INSTITUTO SUPERIOR TÉCNICO

UNIVERSIDADE DE LISBOA

PROJETO INTEGRADOR DE 1º CICLO

Design of quasisymmetric fusion devices using novel machine learning methods

---

Author:
João CÂNDIDO

Supervisor:
Prof. Rogério JORGE

Research work performed for the Bachelor in Engineering Physics

at

Instituto de Plasmas e Fusão Nuclear
Physics Department

June 16, 2023

Abstract

Nuclear fusion could provide safe, clean, and virtually limitless energy production. This work focuses on stellarator devices as the solution for harnessing fusion energy. However, designing stellarators with precise magnetic field shaping is a complex task due to the high dimensionality of the problem. A method is proposed for integrating machine learning techniques, specifically the neural network, with a first-order near-axis expansion, to map desired characteristics of stellarators to the corresponding magnetic field parameters required to generate the device. Additionally, the first-order near-axis expansion allowed for the creation of a data set of quasisymmetric stellarators. The analysis of this set confirmed previous results that showed a clear division between well-performing quasi-axisymmetric and quasi-helical stellarators, forming continuous bands for each number of field periods. High-performing quasi-axisymmetric configurations with a high number of field periods are found, that have not been previously reported in the literature.

1 Introduction

Nuclear fusion holds excellent promise as a solution to the current energy crisis. Unlike traditional energy sources, fusion requires a relatively small amount of fuel to generate power. For instance, the energy produced per kilogram of fuel in a Deuterium-Tritium (D-T) reaction is approximately seven orders of magnitude higher than the energy released by burning gasoline.[1] Deuterium is abundant in Earth's oceans and tritium can be obtained from lithium or be a byproduct of fusion reactions. Furthermore, fusion is considered to be a safe and environmentally sound alternative to other forms of energy production because it does not contribute to greenhouse gas emissions and unlike nuclear fission, it does not pose a risk of uncontrolled chain reactions nor generate long-lived radioactive waste. Consequently, fusion could provide safe, clean, and virtually limitless energy.

During fusion, light nuclei collide forming heavier elements, due to the fact that, for light nuclei, the binding energy per nucleon increases with nuclear mass, these reactions are exothermic[2], releasing energy according to $E = mc^2$, with m the loss of mass during the reaction. For two particles to undergo fusion they must overcome the repulsive Coulomb force so that the short-range strong nuclear force can take effect. This means that the reaction's cross-section (probability per interaction) is extremely low in Earth's conditions, but high in stars like the Sun, due to their high temperature and strong gravitational field.[1] There are multiple reaction candidates for energy production, however, the Deuterium-Tritium (D-T) is the first option, because it has the fastest reaction rate and requires the lowest temperature.[3] The reaction is represented by[1]

$$ {}^{2}_{1}H + {}^{3}_{1}H \longrightarrow {}^{4}_{2}He(3.5\text{MeV}) + {}^{1}_{0}n(14.1\text{MeV}). \tag{1} $$

The triple product $nT\tau_E$ is used to quantify fusion performance in a single value. A lower bound for ignition (self-sustaining reaction), known as Lawson's criterion, can be expressed as

$$ nT\tau_E > 3 \times 10^{21}\ \text{m}^{-3} \cdot \text{keV} \cdot \text{s}, \tag{2} $$

where n is the plasma's density, T is the plasma's temperature and τ_E is the energy confinement time, which can be thought of as the timescale of energy loss from the plasma.[1] Ignition is not strictly required for commercial energy production, although achieving this condition equals more energy production which means more economical viability.[4]

The two primary approaches to surpassing ignition are inertial confinement, where a dense plasma is held for a short duration by being compressed by a laser or other pulsed power source, and magnetic confinement, where the plasma is contained using magnetic fields, produced by coils.[5] In these reactors energetic neutrons leave the plasma and collide with a layer containing lithium, producing tritium that is fed back to the reaction and generating heat, which will produce electricity on a standard steam cycle.[2] This work focuses on magnetic confinement.

To effectively confine charged particles, a circular magnetic field is not sufficient, due to the existence of charged particle drifts. The solution is to twist the field lines by introducing a magnetic field that follows a helical trajectory in both the long way around the torus (toroidal direction) and the short way around the torus (poloidal angle). In that regard, it is important to consider the rotational transform ι, defined as the average number of poloidal turns of a field line in one toroidal revolution.[6] The tokamak approach achieves this twisting of the field lines by applying a toroidal electric current, whereas in the stellarator approach, the magnetic field is shaped using the coils themselves.[1] Tokamaks are inherently axisymmetric, guaranteeing the confinement of all collisionless particle orbits. However, this axisymmetry also makes them susceptible to instabilities caused

by the finite toroidal electric current. On the other hand, stellarators sacrifice this symmetry, resulting in more complex coil configurations, that do not guarantee particle confinement. Nevertheless, the lack of induced current enables the plasma to operate in a steady state, effectively avoiding the instabilities encountered in tokamaks.[1,7] This work will focus on stellarators.

