimensionless constant, f is the fitness of that individual member, and f_{min} and f_{max} are the minimum and maximum fitnesses of all members of the population respectively. A justification of the benefits of this mapping can be found in Appendix D.

B. Particle swarm optimisation

Particle swarm algorithms have gained wide use in optimisation problems, primarily due to their relative simplicity and effective results [2].

The general structure of a particle swarm algorithm can be found in Appendix E. This general structure leaves numerous aspects up to the specific implementation.

1. Boundary handling

All parameters are given boundaries. A *reflection* method is then used to handle members that try to leave these boundaries. For a given parameter i, we set the interval of acceptable values as $[a_i, b_i]$.

Before updating any position by its respective velocity, we check for each component to see if it will leave its bounds. If it does, then we reverse the velocity for that component. If the reversed velocity will also cause that component to leave its bounds, then we set that velocity component to be zero. This means that positions are updated by

$$x_i = x_{0,i} + \begin{cases} v_i \text{ for } (x_{0,i} + v_i) \text{ in } bounds_i \\ -v_i \text{ for } (x_{0,i} - v_i) \text{ in } bounds_i \\ 0 \text{ otherwise} \end{cases} \tag{10}$$

2. Inertia

In the velocity calculation (Eqn. E1), we have the constant ω multiplying the previous velocity. This dictates how much influence the previous velocity has on the new velocity. An ω value of one would mean that the all of the previous velocity is carried over into the new velocity. An ω value of zero would mean that the new velocity is in no way affected by the previous velocity.

The value of ω is useful in determining the relative weight of local search to global search [2].

3. Random constants c_1 and c_2

These constants dictate the effect that the *personal* and *global best positions* have on the change of velocity of a member. The value of c_1 determines the influence of the *personal best*, while c_2 determines the influence of the *global best*. Each of these are chosen from uniform distributions. Different ranges are used for c_1 and c_2, allowing the mean values to be controlled independently. This allows us to determine how much we favour local search to global search. If we have typically higher values of c_1 than c_2, the we will typically favour local search. Striking a balance between these two values can have a great deal of effect on the overall performance of the algorithm.

C. Fitness functions

The fitness function primarily used in this study is the *mean squared fractional error*. Given two data sets f and g defined across the values $\{x_1, x_2, ..., x_n\}$: $\{f(x_1), f(x_2), ..., f(x_n)\}$ and $\{g(x_1), g(x_2), ..., g(x_n)\}$, we define the *mean squared fractional error* as

$$\zeta = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{f(x_i) - g(x_i)}{f(x_i)} \right)^2 . \tag{11}$$

In this study, we always define the f values to be the experimental data, and the g values to be the data from the computational model. This is because we normalise ζ by the f values. Considering the limit as $f \to \infty$, $\left(\frac{f(x_i)-g(x_i)}{f(x_i)}\right)^2 \to 1$, while in the limit $f \to 0$, $\left(\frac{f(x_i)-g(x_i)}{f(x_i)}\right)^2 \to \infty$. This means that f values just a small amount below g are penalised very heavily, while those above g are never penalised greater than 1. Hence, it is sensible to choose f to be the experimental data, such that when the search algorithm is varying g, the ζ contribution is parabolic around $f(x_i)$.

When the two data sets do not match in their x_i values, linear interpolation is used.

D. Experimental data and the EXFOR database

In this study, experimental data is used to compare with the theoretical models. The data used is from the *EXFOR* database; a database of nuclear reaction data maintained by the *International Network of Nuclear Reaction Data Centres (NRDC)*, and coordinated by the *International Atomic Energy Agency (IAEA)*.

From this database, we can find experimental results from elastic neutron collision off a wide range of different nuclei, measuring their cross sections at a variety of energies. For constancy, only datasets with data taken at neutron energies of the order ~ 1 MeV have been used. This ensures that when we talk about the accuracy of a particular model, we are talking about its accuracy at a common energy scale.

E. Models of nuclear potential

We use a variety of different models for the nuclear potential. Each has a different number of variable parameters, resulting in varied optimisation runtimes.

