# Learning Fractal Nodal Structures with Continuous Normalising Flows

Casano Kirlew
Department of Chemistry
Imperial College London

October 7, 2025

## 1 Project Description

### 1.1 Background

In trying to model Fermi liquids, the infamous minus-sign problem has made conventional path integral methods essentially useless. However, the constrained path formalism introduced by Ceperley [1] has previously been used to describe the quantum critical states found in heavy Fermion metals. In particular, it can be used to describe what happens to the liquid's properties when it reaches a quantum critical state (i.e., at 0 Kelvin).
The concept of 'backflow', introduced by Feynman and Cohen [2], uses a quasi-particle transform to include particle-particle interaction. This was generalised as a method to improve convergence (i.e., reduce the sign error) of quantum Monte Carlo calculations in the electron gas. Applied to a simple two-dimensional Fermion problem [3], this backflow wavefunction can be seen to develop fractal structure in the Fermion nodal surface as a function of backflow strength, as seen in figure1.

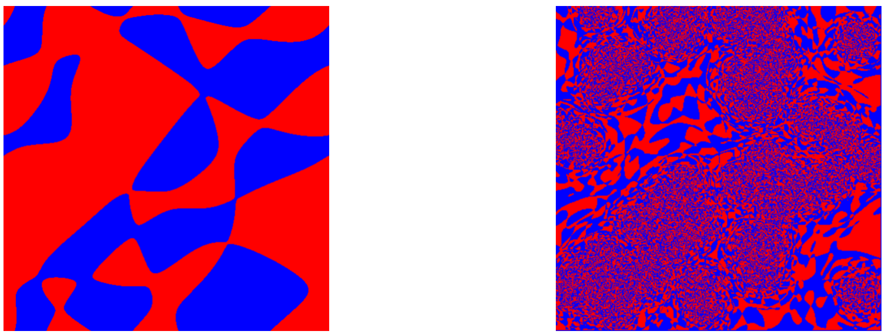


Figure 1: Two dimensional cuts through the nodal hypersurfaces of fermionic backflow wavefunctions for N=49 particles, with and without backflow on the left and right, respectively. On the right nodal pockets start to develop and the surface qualitatively changes its geometry and seems to turn into a fractal object.

### 1.2 Aims

The project was approached in the following way:

1. Following the approach of Krüger & Zaanen [3], electron wavefunctions exhibiting fractal character were generated. These wavefunctions were then modelled as hydrodynamic backflow wavefunctions. The nodal surface was subsequently modelled by varying backflow strength $\alpha$.

2. Machine learning was used to map the backflow parameters to the resulting nodal surface properties. A neural network based on an ordinary differential equation (ODE) was trained to generate the corresponding nodal structure from a given set of initial parameters.

## 1.3 Neural ODE's and Continuous Normalising Flows(CNF)

Normalising Flows are a class of generative models whose goal is to learn a complex data distribution, $p(x)$, by finding an invertible function, $f$, that transforms it into a simple, known distribution, $p(z)$ (in our case, it was Gaussian)

The relationship between the probability of a data point $x$ and its transformed version $z = f(x)$ is given by:

$$\log p(x) = \log p(z) + \log \left| \det \left( \frac{\partial f}{\partial x} \right) \right| \tag{1}$$

The term $\log \left| \det \left( \frac{\partial f}{\partial x} \right) \right|$ is the logarithm of the Jacobian determinant, which measures how much the function $f$ stretches or shrinks space.