Particle confinement on a stellarator requires precise shaping of the magnetic field in a space with high-dimensionality[8] and multiple local minima,[9] therefore it is a complex and computationally expensive task, dependent on initial conditions. To overcome this the Near-Axis (NA) expansion formalism[10] is used. This allows us to perform a global study of the space of optimized stellarators, possibly revealing relevant initial conditions for posterior optimization.[8]

However new analytical and numerical techniques are still needed to quickly find globally optimized NA configurations. Machine learning offers a variety of tools with applications in different branches of physics, including nuclear fusion.[11,12], One of these tools is the neural network (NN), a computational model inspired by the brain, consisting of basic computing devices called neurons, connected to each other in a complex communication network, which can be used for regression tasks.[13] In this work neural networks were combined with the NA expansion in order to obtain a fast and efficient method to generate optimized magnetic field equilibria. This allows us to generate optimal configurations orders of magnitude faster than previous approaches.

2 Materials and methods

The physical model used to obtain an equilibrium solution is the ideal magnetohydrodynamics (MHD) model. It describes the plasma as a single fluid. It is obtained from Maxwell's equations and the MHD model, assuming static equilibrium as well as the conservation of mass, momentum density, and entropy. The static ideal MHD equations are as follows[1]

$$\boldsymbol{J} \times \boldsymbol{B} = \nabla p, \qquad \nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J}, \qquad \nabla \cdot \boldsymbol{B} = 0, \tag{3}$$

where $\boldsymbol{J}$ is the current density, $\boldsymbol{B}$ is the magnetic field and p is the pressure. These are the equations typically solved in a stellarator in order to find a magnetic field equilibrium $\boldsymbol{B}$ given a plasma pressure p.

Stellarator design seeks magnetic fields composed of a series of magnetic flux surfaces, i.e., smooth surfaces in which for every point $\boldsymbol{B} \cdot \hat{\boldsymbol{n}} = 0$, where $\hat{\boldsymbol{n}}$ is a vector normal to the surface, that are continuously nested around the magnetic axis. It is helpful to introduce a set of non-orthogonal magnetic flux coordinates denominated Boozer coordinates (ψ, θ, φ), where θ and φ are the poloidal and toroidal angles, respectively, and $2\pi\psi$ is the toroidal magnetic flux, which can be thought of as a radial coordinate. The Boozer angles are chosen in a way that $\boldsymbol{B}$ can be written as[1,14]

$$\boldsymbol{B} = \nabla\psi \times \nabla\theta + \iota \nabla\varphi \times \nabla\psi = \beta(\psi, \theta, \varphi)\nabla\psi + I\nabla\theta + G\nabla\varphi, \tag{4}$$

where ι, I and G are constant on flux surfaces, i.e. $\iota = \iota(\psi)$, $I = I(\psi)$ and $G = G(\psi)$.

Two important conditions in stellarator design are omnigenity and quasisymmetry. Omnigenity ensures that the time-averaged magnetic drift off of a magnetic surface vanishes for all particles, implying the confinement of all collisionless particle orbits. Quasisymmetry, a subclass of omnigenity, is characterized by a symmetry in the field strength.[1,15] It is defined in terms of Boozer coordinates as

$$|\boldsymbol{B}| = B(\psi, \theta, \varphi) = B(\psi, M\theta - N\varphi), \tag{5}$$

where N and M are integers.[1] Quasisymmetric stellarators can be divided into three categories: quasi-axisymmetric (QA) for $N = 0$, quasi-helical (QH) symmetric for $N \neq 0$ and $M \neq 0$, and finally quasi-poloidal (QP) symmetric for $M = 0$.[1] The latter will not be considered as QP symmetry cannot be constructed near the magnetic axis.[15]

A quasisymmetric magnetic field equilibrium can be approximately obtained from the MHD equations with the use of the NA expansion. By accounting for the conditions of quasisymmetry and stellarator symmetry, defined as invariance when rotating the stellarator upside down, $\psi(R, \phi, Z) = \psi(R, -\phi, -Z)$,[1] the following first-order expression for the magnetic field strength can be derived

$$|\boldsymbol{B}| = B(r, M\theta - N\varphi) = B_0[1 + r\bar{\eta}\cos(\theta - N\varphi) + O(\epsilon^2)], \tag{6}$$

where $r = \sqrt{2|\psi|/B_0}$ labels the flux surface and $\bar{\eta}$ can be interpreted as a measure of the variation of B.[14] Although this work focuses on the first order of the expansion, it is possible to obtain an expression for higher orders, with the downside that quasisymmetry cannot be obtained for most magnetic axis shapes.[8]

To generate quasisymmetric stellarator configurations using the NA expansion, the *pyQsc*[16] Python package was used. For the first order expansion the required inputs are $\bar{\eta}$, and the magnetic axis shape, represented in cylindrical coordinates (R, ϕ, Z) as

$$R(\phi) = r_{C0} + r_{C1}\cos(n_{fp}\phi), \qquad Z(\phi) = z_{S1}\sin(n_{fp}\phi), \tag{7}$$

where n_{fp} is a natural number referred to as the number of field periods. After the stellarator is generated various properties can be obtained. These include $L_{\nabla B} = B\sqrt{2/||\nabla \boldsymbol{B}||^2}$, where $||\quad||$ represents the Frobenius norm, $L_{\nabla B}$ provides an estimate of the minor radius for which the expansion remains accurate ($r \ll L_{\nabla B}$); the minimum radial location of the axis R_{0min}, useful for assessing if there is enough space for coils in the center of the device; the rotational transform ι; the length of the magnetic axis $\bar{L}$, which relates to excursion and therefore to the complexity of the device;[8] the maximum elongation κ_{max}, where k measures the ellipticity of a poloidal cross-section of the last flux surface;[17] and finally the helicity N,[18] used to distinguish QA from QH stellarators. On the one hand, a stellarator requires sufficient ι for confinement, on the other hand, simpler coils require low $\bar{L}$ and κ, with the added benefit that a lower elongation reduces the probability of a particle colliding with a wall.