1. Saxon-Woods potential

The Saxon-Woods potential is a widely used model for nuclear potential [4]. It represents an attractive potential, in the form of a square well with a diffuse shell. It takes the form

$$V(r) = -\frac{V_0}{1 + e^{\frac{r-R}{a}}}, \tag{12}$$

where V_0 represents the depth of the potential well, R the radial extent of the well, and a the thickness of the diffuse shell. The value of V_0 can be further written as $V_0 = V_1 - V_2 E - (1 - \frac{2Z}{A})V_3$, and R as $r_0 A^{\frac{1}{3}}$.

In this form, the potential is not constant with respect to the energy on the incoming neutron. It also adjusts itself for various nuclei, due to the Z and A dependence.

2. EDCRTn

A new general form of a possible potential presented here is the *Exponentially Decreasing Corrected r Taylor series of order n* (EDCRTn). This is of the form

$$V = e^{-\frac{\alpha}{c}r} \sum_{k=0}^{n} a_k \left(\frac{r}{c}\right)^k. \tag{13}$$

This takes the form of a Taylor series, scaled in r by a factor c, and multiplied by an exponential decay term controlled by α. For a length n, the number of variable parameters is typically $n+2$. This is taking into account all $n+1$ a_k values, and the value of α. We exclude the c value, as we can typically preset this based on what range of radial values we are considering in our calculations. If we say we consider the Taylor series in the range $[0, 1]$, then the scaling maps this on to $[0, c]$.

3. BEDCRTn

A modification on the EDCRTn potential is the *Bounded Exponentially Decreasing Corrected r Taylor series of order n* (BEDCRTn). This is the same as the EDCRTn potential, with the following modification

$$e^{-\frac{\alpha}{c}r} \to e^{-\frac{\alpha}{c}r} - e^{-\frac{\alpha}{c}c} = e^{-\alpha}(e^{\frac{r}{c}} - 1). \tag{14}$$

This serves a similar purpose as before, but now it forces the potential to be zero when $r = c$.

III. RESULTS

A. Optimising Saxon-Woods

Before applying an optimisation technique, it is important to understand how the fitness function varies across the parameter space. If we have a fitness function that has very little overall structure, then the effectiveness of the optimisation methods will be greatly reduced. In the case of no overall structure, at best any optimisation method will be as good as a series of random guesses. I.e. the genetic algorithm or particle swarm algorithm would not typically perform better over N fitness function evaluations than just selecting those N points randomly.

We begin by looking at the variation of the fitness function for the platinum data [9], using the Saxon-Woods potential. In Fig. 1 we see four parameter scans, with the colour representing the value of the fitness function. Of course, it is impossible for us to visualise the variation across the entire five-dimensional parameter space of the Saxon-Woods model, but we should get a qualitative idea from these four pairs of parameters. We see that there are overall structures in these scans, with regions of high and low fitness values. This implies that it is a sound method to apply our search algorithms, and we should expect better performance than a simple random search.

1. Fitting to platinum data

The Saxon-Woods model has found its most success in fitting the platinum data [9]. In Fig. 2 we see the final result of optimising the Saxon-Woods potential using the particle swarm algorithm. The computational results fit the experimental data rather well, particularly at higher E values. Overall, this achieved a mean squared fractional error of $\zeta = 0.002776$, which is very accurate.

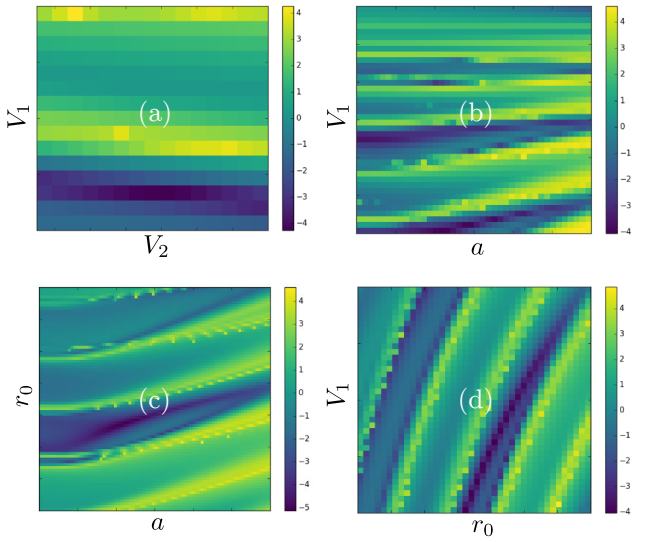

FIG. 1: Four parameter scan plots of the natural logarithm of the fitness function for the platinum data, using the Saxon-Woods model. Pairs of parameters have been chosen, with their ranges being the typical bounds used in the search algorithms. The plots are all logarithmic, due to the high variation in the fitness function which can span many orders of magnitude. In all of these, we see that there is an overall structure, with high and low regions, as opposed to a completely random distribution. This implies that it is viable to use search algorithms like particle swarm and genetic algorithms, as they should perform better than a simple random search, being able to make use of the overall structure.