### 1.3.1 Continuous Normalising Flows (CNFs)

Continuous Normalising Flows (CNFs) achieves this by defining the transformation as a continuous "flow". This flow is described by a Neural Ordinary Differential Equation (ODE):

$$\frac{dy}{dt} = v(y, t) \tag{2}$$

Here, a neural network learns the vector field $v(y, t)$, which defines the path that transforms a simple noise vector at time $t_0$ into a complex data sample at time $t_1$. This approach has two major advantages:

- **Memory efficient:** CNF's are more memory efficient than discrete layers. i.e the adjoint method used for calculating gradients has an $O(1)$ computational cost. [4]

- **Good at modelling Arbitrary Topologies:** CNF's have been shown to be good at learning complex data distributions, i.e spirals and rings, something which discrete models have problems with.[5]

## 1.4 Model Architecture

A CNF model based on the FFJORD architecture was used for this project; the nn was constructed using a series of residual blocks (ResBlocks) containing GroupNorm and Conv layers, designed to process the 2D spatial information of the images. ConvTranspose layers were used to upsample the features back to the original image dimensions.

### 1.5 Training Paradigms

During the project, I planned to explore 2 approaches in learning the nodal surface

- **Conditional Model:**Treat alpha as a conditional input. Then train over a fixed time interval (tspan), and for each image, the corresponding alpha value was concatenated to the input vector. So the model learns a set of images for each alpha.
- **Time-Evolving Model (alpha as Time)** A more theoretically motivated approach was then implemented where alpha was treated as the fictitious time variable itself. The model was no longer conditional but was trained to learn a single, continuous evolutionary path. To make the time-dependency explicit, the time variable t was fed directly into the neural network at each step.

I judged the quality of the models produced qualitatively (just looking at the images they produced) and quantitatively (Looking at the power spectra of the images and then radially average the power spectra into 1d plots and prints a correlation score)

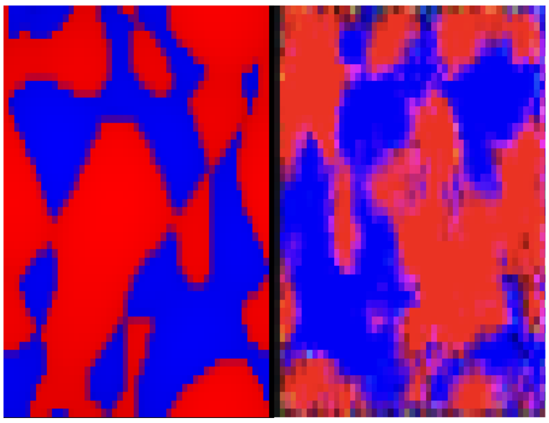
## 2 The Project

The first part of the project was spent adapting code written in the Frost group to produce multiple nodal surfaces[1]. For just training the model and to save memory, the images to be used in training initially were 50 x 50 low-resolution images of the surface. The main code I wrote during the project was adapted from a Julia CNF package found on GitHub[6]. The first main issue of the project was to do with scalar indexing on the Gpu, it forces code to run on cpu, which, for us, was much slower. (we will spend the next 2 weeks finding solutions to this issue.) The code takes too long to run and takes up too much memory. We also tried a different method, instead of training the Neural network(nn) on the rgb data from the images. I then worked on training the nn on the lines of the nodal surface images, put the line data into csv files to then be loaded to train with [2]. It was also recommended to me that if I am going to train with the lines on the nodal surface rather than the image itself them I should adopt a method similar to one Zannen and Kruger used, a sort of triangulation method to find the zeros of the wave function, that can be replicated use cubic splines. I instead adopted a simpler method, where scanned each pixel and its neighbours to find the zeros of the wave function [3].

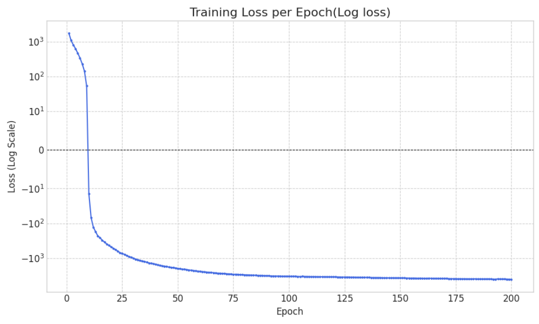
### 2.1 Conditional model

The majority of the time spent of this project was done on training the nn on a conditional model and fixing small bugs within the code. But we have shown that with a continuous normalising flows method it is possible to train a nn which starts to learn the nodal surface as shown in figure 2 [4].