A neural network consists of a series of layers, each comprising multiple neurons. In the input layer, each neuron represents an input feature. Followed by hidden layers, in which neurons operate by computing a weighted sum of the outputs from all connected neurons and passing the result through an activation function, producing a scalar value that is transmitted to the next layer. Finally, the output layer consists of neurons representing the network's outputs. Training a neural network corresponds to fine-tuning the weights and biases to accurately represent the desired function. This iterative process employs the backpropagation algorithm, which minimizes a loss function, in this case, the Mean Squared Error (MSE) between the training data outputs, and the predicted outputs, with the objective of generalizing well to unseen data.[13, 19]

An optimization algorithm based on backpropagation named *Adam* is used, as it offers several advantages over other algorithms. It is computationally efficient and particularly well-suited for problems characterized by large datasets and high dimensionality.[20] The Adam algorithm and other regressors are implemented using the *scikit-learn* Python library. Adam's hyper-parameters (values set before the training process) include alpha (a regularization term), batch_size (number of points evaluated between updating the weights), learning_rate_init (initial step used for updating weights), the activation function, and the hidden_layer_sizes (list of the number of neurons for each layer). In this work, the number of neurons was set to be the same across hidden layers. In order to prevent overfitting, which occurs when a model becomes too specialized to the training data performing poorly on unseen data, early stopping is employed, this involves setting aside a portion of the training set to evaluate the network's performance during training. If the evaluation on this set does not show improvement, training is halted, even if the loss function continues to decrease.[21] Regressors are evaluated with the R^2 score, which measures how well the regression model fits the data, where 1 is a perfect score.

Cineca's supercomputer MARCONI was employed to expedite the creation of the dataset and the optimization of the neural network. All the code produced for this work is open-source and available in a GitHub repository[22]

3 Results and Discussion

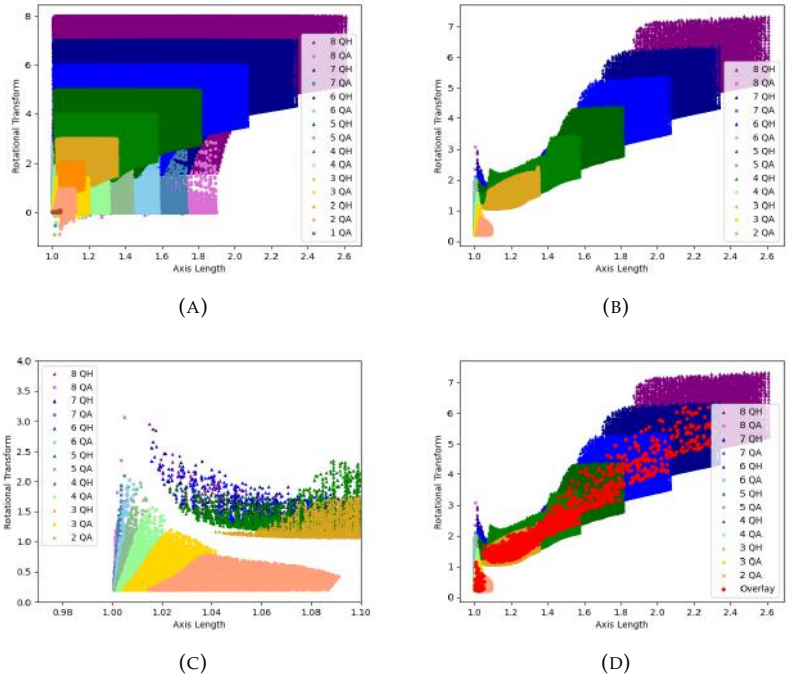
3.1 Creating and Analysing the Data Set

The space of quasisymmetric stellarators was mapped by employing both grid and random scans using *pyQSC*. The following intervals for the *pyQsc* inputs were scanned

$$0 < r_{C1} \leq 0.3, \qquad -0.3 \leq z_{S1} < 0, \qquad 1 \leq n_{fp} \leq 8, \qquad -3 \leq \bar{\eta} \leq -0.01, \tag{8}$$

where r_{C1} and z_{S1} were set as positive and negative, respectively, in order to select positive ι, avoiding redundant stellarators. About 2.2 million points were obtained. These are represented in Fig. 1a. A database cleaning was performed, after which 200'000 devices, represented in Fig. 1b, with the following characteristics, that we term high-performing, remained

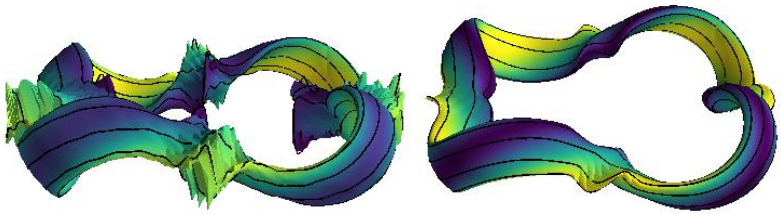
$$\iota > 0.2, \qquad \kappa_{max} < 10, \qquad L_{\nabla B} > 0.2, \qquad R_{0min} > 0.4, \tag{9}$$



(A) (B)

(C) (D)

FIGURE 1: Data set in terms of the rotational transform and axis length, with paler colors and a circle marker representing QA stellarators with different n_{fp} and with brighter colors and a triangle marker representing QH stellarators. Fig. 1a: Complete data set, Fig. 1b: Clean data set, Fig. 1c: Clean data set zoomed in; Fig. 1d: Stellarators with different n_{fp} but close ι,$\bar{L}$ and κ_{max} (relative differences bellow 1%).