2. Fitting to plutonium data

The Saxon-Woods model has mixed success when fitting to the plutonium data [10]. In Fig. 3 we see the final result of optimising the Saxon-Woods potential. Certain aspects of this are positive, e.g. the model matches the very high values of σ around $E = 1$ MeV. Furthermore, the value of σ is on the correct order of magnitude. However, it is clear that the overall shape does not match very well, with the computational result having far more pronounced curves. Plus, there are two resonances in the computational result that do not show up at all in the experimental data. The model suggests that given the resolution of the experimental data (distance between E values) these should not have been missed, suggesting that they do not in fact exist. This fit gives a mean squared fractional error of $\zeta = 0.0226$.

3. Fitting to oxygen data

The results of fitting the Saxon-Woods model to the oxygen data are very poor [11]. The shape of the fit in no way resembles that of the data. The only positive is that the results are on the same order of magnitude as the experimental data.

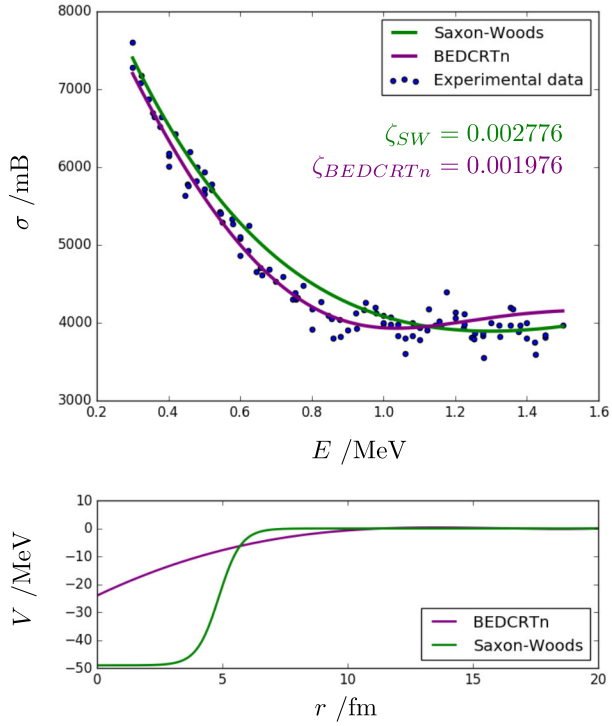

FIG. 2: The results of fitting the Saxon-Woods potential and the BEDCRT5 potential to the platinum data. The configuration of the Saxon-Woods potential is given by $\{r_0 = 1.286$ fm$, a = 0.4408$ fm$, V_1 = 50.38$ MeV$, V_2 = 0.4594$ MeV$, V_3 = 11.50$ MeV$\}$, with $\zeta = 0.002776$. The configuration of the BEDCRT5 potential is given by $\{c = 20$ fm$, n = 5, \alpha = 0.5867, a_0 = -54.14$ MeV$, a_1 = 120.8$ MeV$, a_2 = -6.541$ MeV$, a_3 = -58.22$ MeV$, a_4 = -12.53$ MeV$, a_5 = -6.798$ MeV$\}$, with $\zeta = 0.001976$.