[1]Github code (lines 107-128)
[2]Github code (lines 1-64)
[3]Github code (lines 18-44)
[4]Github code (lines 1-199)



(a) Ground-truth vs. Generated Nodal Surface for $\alpha = 0.0$.

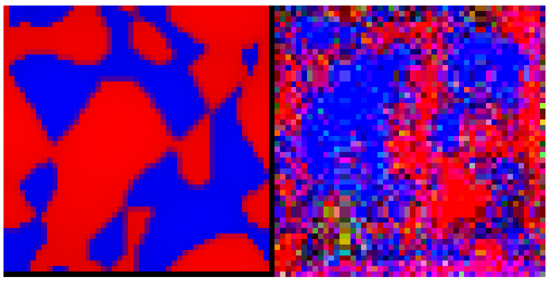


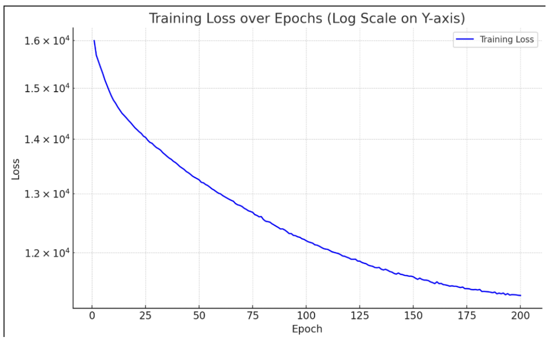
(b) Training loss curve (log scale) over 200 epochs.

Figure 2: Qualitative and quantitative results for the best CNF model that I had. (a) A visual comparison of a ground-truth nodal surface (left panel) and a model-generated sample (right panel) for a high backflow strength. (b) The corresponding training loss curve, showing some convergence.

## 2.2 Time-Evolving Model

Due to time constraints we only had time to test this once, as such I was not able to play around with the hyper parameters to see which ones would yield the best model, as such the images produced were not the best. In the future I would like to train the model for much longer [5] .





(a) Ground-truth vs. Generated Nodal Surface for $\alpha = 0.0$.

(b) Training loss curve (log scale) over 200 epochs.

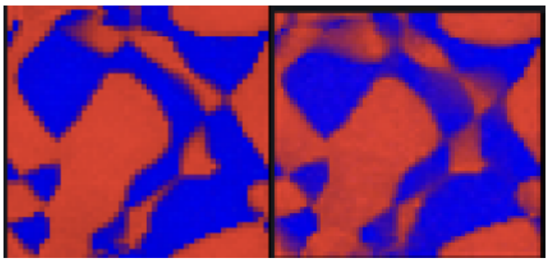
Figure 3: Qualitative and quantitative results for the trained CNF model. (a) A visual comparison of a ground-truth nodal surface (left panel) and a model-generated sample (right panel) for a high backflow strength. (b) Unlike the previous model, the loss is still trending downwards at epoch 200 suggesting more epoch's are needed in training

## 2.3 Non CNF Models

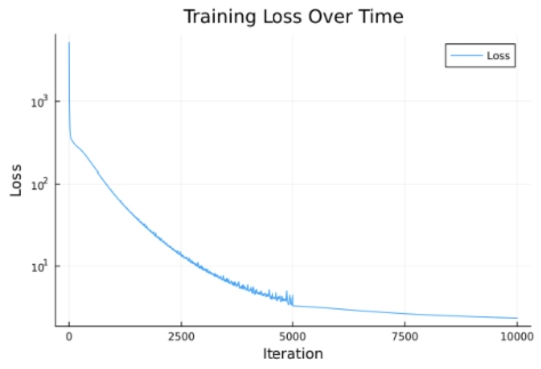
A non CNF nn was created so I could see, if the nodal surface could be learn any other way. There seems to be some promising results bellow in figure 4. The model is standard conditional neural network that learns a direct, one-way mapping, it takes a random vector $z$ and an $\alpha$ value as an input and directly outputs an image [6]. The model's

[5]Github code (lines 1-225)
[6]Github code (lines 40-67)

performance is measured using Mean Squared Error (MSE). Which is done by looking at the pixel-by-pixel difference between the model's output image and the true image [7].