(A) (B)

FIGURE 2: Examples of stellarators created with *pyQsc*. Fig. 2a: Unoptimized stellarator; Fig. 2b: Optimized stellarator

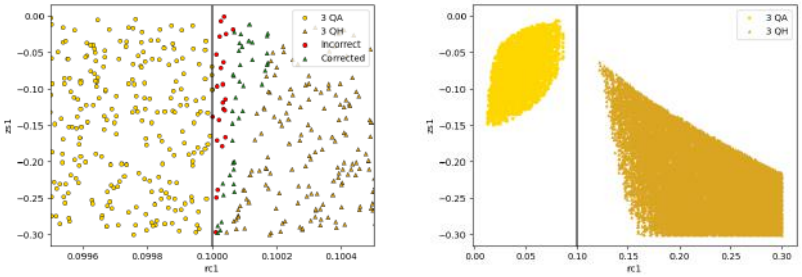
These conditions were previously used in conjunction with other conditions exclusive for second-order searches in Ref. [8].

Unlike previous second-order results[8] where several QA stellarators with $1 \leq n_{fp} \leq 3$ and QH stellarators with $n_{fp} \geq 2$ were able to pass the filters, the first-order approach presented in this work did not allow any QA stellarators with $n_{fp} = 1$ to pass the filters. However QA devices with higher n_{fp} remained. As we are aware, there are no high-performing QA stellarators with $n_{fp} > 3$ previously reported in the literature. As for the QH stellarators, none of the devices with $n_{fp} \leq 2$ was able to pass the filters. It is worth noting that previous literature implemented a second-order approach, that employed additional filters.[8] A notable example of a QA stellarator with $n_{fp} = 4$ that successfully passed the filters is provided in Table 1. This particular device exhibits high ι and low $\bar{L}$ and κ_{max}, while still accommodating space for coils. Hence, it shows potential as a candidate for future investigations. As previously observed[8, 23], while represented in terms of ι and $\bar{L}$, QH stellarators form continuous bands for each n_{fp} spanning high ranges of $\bar{L}$ (Fig. 1b), while QA devices form these bands for lower values of ι and $\bar{L}$ (Fig. 1c). It is also noticeable a clear division between QA and QH solutions (Fig. 1c).

It is numerically and analytically proven[24, 25] that for the used representation of the magnetic axis, Eq. (7), the QA-QH phase transition is independent of z_{S1} occurring at $r_{C1} = 1/(1 + n_{fp}^2)$. There are however small errors

r_{C1}	z_{S1}	n_{fp}	$\bar{\eta}$	ι	$\bar{L}$	κ_{max}	$L_{\nabla B}$	R_{0min}
0.043	-0.037	4	-0.681	1.303	1.013	5.635	0.4111	0.957

TABLE 1: Characteristics of a well-performing QA stellarator with $n_{fp} = 4$, obtained using a first-order NA expansion. Left: *pyQsc* inputs; Right: Stellarator properties.



(A) (B)

FIGURE 3: Division between QA stellarators (represented with yellow circles) and QH stellarators (represented with orange triangles) for $n_{fp} = 3$, red markers indicate devices whose helicity was not correctly predicted, and green markers represented points whose classification was corrected with increased resolution. Fig. 3a: Complete data set; Fig. 3b: Clean data set

in the order of 10^{-4}, which can be noticed in a zoomed-in representation of the complete data set (Fig. 3a), these errors tend to decrease as z_{S1} furthers from 0 and n_{fp} increases. To assure that these are computational errors, the incorrectly labeled points, represented in red and calculated with the default toroidal resolution ($N_\varphi = 61$), were compared with the points correctly labeled after a recalculation using a higher resolution ($N_\varphi = 100$), represented in green. The black line represents $r_{C1} = 1/(1 + n_{fp}^2)$. As expected the errors decreased with a higher resolution, so it is safe to assume it was a computational error due to insufficient resolution. In contrast, for dealing with the clean data represented in Fig. 3b, the default resolution is sufficient, as there is a clear gap between QA and QH devices. It is also notable that in the representation in terms of r_{C1} and z_{S1} QA and QH stellarators that passed the filters also form continuous bands.

