Machine Learning Based Variational Approaches for Problems in Solid-State Physics

Bradley Magnetta

Yale University Graduate School of Arts and Sciences, bmagnett@gmail.com

Follow this and additional works at: https://elischolar.library.yale.edu/gsas_dissertations

Recommended Citation
Magnetta, Bradley, "Machine Learning Based Variational Approaches for Problems in Solid-State Physics" (2021). Yale Graduate School of Arts and Sciences Dissertations. 375.
https://elischolar.library.yale.edu/gsas_dissertations/375

This Dissertation is brought to you for free and open access by EliScholar – A Digital Platform for Scholarly Publishing at Yale. It has been accepted for inclusion in Yale Graduate School of Arts and Sciences Dissertations by an authorized administrator of EliScholar – A Digital Platform for Scholarly Publishing at Yale. For more information, please contact elischolar@yale.edu.
Abstract

Machine-learning-based Variational Approaches for Problems in Solid-State Physics

Bradley Magnetta

2021

In this work we utilize principles from optimization and machine learning to solve variational problems in solid state physics. First we extend the quantum variational method to include an additional objective function term that ensures ground state solutions will have localized character, obtaining the so called Wannier functions. Here the additional term relies on dictionary learning to extract localized Wannier features from a dataset of known Wannier functions. Our approach displays a systematically controllable energy-localization trade-off, has an objective function that allows for the use of fast numerical solvers, and is capable of being used in highly-efficient self-consistent algorithms. Next we improve upon this approach by replacing the manual tasks required of dictionary learning with a three-variable optimization problem so that we can learn Wannier features ”on the fly”. We also demonstrate how to use the many-body quantum variational method to guide the training of deep neural networks (DNNs) that approximate the many-body ground state wave function. The DNN representation of the wave function is also used to instruct Monte Carlo (MC) basis sampling. Finally, we apply space group symmetry to both MC and DNN training which reduces the size of DNN training data while achieving excellent energy accuracy for a two-dimensional SU(2) spin chain model.
Machine-learning-based Variational Approaches for Problems in Solid-State Physics

A Dissertation
Presented to the Faculty of the Graduate School of Yale University in Candidacy for the Degree of Doctor of Philosophy

by
Bradley Magnetta

Dissertation Director: Vidvuds Ozolins

December 2021
Acknowledgments

The author acknowledges financial support from the National Science Foundation Grant No. DMR-1611507.
Contents

1 Overview .......................... 1
   1.1 Calculating Wannier functions using dictionary learning .............. 1
   1.2 Calculating Wannier functions using adaptive dictionary learning ... 5
   1.3 Deep learning for quantum many-body physics ....................... 8

2 Calculating Wannier functions using dictionary learning .............. 11
   2.1 Lattice Hamiltonian .............................................. 11
   2.2 Wave function properties ........................................ 14
   2.3 Constructing a Wannier function dictionary ........................ 17
   2.4 Optimization problem for Wannier functions ....................... 21
   2.5 Optimization algorithm ........................................... 23
   2.6 Numerical results .................................................. 27
      2.6.1 Implementation ............................................... 27
      2.6.2 Interpolating Wannier functions for different potential widths and depths ........................................ 28
      2.6.3 Extrapolating Wannier functions for different potential types ... 34
      2.6.4 Extrapolating Wannier functions for different symmetries ....... 35
      2.6.5 Entangled bands .............................................. 36

3 Calculating Wannier functions using adaptive dictionary learning .... 37
3.1 Alternating algorithm ........................................... 37
  3.1.1 Online dictionary learning ................................. 38
  3.1.2 Nesterov’s method on the Stiefel manifold .............. 45
  3.1.3 Practical considerations ................................. 46
3.2 Material simulations .......................................... 47
3.3 Silicon ......................................................... 48
  3.3.1 Convergence and Stability ................................. 52
3.4 Tellurium materials ............................................ 55