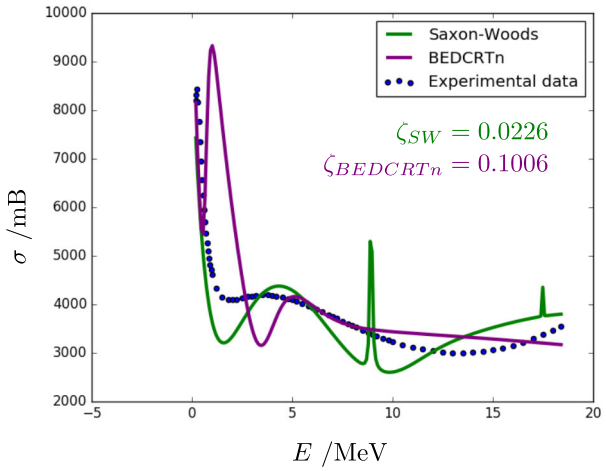

FIG. 3: The result of fitting the Saxon-Woods potential and the BEDCRT5 potential to the plutonium data. The configuration of the Saxon-Woods potential is given by $\{r_0 = 0.9605$ fm$, a = 0.3678$ fm$, V_1 = 118.4$ MeV$, V_2 = 0.6647$ MeV$, V_3 = 15.27$ MeV$\}$ with $\zeta = 0.0226$. The configuration of the BEDCRT5 potential is given by $\{c = 20$ fm$, n = 5, \alpha = 6.282, a_0 = -65.86$ MeV$, a_1 = 94.07$ MeV$, a_2 = 122.9$ MeV$, a_3 = -14.90$ MeV$, a_4 = -130.9$ MeV$, a_5 = 11.45$ MeV$\}$ with $\zeta = 0.1006$.

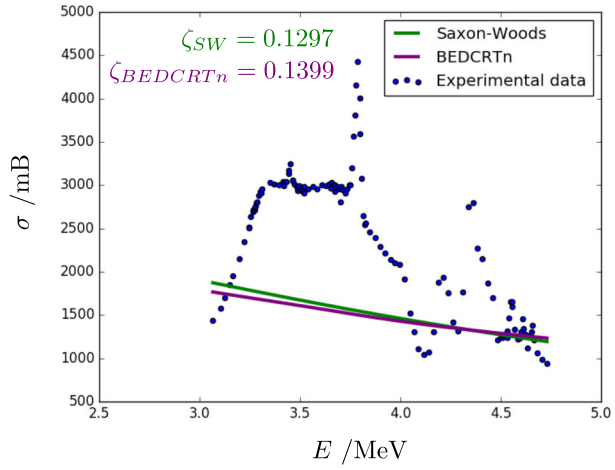

FIG. 4: The results of fitting the Saxon-Woods potential and the BEDCRT5 potential to the oxygen data [11]. The configuration of the Saxon-Woods potential is given by $\{r_0 = 0.9670$ fm$, a = 0.9052$ fm$, v_1 = 17.04$ MeV$, v_2 = 0.9169$ MeV$, v_3 = 35.20$ MeV$\}$, with $\zeta = 0.1297$. The configuration of the BEDCRT5 potential is given by $\{c = 20$ fm$, n = 5, \alpha = 5.392, a_0 = 6.708$ MeV$, a_1 = -23.80$ MeV$, a_2 = -101.1$ MeV$, a_3 = 130.6$ MeV$, a_4 = 93.97$ MeV$, a_5 = 50.19$ MeV$\}$ with $\zeta = 0.1399$.

4. *Fitting to platinum and plutonium data*

The V_0 term in the Saxon-Woods model (Eqn. 12) contains a term that varies according to the nucleus in question. The purpose of this is to design a completely general form of the Saxon-Woods potential, such that by entering the A and Z values, the specific potential for the nucleus in question can be calculated. To test this, the Saxon-Woods model was fit to two sets of scattering data for two different nuclei, simultaneously. The fitness function used is the average of the fitness function for each data set.

The overall results are fairly poor. The platinum data produces a ζ value of 0.01454, while the plutonium data has a ζ of 0.08881. This gives an average ζ value of 0.05168.

B. Optimising EDCRTn

Over the course of numerous tests of the EDCRTn model, a common pattern emerged. Despite the presence of the exponentially decreasing term to bias resulting potentials such that $V \to 0$ as $r \to 20$ fm, many of the resulting potentials would not be close to zero out to the extent of 20 fm. This suggested that while the exponentially decreasing term may bias the search, it is not sufficient to prevent unphysical results. Hence, further study of this potential was abandoned in favour of the BEDCRTn potential, which addresses this issue.