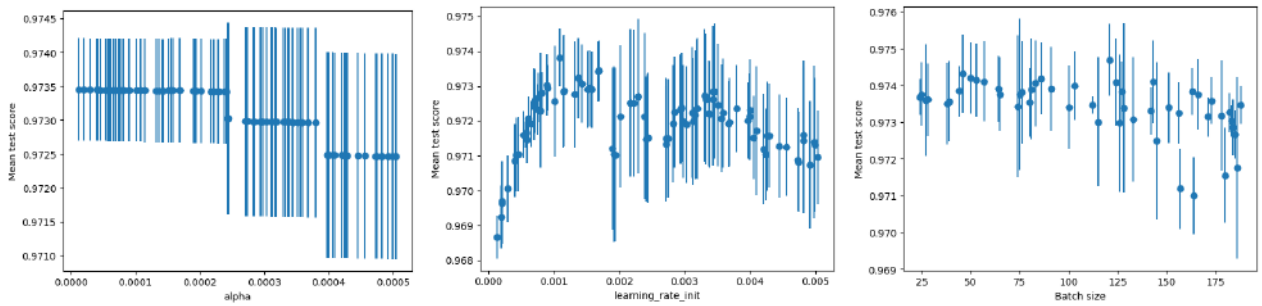
3.2 Neural Networks and Regression Models

Instead of generating stellarators with *pyQSC*, which requires searching through the inputs to find a device with the desired properties, a regression neural network is trained to map a function $(\iota, \bar{L}, \kappa_{max}) \longrightarrow (n_{fp}, r_{C1}, z_{S1}, \bar{\eta})$, allowing stellarators to be directly obtained from their magnetic field properties.

The first challenge encountered is that neural networks are not suitable for representing non-bijective functions, and as shown in Fig. 1d, there are devices with close properties, but different n_{fp}. To address the issue of multiple solutions, one approach is to train multiple neural networks, including a term in the loss function that evaluates the proximity of the predicted functions generated by each network, this means that as the loss decreases the NNs will diverge from one another allowing multiple solutions.[26] However, this complexity is not necessary for this specific problem. A simpler solution is to train a NN for each n_{fp}, which ensures better regression by simplifying the predicted function and by removing a feature. This approach also offers the added benefit of being able to select the desired n_{fp} when obtaining a stellarator. In this work, $n_{fp} = 3$ is chosen, but the following procedure is easily replicable for other n_{fp}.

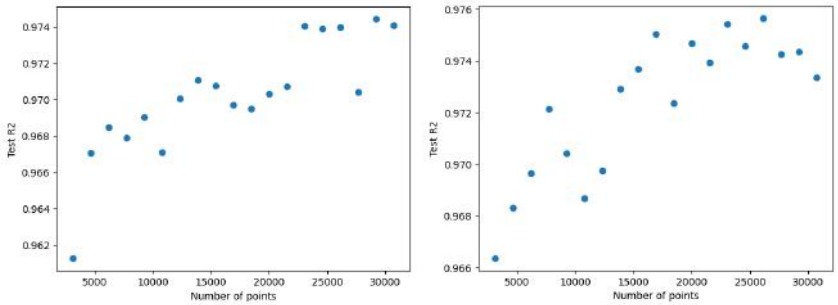
A neural network was created using the Adam algorithm with the following hyper-parameters (HP), from now on referred to as default: hidden_layer_sizes = (35, 35, 35), activation = 'tanh', solver = 'adam', alpha = 0.0001, batch_size = 'auto', learning_rate_init = 0.001, max_iter =1 000, shuffle = True, to = 0.0001, verbose = args.verbose, warm_start = False, early_stopping = True, validation_fraction = 0.1, n_iter_no_change = 10. To ensure reproducibility, all random seeds were set to 0.

The clean data set was chosen to train the NNs, despite being less comprehensive than the complete data set. This decision was made to enable a more accurate regression of the higher-performing stellarators. Since the primary objective of this neural network is to find good initial conditions for optimization this trade-off is reasonable. This unoptimized NN achieved a R^2 score of 0.9741 on the test set.



(A) (B) (C)

FIGURE 4: Mean test score from cross-validation using R^2 score, the error bars represent the standard deviation between folds. (Fig. 4a): alpha; (Fig. 4b): learning_rate_init; (Fig. 4c): batch_size.



(A) (B)

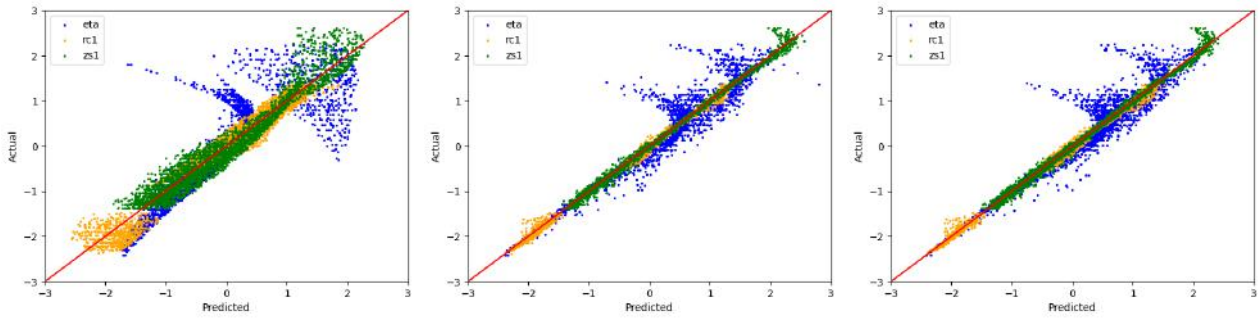
FIGURE 5: R^2 test score in terms of the number of points of the training set (the same test set was used for every evaluation). (Fig. 5a) default NN; (Fig. 5b) optimized NN.