4 Deep learning for quantum many-body physics .............. 58
  4.1 Background .................................................. 58
  4.2 DNN training ................................................ 60
    4.2.1 Monte Carlo ........................................... 63
    4.2.2 Neural network optimization via multi-use MC ........ 64
  4.3 Space group symmetry for MC-DNN applications ............ 67
    4.3.1 Background ........................................... 68
    4.3.2 Implementing space group symmetry ................... 71
  4.4 Symmetrized MC ............................................ 78
  4.5 Symmetrized multi-use MC and DNN training ................. 79
  4.6 DNN architecture ........................................... 82
  4.7 Spin model .................................................. 84
    4.7.1 Numerical setup ....................................... 84
    4.7.2 Numerical results for different k-point and irreps ... 84
    4.7.3 Numerical results for different networks .............. 87
    4.7.4 Numerical results for different learning rates ....... 89
List of Figures

1.1 Diagram demonstrating the connections between Monte Carlo (MC) sampling and neural networks for neural network enhanced MC approaches for finding the many-body ground state. x are many-body basis states and the network acts as an inexpensive mapping from basis states to wave function values. Common approaches alternate between MC sampling (black top left arrows) and network training based on energy minimization (red bottom right arrows). .......................................................... 10

2.1 A diagram demonstrating the relations between real and reciprocal space lattices for the case of two-dimensions. .......................................................... 12

2.2 A Wannier database \( W_D \) obtained using the compressed mode algorithm of Ref. [7] for a \( C_{4v} \) supercell. Four single Gaussian centered potentials were used forming the set \( V_D = \left\{ v_i \exp \left[ -\frac{(r-R)^2}{2\Delta_i^2} \right] \right\}_{i=1}^{N_W} \) for \( v = [60, 60, 100, 100] \) and \( \Delta = [0.2, 0.4, 0.2, 0.4] \). .................................................. 18

2.3 We display some of the \( N_B \approx 200 \) dictionary functions calculated using the online dictionary learning algorithm in Ref. [10]. The dictionary functions extract general features from an input data set; in our case \( W_D \) from Figure 2.2. ................................. 19
2.4 Using standard formulas of group theory [31] we project $D$ from Figure 2.3 into square $C_{4v}$ symmetry. ........................................ 19

2.5 Using standard formulas of group theory [31] we project $D$ from Figure 2.3 into square $C_{6v}$ symmetry. ........................................ 20

2.6 Exact (black lines) and interpolated (blue dots) band structures for Wannier functions obtained from Algorithm 1. The top and bottom plots use $V(r) \in V_D$ (potentials used to create $W_D$ shown in Figure 2.2) with parameters of $(v, \Delta, \mu) = (60, 0.2, 5)$ and $(v, \Delta, \mu) = (100, 0.4, 5)$, respectively. The middle plot uses a potential with parameter values $(v, \Delta, \mu) = (80, 0.3, 10)$ that was not included in the dictionary set $W_D$. ............ 29

2.7 We inspect more carefully the Wannier functions from the middle case of Figure 2.6. The top plots demonstrate the localized character of these Wannier functions while the middle plot verifies that the localization is exponential. Here for each Wannier function (line) we plot the difference between the full norm and the norm of a sub-region with radius $R$ centered at the Wannier center. Given a log-scale y-axis, the linear character of each line demonstrates exponential localization. The bottom plot demonstrates the convergence properties of Algorithm 1. .................. 30

2.8 $\mu$-dependence of the energy-localization trade-off. Here the solid curve and left y-axis demonstrates that the average $\ell_1$-norm decreases with increasing $\mu$, which corresponds to an increase in localization. The dashed curve and right y-axis demonstrates that the total energy increases with increasing $\mu$. .................. 31
2.9 The effects of \( \mu \) on localization of Wannier functions. Each plot represents the same Wannier function calculated using a different \( \mu \) (indicated by color). The linear slopes demonstrate exponential localization for each \( \mu \). Here as \( \mu \) increases the linear slope becomes more negative suggesting that larger \( \mu \) leads to a more rapid decay of the Wannier functions and an increase in localization.

2.10 We track the evolution of a single function for \( D\alpha \) for results shown in the middle plot of Figure 6 in our main text. Our algorithm implements basis pursuit every \( n_L = 10 \) iterations and thus dynamically updates \( D\alpha \).