C. Optimising BEDCRT5

In all these searches the bounds $\alpha \in [0, 10]$, $a_0 \in [-200, 200]$ and $a_{1,2,\ldots,5} \in [-10000, 10000]$ were chosen, along with n n value of 5, giving a 7-dimensional parameter space.

1. Fitting to platinum data

A very successful fit for the BEDCRT5 potential was found to the platinum data (Fig. 2). This fit is very accurate across the entire range of energies. This fit has $\zeta = 0.001976$ which is substantially better than the Saxon-Woods fit.

2. Fitting to plutonium data

The BEDCRT5 model finds a poor fit to the plutonium data ($\zeta = 0.1006$), as seen in Fig. 3. It performs worse than the Saxon-Woods model, primarily due to the large peak around 2 MeV which is not present in the experimental results.

3. Fitting to oxygen data

Like the fit using the Saxon-Woods potential, this fit is also very poor. The result is very similar, in that the fit in no way resembles the shape of the actual data. Instead, all that has been achieved is a fit that is on the same order of magnitude as the experimental data. This produces a ζ value of 0.1399.

IV. DISCUSSION

It is important to note that the optimisations above were not exhaustive. Indeed, some potentials may produce better results given more search time.

A. Saxon-Woods

The results of the Saxon-Woods fit are very mixed. The model fits the platinum data very well, giving a ζ value of 0.00654. This implies that across this data set, the model was typically only around 8% off. However, when applied to other data sets, the results are much worse.

For plutonium, while some aspects of the shape are similar, others are significantly different, particularly the resonances. This suggests that the Saxon-Woods model is not a very good model for the plutonium nucleus. Unfortunately, these results do not explain why this is so.

The results get even worse for oxygen, where the shape does not at all match. The best the Saxon-Woods potential can do is get its predicted cross sections to be over the same order of magnitude as the experimental data. This was not a surprise, as we do not expect this model to perform well for low mass nuclei [4].

B. BEDCRT5

The BEDCRT5 potential produces mixed results. This is likely due to the greater number of degrees of freedom allowed in it's definition, vastly increasing the size of the parameter space. This means that if a search is to produce good results, it will tend to need many more fitness function evaluations, and hence much more computation time than say, the Saxon-Woods model. It is difficult to compare the Saxon-Woods model to the BEDCRT5 model for this reason. The BEDCRT5 model has the ability to approximate any Saxon-Woods potential quite well, due to its roots in the Taylor expansion. Hence, given unlimited computation time, we shouldn't expect it to perform worse than the Saxon-Woods model. The fact that it does in two of these results, suggests the searches missed potentially significantly better configurations. Such is the draw back of having a very large parameter space, and finite computation time.

The comparison of the Saxon-Woods and BEDCRT5 models on the platinum data also exposes a major limitation in this method. We note that while the cross-sections against energy data (see Fig. 2) are very similar, and both fit the experimental data well, the underlying potentials are very different. This suggests that there are potentially many different types of potential that will all produce matching cross-section data. Hence, given a set of cross-section data, there is no way to work backwards to find it's potential, as there are conceivably many different types of potentials that could have produced those results. Future work could explore using multiple sets of scattering data for a particular nuclei. For example, we could use differential cross-section data, or cross-section data taken from a wider variety of neutron energies. These added restrictions on results could narrow down what potential produces them.

We also note the limitations in this study imposed by the data available. Finding appropriate data sets from the EXFOR database is difficult, and many of those found do not have their associated uncertainties listed. If these were listed, then we would not be forced to use the mean squared fractional error, and could instead perform χ^2 analysis. This would allow us to give more appropriate weight to certain data sets, or certain points within data sets, while providing a better statistical grounding.

V. CONCLUSIONS

In this work we have shown that using total cross-section data alone does not appear to be enough to determine the potential around a nucleus. Indeed, two very different potentials were found that both appear to fit the platinum data well.

We have also shown that the Saxon-Woods model appears to be limited in its applicability, only providing good results for the platinum data in this study. We also introduced the BEDCRTn potential, which shows promise at modelling certain nuclei. This model benefits from its large flexibility, having its basis in the Taylor series.

Furthermore, we have shown that particle swarm optimisation, and to a lesser extent genetic algorithms, appear to be effective methods for solving these types of optimisation problems.