The most important hyperparameters, namely activation, batch_size, alpha, and learning_rate_init were evaluated through a randomized search using k-fold cross-validation with R^2 score. In this approach, the training set was divided into k=5 subsets. The NN was then trained and evaluated five times, with each subset used as a validation set while the remaining subsets served as the training data.[27] Each HP was evaluated individually while keeping the rest set to default. The advantage of random sampling of HP is that, unlike grid search, the number of calculations doesn't scale exponentially with the number of HPs. Therefore, random sampling can be several times more efficient than grid search, when dealing with more than 2 or 3 HPs.[28]

Among the activation functions tested, "tanh" exhibited the highest performance with a score of 0.9734 ± 0.0008, followed by "relu" with a score of 0.9724 ± 0.0022. On the other hand, the "logistic" and "identity" activation functions performed relatively worse, with scores below 0.9700. Consequently, "tanh" was chosen as the activation function. As for the hidden_layer_sizes, no single configuration stood out. Several configurations with depths of 3 or 4 and containing 30 and 45 neurons per layer presented scores above 0.9720, so these configurations were selected for further search. From Fig. 4a it can be observed that configurations with alpha bellow 2.5×10^{-4} performed better. The analysis of the Fig. 4b reveals that the score has a peak for learning_rate_init equal to 0.001. Additionally, Fig. 4c shows that neural networks with batch_size above 150 tended to perform worse.

Based on the preliminary results, a further search was conducted, scanning the selected hidden_layer_sizes and batch_size as well as normal distributions for learning_rate_init and alpha, with (mean, standard deviation) equal to $(1 \times 10^{-3}, 2 \times 10^{-4})$ and $(1 \times 10^{-4}, 5 \times 10^{-5})$, respectively. Out of the 40,000 configurations scanned, the highest-performing setup, as shown in Fig. 6c, was found to have the following HP: alpha = 7.91e-05, batch_size = 87, hidden_layer_sizes = [45, 45, 45, 45], learning_rate_init = 9.3×10^{-4}. This configuration achieved a cross-validation score of 0.9750 ± 0.0007. To ensure a reliable evaluation, it is important to assess the model's performance on an unbiased test set. This set is unbiased because it was not used for training or HP selection. A R^2 score of 0.9734 was achieved, as expected this score is lower than the cross-validation score.

After applying the filters and selecting a value for n_{fp} the number of data points used to train the NN was significantly reduced to about 34'000. Within this set 10% of the points were allocated for the test set, leaving approximately 30,500 points in the training set. To assess whether this amount of data is sufficient, the test R^2



(A) (B) (C)

FIGURE 6: Predicted-actual plot, with standardized data, for different regression models. (Fig. 6a): Linear Regression; (Fig. 6b): Polynomial Regression; (Fig. 6c): NN with optimized hyper-parameters.

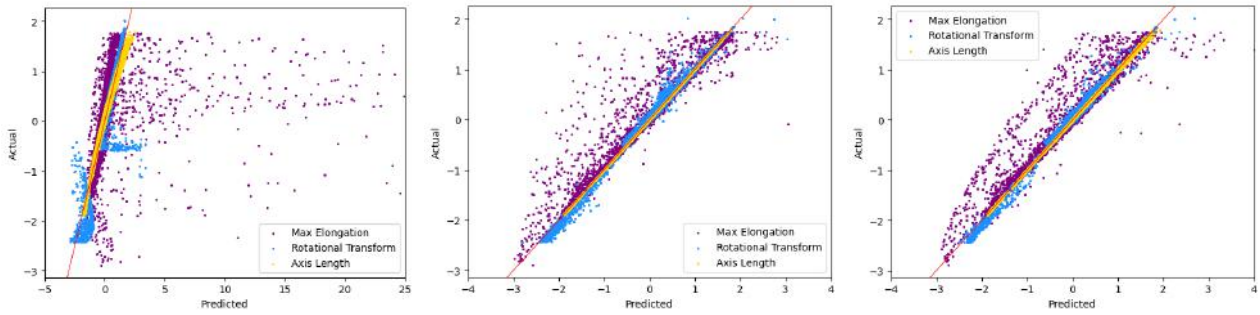
scores were plotted for NNs trained on data sets of varying sizes (Fig. 5). From the graphs, it can be observed that both functions appear to converge for data set sizes that exceed 22,500 points. However, it is worth noting that there are instances of NNs trained on similar-sized data sets that differ in R^2 score by approximately 0.002. This indicates that the testing set may not be adequate for accurately evaluating subtle differences in performance between models. To evaluate the error of the test scores, the standard deviation of the test score was calculated based on 100 NNs trained and tested on different sets. The obtained results were 0.0016 and 0.0017, for the optimized and default networks, respectively. These values can be used to assess whether the R^2 test score is capable of reliably evaluating small differences in performance among models.

It is beneficial to compare the performance of the neural network with simpler models, as such linear and polynomial regression models were fitted. On the polynomial model, the degree HP was set to 12, in an analogous process to the NN HP optimization. On the test data set, the linear model (Fig. 6a) achieved a R^2 score of 0.8724, the polynomial regression (Fig. 6b) yielded a higher score of 0.9748.