2.11 The Wannier functions and band structure, obtained from Algorithm 1, for a two-atom type potential (two Gaussians centered at the middle and origin lattice sites) are shown for \( \mu = 5 \). The exact and Wannier interpolated band structures are shown using black lines and blue dots respectively.

2.12 The Wannier functions and band structure, obtained from Algorithm 1, for a single Gaussian potential on a hexagonal lattice are shown. The exact and Wannier interpolated band structures are shown using black lines and blue dots respectively.

2.13 The Wannier functions and band structure, obtained from Algorithm 1, for a single Gaussian potential on a \( C_{4v} \) lattice are shown. The exact and Wannier interpolated band structures are shown using black lines and blue dots respectively. Here we chose a Gaussian potential with depth \( v = 80 \) and width \( (\Delta) \) and \( \mu \) of \((0.3, 20)\) and \((0.35, 10)\) for the top and bottom plot respectively.
3.1 Algorithm 2 progress for Si. Here we display Wannier interpolated band structures (dashed lines) for $\mu = 50$ (top plot - 5 alternating iterations), $\mu = 2.5$ (middle plot - 10 alternating iterations), and $\mu = 0.00625$ (bottom plot - 20 alternating iterations).

3.2 The final alternating iteration of the implementation of Algorithm 2 in Figure 3.1 ($\mu = 0.00625$). Symmetrical $s$-like and $p$-like Wannier functions are displayed using multiple amplitudes values.

3.3 The $p$-like orbital ($W$) from Figure 3.2 and the corresponding localized target ($D\alpha$) are plotted for a specific amplitude value. We display the dictionary functions with the three largest $\alpha$ coefficient values and the sparse coefficient vector $\alpha$.

3.4 More dictionary functions are shown for the dictionary used in Figure 3.3.

3.5 Total energy vs iteration (t) of Algorithm 2 is displayed using both a dictionary initialized from scratch ($D_{\text{Random}}$) and a warm restart corresponding to the results shown in Figures 3.2, 3.3 and 3.4 ($D_{\text{Si}}$).

3.6 Symmetrized $s$-like and $p$-like Wannier functions calculated for SnTe. The interpolated band structure (red dashed) is provided for $\mu = 0.00625$ at alternating iteration 20.

3.7 Symmetrized $s$-like, $p$-like, and $d$-like Wannier functions calculated for PbTe corresponding to the interpolated band structure (red dashed) for $\mu = 0.05$ and alternating iteration 40.
4.1 A simple illustration of orbits. Each dot represents a basis state. Applying $G_S$ to any basis state $(x_i, x_j)$ of the orbit, say $x_j$, yields the same orbit $O(x)$. $x_E$ is a reference configuration (canonical) which should be the same regardless of the starting basis state $(x_i, x_j)$. Red dots, $\hat{g}^\prime_m x_E$, represent an invertible subspace of the full orbit.

4.2 Diagram illustrating our process for performing symmetrized multi-use MC and DNN training. The outer and inner loop are notated by the black and red arrows and mirror the for loops in Algorithm 11. Here we elaborate on the specific expressions needed at each stage of our process.

4.3 A diagram describing a symmetrized complex neural network. Object dimensions and elements are notated using parenthesis and brackets respectively. As indicated by Eq. (4.44), our input is a single inverted basis function (See Section 4.3.2) and output the complex wave function $\Psi_{\text{net}}(\hat{g}^\prime_m x_E)$. Here the actual output of the network, $A$ and $\phi$, are amplitude and phase respectively. Convolutions and dense layers are defined by $C$ and $D$ respectively. The green objects (widest) represent the feature maps of the convolutional layers. The sum aggregates information along the spatial indices. Periodic boundary conditions (PBCs) are enforced by wrapping enough spatial information so that the convolutional filters can reach beyond the boundaries.