Future work could consider datasets besides total cross-sections, and perhaps find data sets that uniquely determine the potentials that generate them. Other potentials could also be considered, particularly those that vary with incoming neutron energy as the Saxon-Woods potential does. Further research could also focus on finding physical constraints on any possible potentials. I.e. do we require $\frac{dV}{dr}\big|_{r=0} = 0$? Can we find a bound on V?

With regards to computation, increasing population sizes, iterations, and the quality of the fitness function should all lead to better results.

[1] O. Schwerer, **"EXFOR Formats Description for Users (EXFOR Basics)"**, *International Atomic Energy Agency Nuclear Data Services Documentation Series of the IAEA Nuclear Data Section*, Jun. 2008, IAEA-NDS-206

[2] E. Elbeltagia, T. Hegazy & D. Grierson, **"Comparison among five evolutionary-based optimization algorithms"**, *Advanced Engineering Informatics*, 2005, Vol. 19, Pg. 45

[3] Conclusion of: E. Elbeltagia, T. Hegazy & D. Grierson, **"Comparison among five evolutionary-based optimization algorithms"**, *Advanced Engineering Informatics*, 2005, Vol. 19, Pg. 51

[4] See "Results for ligher elements...": R. D. Woods & D. S. Saxon, **"Diffuse Surface Optical Model for Nucelon-Nuclei Scattering"**, *Physical Review Letters*, Jul. 1954, Vol. 95, Issue 2, Pg. 578

[5] R. B. Wiringa, V. G. J. Stoks & R. Schiavilla, **"Accurate nucleon-nucleon potential with charge-independence breaking"**, *Physical Review C*, Jan. 1995, Vol. 51, Number 1, Pg. 38 - 51

[6] B. H. Bransden & C. J. Joachain, *Quantum Mechanics, 2nd edition*, Pearson Education Ltd., Essex, UK, 2000, Pg. 595 - 599

[7] J. J. Sakurai, *Modern Quantum Mechanics, 1st edition*, The Benjamin/Cummings Publishing Company, Inc., California, USA, 1985, Pg. 411

[8] B. Povh, K. Rith, C. Scholz, F. Zetsche, W. Rodejohann, *Particles and Nuclei: An Introduction to the Physical Concepts, 7th edition*, Springer-Verlag, Berlin, Germany, 2015, Pg. 288 - 290

[9] *Platinum data:* A. B. Smith, P. T. Guenther, J. F. Whalen, **"Fast-Neutron Scattering from Ta, Re, and Pt"**, *Argonne National Laboratory, Argonne, IL, USA*, EXFOR Entry 10631, EXFOR Subentry 10631001

[10] *Plutonium data:* S. Zhao-Min, **"Review of experimental data of Pu-239 fast neutron scattering cross-sections"**, *Beijing University, Beijing, China*, EXFOR Entry V0039, EXFOR Subentry V0039001

[11] *Oxygen data:* D. Lister, A. Sayres, **"Elastic scattering of neutrons from carbon and oxygen in the energy range 3.0 to 4.7 MeV"**, *Columbia University, New York, NY, USA*, EXFOR Entry 11334, EXFOR Subentry 11334001

Appendix A: Definition of terms with respect to optimisation methods

Solution A set of possible values that various parameters could take. I.e. the set of parameters $\{a, b, c\}$ could take the values $\{1, 2, 3\}$. This would be one possible solution.

Fitness function A function that takes a solution, and returns a single number, quantifying how 'well' the solution solves the problem.

Crossover function Given two solutions, we can apply a crossover function to produce a child solution. This child solution will share certain traits from both parent solutions. I.e. one way to crossover two solutions $\{1, 2, 3\}$ and $\{4, 5, 6\}$, might be to randomly select a parameter from each parent, possibly producing $\{1, 5, 3\}$.

Mutation function Given a solution, we can apply a mutation function to produce a new solution with similar, but slightly modified parameters to the original solution. I.e. we might take the solution $\{1, 2, 3\}$, and mutate it into $\{1.01, 2.11, 2.93\}$.

Selection function Given a population of solutions, a selection function will select a single member of this population. Typically selections functions will be weighted to more frequently select members with a more desirable fitness value.