(a) Ground-truth vs. Generated Nodal Surface for $\alpha = 0.0$.

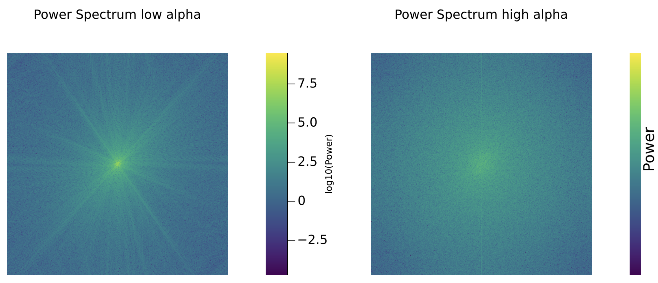


(b) Training loss curve over 10000 epochs.

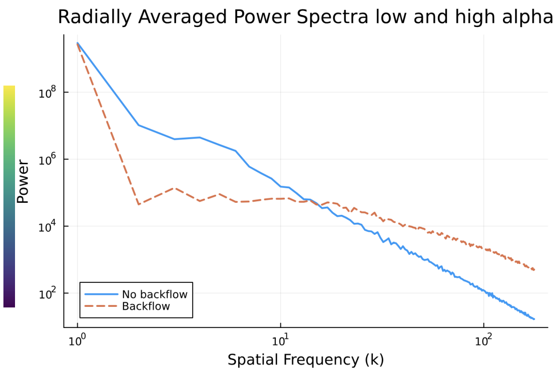
Figure 4: Qualitative and quantitative results for the trained Non-CNF model. (a) A visual comparison of a ground-truth nodal surface (left panel) and a model-generated sample (right panel) for a high backflow strength. (b) Model was able to be trained for longer due to less complex architecture

## 3 Power Spectra

To go beyond a qualitative visual inspection of the images I performed an Fast Fourier Transform (FFT) [8] of the images. This allowed us to see the the 2D power spectrum for both the ground-truth and generated images [9]. An example of which is the figure 5 bellow.



(a) Two-dimensional Fourier power spectra of low and high alpha.



(b) Radially averaged 1D power spectrum

Figure 5: For low alpha ($\alpha = 0$) power is highly concentrated in the centre with it decaying outwards. Where as for a higher alpha ($\alpha = 0.8$) the power is much more spread out, with higher spacial frequencies taking up more of the image, which shows a geometric transition into a more fractal image.

### 3.0.1 Hyper Parameter Adjustment

When training the Conditional model and the Time evolving model, I spent time adjusting the hyper parameters within my nn. Creating multiple different models as seen

[7]Github code (lines 81-99)
[8]Github code (lines 16)
[9]Github code (lines 7-23)

in figure 4. I first assumed a lower decaying learning rate may work best, ranging from $10^{-4}$ to $10^{-6}$. One of the key problems was the long training time, which was fixed by changing Interpolater and the ode solver within my Icnf function, changing it from *backwardssolvingadjoint* to *interpolatingadjoint* which reduced per time epoch from 1 hour to 6 mins. In addition for the Conditional model having an integration time span from 0-1.0 seemed to produce the best models.

## 3.1 Model Evaluation

I then calculated the 1D power spectra for my ground truth and generated images, from this I calculated a correlation score between them [10] . The performance of five separately trained conditional models and the time-evolving (alpha as time) model are compared in Figure 6 [11].