So far, the errors have been measured by comparing the predicted r_{C1}, z_{S1} and $\bar{\eta}$ with the *pyQsc* input values used to obtain ι, $\bar{L}$ and κ_{max}, which will be referred as model error. However, it is valuable to compare the pretended ι, $\bar{L}$ and κ_{max}, with the actual values produced by a stellarator with the predicted r_{C1}, z_{S1} and $\bar{\eta}$. This will be referred to as the real error. To ensure a fair comparison between ι, $\bar{L}$, and κ_{max}, all values were scaled using the same scaler. While examining Fig. 7c it is noticeable that κ_{max} is the primary contributor to the error. Its relative contribution was calculated with $(\text{MSE}_{\kappa_{max}}/3\text{MSE}).100\% = 94.39\%$. In comparison ι had a contribution of 5.34% while $\bar{L}$ had a contribution of 0.2%. This implies that small errors on the predicted magnetic axis and $\bar{\eta}$ can have a significant impact on the maximum elongation. In this analysis, for every model, there are outliers, these configurations presented κ_{max} ranging from 15 to 1400. All the regressors predicted less than 4 of these points with $\kappa_{max} < 150$, except for the linear regressor which presented 267 points with $\kappa_{max} > 15$. Even in the models with few outliers, these caused a notable increase in the MSE for κ_{max} which was reflected in the R^2 score, in order to obtain a realistic measure of the error, configurations with $\kappa_{max} > 15$ were ignored in the calculation of the MSE (without scaling) and R^2 score, these values are represented in parenthesis in Table 2.

Finally all the conditions are met for a fair comparison of the models, by examining Table 2. Considering the R^2_{test} score, for the model error, it is noticeable that all the models except the linear regressor performed similarly well, achieving scores over 0.97, the linear model also performed reasonably, accounting for the simplicity of the regressor. However, analyzing the real error reveals a much poorer performance by all the models. This can be attributed to error propagation during the generation of configurations with *pyQsc*. In extreme cases, these errors can propagate into significant discrepancies in κ_{max}, which significantly impacts the scores.

By excluding points with $\kappa_{max} > 15$ when evaluating the real error, we obtain a more reasonable assessment. Analyzing the R^2_{test} score under these conditions reveals that both the NN and the polynomial still perform similarly but with lower scores compared with the model error. As it can be seen in Fig. 7a the linear regression was particularly affected by errors on κ_{max}. Even with the exclusion of the points, its score was reduced to 0.734. However, this approach might not be suitable for studying the linear regressor, since 7.81% of its predictions surpass $\kappa_{max} > 15$. While analyzing the model error alone might indicate that a polynomial regression is sufficient for this application, a closer examination of the MSE for each variable, reveals that NNs predicted ι and κ_{max} more accurately, while the polynomial regressor achieved precise results for $\bar{L}$, performing worse for the other variables. Therefore the NN outperforms the polynomial regressor, due to a more balanced distribution



(A) (B) (C)

FIGURE 7: Predicted-actual plots, comparing the pretended ι, $\bar{L}$ and κ_{max} with the values obtained from the predicted magnetic axis and $\bar{\eta}$, with standardized data, meaning that the NN inputs and the resulting ι, $\bar{L}$ and κ_{max} obtained from *pyQsc* were normalized with the same scaler. (Fig. 7a): Linear Regression; (Fig. 7b): Polynomial Regression; (Fig. 7c): NN with optimized hyper-parameters.

	Model error	**Real error**			
Model	R^2_{test}	R^2_{test}	MSE_ι	$\text{MSE}_{\bar{L}}$	$\text{MSE}_{\kappa_{max}}$
Linear	0.8724 ± 0.0039	$-349 \pm 144 (0.734 \pm 0.013)$	0.0609	2.30E-04	3027 (2.00)
Polinomial	0.9748 ± 0.0015	0.2 ± 1836 (0.9520 ± 0.0036)	0.00224	7.85E-08	7.14 (0.389)
Default NN	0.9741 ± 0.0017	0.9 ± 1.2 (0.9578 ± 0.0081)	0.00173	1.34E-06	0.619 (0.343)
Optimised NN	0.9734 ± 0.0016	0.95 ± 0.21 (0.9579 ± 0.0083)	0.00164	3.30E-06	0.376 (0.343)

TABLE 2: Measures for the different models. The uncertainties presented correspond to the standard deviation of the scores obtained from 100 regressors trained and tested in different data sets. **Model error**: refers to comparing the predicted r_{C1}, z_{S1} and $\bar{\eta}$ with the actual value for a determined ι, $\bar{L}$ and κ_{max}; **Real error**: refers to comparing the asked ι, $\bar{L}$ and κ_{max} with the values obtained from *pyQsc* with the predicted magnetic axis and $\bar{\eta}$, the values of MSE are not standardized and can be used as an error for the model, values in parenthesis were calculated ignoring points with $\kappa_{max} > 15$.

of the errors across variables.

Comparing the optimized and the default NN reveals that the optimization did not yield significant improvements in performance. This supports the claim that the hyper-parameters for the Adam algorithm require little tuning.[20] However, further optimization with different hyper-parameter combinations could possibly provide better results.

4 Conclusions and Next steps

Firstly, this work consisted of an analysis of the space of quasisymmetric stellarators, using a first-order near-axis expansion. This analysis confirmed previous second-order results in the organization of QA and QH well-performing stellarators in continuous bands for different n_{fp} while also uncovering new high-performing configurations of QA devices with high n_{fp}. Secondly, various regressors were trained to establish a mapping between the desired characteristics of a stellarator and the corresponding elements required to generate the device. After training and evaluation, it was determined that the neural network performed slightly better than the polynomial regressor, this can be attributed to a more balanced distribution of errors across the variables.