Appendix B: General structure of a genetic algorithm

A genetic algorithm will generally follow the following, or some variation on the following general structure:

1. Begin with a randomised population of solutions
2. Using a selection function, select two solutions from the population
3. Crossover these solutions to produce a child
4. Decide whether to mutate this child, with a given probability
5. Evaluate this child's fitness, and add it to the new population
6. Repeat the above steps until a new population has been generated with the same size as the original
7. Repeat the above process, beginning with this new population

Appendix C: Outline of roulette wheel selection

The roulette wheel selection method is as follows. We first assign every member of the population a weight. A higher weight will mean that member is more likely to be selected than another member with a lower weight. We now divide the numbers $[0, 1]$ between all members, such that their share of the interval is proportional to their weight. I.e. to divide the numbers $\{2, 3, 5\}$, we would first note that their sum is 10. This means that the number 2 will get $\frac{1}{5}$ of the interval $[0, 1]$, the number 3 would get $\frac{3}{10}$, and so on. Hence we now get a set of intervals corresponding to each number:

$$\big\{[0, 0.2], [0.2, 0.5], [0.5, 1]\big\}. \tag{C1}$$

We now select a number randomly from the uniform distribution $[0, 1]$, and lookup what number this interval this falls in. In the example above, if we selected the number 0.74, we would see that this is a member of the interval $[0.5, 1]$ belonging to the member with weight 5.

Appendix D: Justification of Eqn. 9, the roulette wheel selection weight mapping

The fitness to weight mapping chosen is a function of both the individual member's fitness, and the overall population's fitnesses. The weight is given by

$$w = \exp\left(-\beta \cdot \frac{f - f_{min}}{f_{max} - f_{min}}\right), \tag{D1}$$

where w is the assigned weight, β is a chosen dimensionless constant, f is the fitness of that individual member, and f_{min} and f_{max} are the minimum and maximum fitnesses of all members of the population respectively.

This mapping has numerous beneficial qualities:

1. It is monotonically decreasing, meaning it maps lower fitnesses onto higher weights. This is required as we are trying to minimise the fitnesses.

2. If all fitness values are positive, then all weights will be positive. For all fitness functions chosen, only positive numbers will be produced. Roulette wheel selection requires positive weights.

3. This mapping is scale independent. We note that even if the fitnesses have units, these all cancel inside the exponential function. Consider if the units did not cancel. This would mean that we could multiply all the fitnesses by a constant amount, and would change the weighting between different population members, even though the ratio between their fitnesses has not changed. With this scale independence, if the ratio between two member's fitnesses is constant, then so too will their weights, regardless of any scaling that is applied. This ensures the genetic algorithm's search is scale independent.

4. The value of β gives precise control over how much weight is given to the best and worst members. In the case of the best member, i.e. that with the lowest fitness, $w = \exp(0) = 1$. On the other hand, the worst member will have a weight $w = \exp(-\beta) < 1$. All other weights will be distributed in this interval $[e^{-\beta}, 1]$.

Appendix E: General structure of a particle swarm algorithm

A particle swarm algorithm will generally follow the following, or some variation on the following general structure:

1. Begin with a randomised population of solutions. Each solution will be given
   (a) a random position $[x_0, x_1, ..., x_n]$
   (b) a random velocity $[v_0, v_1, ..., v_n]$
2. Now we find the best member of this population. The position of this solution is stored as the *global best position* $\mathbf{x_g}$.
3. We now update all velocities using

$$\mathbf{v} = \omega \cdot \mathbf{v_0} + c_1[\mathbf{x_p} - \mathbf{x}] + c_2[\mathbf{x_g} - \mathbf{x}], \tag{E1}$$

   where ω is a preset constant, $\mathbf{v_0}$ is the original velocity, c_1 and c_2 are two seperate numbers randomly selected from the range $[0, 1]$, $\mathbf{x_p}$ is the *personal best position* found by that member, and $\mathbf{x_g}$ is the *global best position*.

4. We now update the positions of all members using their respective velocities using

$$\mathbf{x} = \mathbf{x_0} + \mathbf{v} \tag{E2}$$

   where $\mathbf{x_0}$ is the current position.

5. Next, the fitness values are recalculated for all members in their new positions.

6. Finally, the *personal* and *global best positions* are updated.

7. The above process is then repeated from step three. This repeats for a preset number of iterations.

8. After the iterations are complete, the *global best position* is returned as the final result of the optimisation.