4.4 Average energy (Eq. (4.48)) shown for different $j = (k, \beta - \text{index})$ notated as $k_{\beta - \text{index}}$. Both real (a) and imaginary (b) components are possible because our network is complex (See Figure 4.3. Setup includes $N_l = 6$ convolutional layers, $F = 64$ filter channels per layer, and a learning rate of $\gamma = 0.0005$. 
4.5 Norm of the gradient (Eq. (4.49)) and size of symmetrized MC sampling (Algorithm 10) is shown in subplots (a) and (b) respectively for different $j = (k, \beta - \text{index})$ notated as $k_{\beta-\text{index}}$. Setup described in Figure 4.4. 88

4.6 Average energy (Eq. (4.48)) shown for different networks parameterized by number of convolutional layers $N_l$ and number of filter channels per layer $F$. Both real (a) and imaginary (b) components are possible because our network is complex (See Figure 4.3. Setup includes $k$-point-irrep collective index $j = \Gamma_5$, and a learning rate of $\gamma = 0.0005$. 90

4.7 Norm of the gradient (Eq. (4.49)) and size of symmetrized MC sampling (Algorithm 10) is shown in subplots (a) and (b) respectively for different networks parameterized by number of convolutional layers $N_l$ and number of filter channels per layer $F$. Setup described in Figure 4.6. 91

4.8 Average energy (Eq. (4.48)) shown for different learning rates $\gamma$. Both real (a) and imaginary (b) components are possible because our network is complex (See Figure 4.3. Setup includes $k$-point-irrep collective index $j = \Gamma_4$, number of convolutional layers $N_l = 6$, and filter channels per layer $F = 64$. 92

4.9 Norm of the gradient (Eq. (4.49)) and size of symmetrized MC sampling (Algorithm 10) is shown in subplots (a) and (b) respectively for different learning rates $\gamma$. Setup described in Figure 4.8. 93

6.1 The reciprocal Hamiltonian $\tilde{H}$ is block-diagonal with block matrix dimensions of $(N_G \times N_G)$. 102

6.2 The Eigenfunctions $\tilde{\Psi}$ are block-diagonal with block matrix dimensions of $(N_G \times N_{\text{bands}})$ due to the Hamiltonian in Figure 6.1. 105
6.3 In real space the Eigenfunctions span the entire matrix. Note however, that all block rows can be obtained from the top block row (highlighted) after multiplication with an exponential. .................................................. 106

6.4 For $W$ each matrix element can be obtained by wrap-shifting the first block column (highlighted). ................................................................. 107

6.5 For $\tilde{W}$ each matrix element can be obtained by multiplying elements from the first block column (highlighted) by an exponential of specific $k$, $G$, and $R$ index. ................................................................. 108

6.6 Schematic illustrating the method of Refs. [16, 17] on the Stiefel manifold (sphere). The pink plane represents the Stiefel tangent space. Both the blue and red lines represent different Stiefel geodesics pointing in the direction of the optimum solution. The dashed line represents the Euclidean path that can be used to extrapolate from $X^{(i+1)}$ towards $Y^{(i+1)}$. The dots mark the key variables in the Nesterov Stiefel method which all occur on the sphere. ................................................................. 110
Chapter 1

Overview

In this work we discuss machine-learning-based variational problems in solid-state physics. In each of our three chapters we develop algorithms for obtaining the ground state of quantum systems. In chapters 2 and 3 we discuss how machine learning can be applied to create an efficient approach for obtaining Wannier functions while in chapter 4 we demonstrate that neural networks can be used as a symmetrized representation of the many-body ground state. While chapter 4 does not directly pertain to Wannier functions, future work in tight-binding Hamiltonian building using Wannier functions as a basis will allow us to apply our work in chapter 4 to more complex quantum systems.

1.1 Calculating Wannier functions using dictionary learning

Using electronic structure methods to perform materials design has had continued success due to the development of algorithms based on ideas from computer science and machine learning (ML). In particular, high-throughput approaches [1, 2] have allowed researchers to amass large databases of materials properties [3] and use data-driven machine learning
approaches to make predictions based on these databases. It has been suggested that to keep pace with accelerated progress, continued implementation of more complex concepts from the data science and machine learning communities will be needed [4].