The high performance of the regressors, coupled with successful analysis of the stellarator space means that the objectives of this work have been met. However further optimization of the neural network and expansion of the data set could yield even better results.

The logical next steps involve training models for the remaining n_{fp} and combining them, potentially in a neural network ensemble, with the integration of a user interface for easy stellarator optimization. Additionally, it would be interesting to generalize the presented method for higher-order expansion and ultimately for optimization without the near-axis expansion. Finally, incorporating methods such as online learning or reinforcement learning could enable the neural network to improve continuously with usage.

Overall, this work highlights the potential of machine learning in enhancing the design and optimization of fusion devices. These advancements could accelerate the progress toward achieving safe, clean, and sustainable nuclear fusion as a viable solution to the current energy crisis.

References

[1] L.-M. Imbert-Gérard et al. "An Introduction to Stellarators: From magnetic fields to symmetries and optimization". In: *arXiv:1908.05360* (2019).

[2] R. D. Gill. *Plasma Physics and Nuclear Fusion Research*. 1st ed. Academic Press, Wiltshire, 1981.

[3] G. McCracken et al. *Fusion: The energy of the universe*. Elsevier Academic Press, San Diego, 2005.

[4] S. E. Wurzel et al. "Progress toward fusion energy breakeven and gain as measured against the Lawson criterion". In: *Physics of Plasmas* 29 (6 2022), p. 062103.

[5] F. F. Chen. *Introduction to Plasma Physics and Controlled Fusion*. 3rd ed. Springer, Los Angeles, 2016.

[6] P. Helander et al. "Plasma Physics and Controlled Fusion Stellarator and tokamak plasmas: a comparison". In: *Plasma Phys. Control. Fusion* 54 (2012), p. 124009.

[7] Y. Xu. "A general comparison between tokamak and stellarator plasmas". In: *Matter and Radiation at Extremes* 1 (4 2016), pp. 192–200.

[8] M. Landreman. "Mapping the space of quasisymmetric stellarators using optimized near-axis expansion". In: *Journal of Plasma Physics* 88 (6 2022), p. 905880616.

[9] A. Bader et al. "Stellarator equilibria with reactor relevant energetic particle losses". In: *Journal of Plasma Physics* 85 (5 2019), p. 905850508.

[10] D. A. Garren et al. "Existence of quasihelically symmetric stellarators". In: *Physics of Fluids B: Plasma Physics* 3 (10 1991), pp. 2822–2834.

[11] M. Szűcs et al. "Detecting Plasma Detachment in the Wendelstein 7-X Stellarator Using Machine Learning". In: *Applied Sciences* 12 (1 2022), p. 269.

[12] J. Kates-Harbeck et al. "Predicting disruptive instabilities in controlled fusion plasmas through deep learning". In: *Nature* 568 (7753 2019), pp. 526–531.

[13] S. Shalev-Shwartz et al. *Understanding machine learning: From theory to algorithms*. Vol. 9781107057135. Cambridge University Press, New York, 2013.

[14] M. Landreman et al. "Magnetic well and Mercier stability of stellarators near the magnetic axis". In: *Journal of Plasma Physics* 86 (5 2020), p. 905860510.

[15] G. G. Plunk et al. "Direct construction of optimized stellarator shapes. Part 3. Omnigenity near the magnetic axis". In: *Journal of Plasma Physics* 85 (6 2019), p. 905850602.

[16] *pyQs*. `https://github.com/landreman/pyQSC`. 2013.

[17] T. C. Luce. "An analytic functional form for characterization and generation of axisymmetric plasma boundaries". In: *Plasma Physics and Controlled Fusion* 55 (9 2013), p. 095009.

[18] W. Sengupta et al. "Vacuum magnetic fields with exact quasisymmetry near a flux surface. Part 1: Solutions near an axisymmetric surface". In: *Journal of Plasma Physics* 87 (2 2021), p. 905870205.

[19] M.-C. Popescu et al. "Multilayer perceptron and neural networks". In: *WSEAS Transactions on Circuits and Systems* 8 (7 2009), pp. 579–588.

[20] D. P. Kingma et al. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980* (2014).

[21] L. Prechelt. "Early Stopping - But When?" In: *Neural Networks: Tricks of the trade* 1524 (1998), pp. 55–69.

[22] *PICNeuralNetworkQuasisymmetricStellarator*. https://github.com/JoaoAGCandido/PICNeuralNetworkQuasisymmetricStellarator. 2023.

[23] E. Rodriguez et al. "Constructing the space of quasisymmetric stellarators". In: *arXiv:2204.10234* (2023).

[24] E. Rodríguez et al. "Phases and phase-transitions in quasisymmetric configuration space". In: *Plasma Physics and Controlled Fusion* 64 (10 2022), p. 105006.

[25] C. Oberti et al. "On torus knots and unknots". In: *Journal of Knot Theory and its Ramifications* 25 (6 2016), p. 1650036.

[26] M. Di Giovanni et al. "Finding multiple solutions of odes with neural networks". In: *Combining Artificial Intelligence and Machine Learning with Physical Sciences 2020*. Vol. 2587. CEUR-WS, 2020, pp. 1–7.

[27] D. Berrar. "Cross-validation". In: *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics* 1-3 (2018), pp. 542–545.

[28] Y. Bengio. "Practical recommendations for gradient-based training of deep architectures". In: *Neural Networks: Tricks of the Trade: Second Edition* 7700 (2012), pp. 437–478.