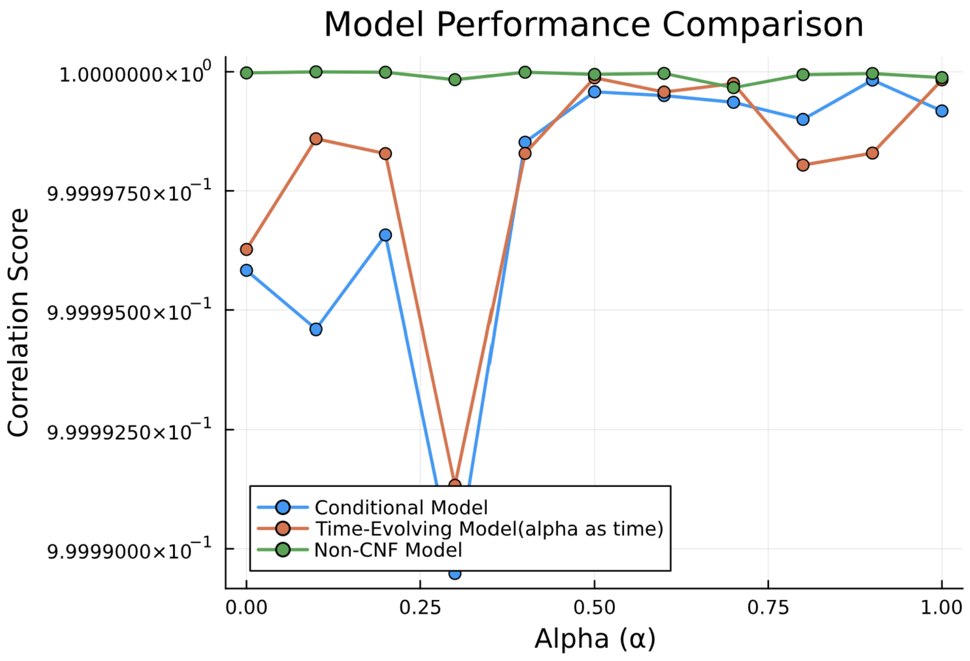


Figure 6: A quantitative comparison of model performance across a range of backflow strengths ($\alpha$). Performance is measured by the correlation score between the radially averaged power spectra of generated and ground-truth nodal surfaces. A score of 1.0 indicates a perfect structural match.

It seems for low alpha all models perform quite well except for a slight dip around $\alpha = 0.25$, it seems all were successful at learning the simple, smooth geometry of the non-interacting system, with the non cnf model having a correlation score consistently near 1 for all alpha. However, as alpha increases and the target nodal surfaces become more fractal, a slight performance gap emerges.

# 4 Conclusions

Overall, this project successfully demonstrated that a Continuous Normalizing Flow (CNF) could potentially serve as a generative model for the complex, fractal nodal sur-

[10]Github code (lines 62-77)
[11]Github code (lines 1-124)

faces of fermionic backflow wavefunctions. The conditional model—where the backflow strength alpha is provided as a direct input—proving to be a good model. Our more theoretically motivated "time-evolving" model so far looks to be a good option, we would need more time to experiment with this further. Interestingly however the non CNF nn has been shown to accurately reproduce the nodal surface. Suggesting that a cnf generative model may not be needed.

## References

[1] D. M. Ceperley. 'fermion nodes'. *Journal of Statistical Physics*, 63(5):1237–1267, 1991.

[2] R. P. Feynman and M. Cohen. 'energy spectrum of the excitations in liquid helium'. *Physical Review*, 102(5):1189, 1956.

[3] J. Krüger, F. & Zaanen. 'fermionic quantum criticality and the fractal nodal surface'. *Physical Review B*, 78(3):035104, 2008.

[4] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.

[5] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31:6571–6583, 2018.

[6] Hossein Pourbozorg ¡prbzrg@gmail.com¿ and contributors. Continuousnormalizingflows.jl: Implementations of infinitesimal continuous normalizing flows algorithms in julia. by GitHub, 8 2025. a Julia package.