In this work we consider data driven machine learning approaches for the single-electron Schrodinger equation which can be cast as the following matrix optimization problem:

$$\min_{\Psi} \text{Tr} \, \Psi^\dagger \hat{H} \Psi \quad \text{s.t.} \quad \Psi^\dagger \Psi = I,$$

(1.1)

where $\Psi$ is a matrix of DFT wave functions arranged column-wise, and $\hat{H}$ is the Hermitian Hamiltonian operator for the system. Classical methods for obtaining the ground state, such as matrix diagonalization, are costly when the size of the Hamiltonian matrix becomes large. However, typically the ground state can be obtained efficiently using iterative methods to solve Eq. (1.1). Our proposed learning method takes advantage of the fact that solutions to the variational principle are determined only up to an arbitrary unitary transformation, called ”gauge” in the physics literature. A specific choice of gauge can result in functions that are exponentially localized in real space – the so-called Wannier functions – enabling efficient computation of numerous physical properties and serving as a basis for more accurate many-body techniques.

Well established physical and chemical intuition suggests that localized solutions reflect the nature of the atoms and their surrounding environments, and therefore they should be transferable to other situations which have similar structure [5, 6]. The key idea of this work is to learn the characteristic features of localized Wannier functions that can be used in a wide variety of materials and situations. We aim to explore if there is a benefit for defining Wannier functions as a linear combination of some localized basis (feature dictionary), which is learned from previously calculated Wannier functions on other systems. We will show that in doing so we can dynamically constrain localization at each iteration
of our algorithm, provide a physically motivated starting point for the optimization algorithm, and eliminate the non-differentiability of the target function which allows for a more efficient handling of the orthogonality constraint than is possible in the work of [7, 8]. In addition, when formulated as a self-consistent problem, our approach allows in principle for the exploitation of Wannier exponential localization to reduce the computational cost of each self-consistent loop which could be indispensable for more complex beyond-DFT methods [9].

In particular, we use dictionary learning algorithms [10, 11] to extract the general features from a small dataset of precalculated Wannier functions, forming a dictionary D. We extend the quantum variational principle to include basis pursuit using D:

$$\min_{W, \alpha} \text{Tr} \hat{H} W + \mu \|W - D \alpha\|^2_2 + \gamma \|\alpha\|_1 \quad \text{s.t.} \quad W^\dagger W = I$$

(1.2)

where W are Wannier functions, \(\alpha\) is a sparse coefficient vector, and I is the identity matrix. Here \(\|W - D \alpha\|^2_2\) entices our solution to have features of the learned dictionary D. This approach is shown to generalize to potentials and environments that were not used in constructing D.

After defining preliminaries in sections 2.1 and 2.2, we describe dictionary construction in section 2.3. Using a small database of Wannier functions (Fig. 2.2) we learn a dictionary being careful to ensure that the number of dictionary functions (\(N_d\)) and the parameter controlling our sparse objective term (\(\gamma\)) are chosen so that the dictionary functions we learn are specialized to our Wannier database and as a result transfer to similar physical environments. In section 2.4 we describe in detail the optimization problem we use to calculate Wannier functions using a previously learned dictionary and discuss a fast numerical solver than can be used to solve this optimization problem in section 2.5.

Our results demonstrate that for two-dimensional potentials our dictionary learning ap-
proach successfully transfers information learned from our Wannier database to new but similar physical environments. The middle plot of Figure 2.6 shows that the Wannier functions we calculate interpolate the exact band structure with great accuracy and Figure 2.7 demonstrates that the Wannier functions we find are exponentially localized. Interestingly, the parameter $\mu$ in our optimization problem controls a trade-off between energy accuracy and localization (see Figure 2.8). We also show that the dictionary we learn can be used to calculate Wannier functions for different potentials and lattices with different crystal symmetries than were included in the training database. In addition, we show that our approach is capable of treating Wannier functions for entangled bands (see Figure 2.13).

High-throughput calculations are automated approaches that can be used to efficiently obtain material properties for many compounds. Further development of efficient and autonomous algorithms may help predict even more materials for solving global problems in energy, space, computing, housing, and more. Of relevance to this field are Wannier functions, which are found by solving for the ground state of the quantum variational principle (Eq. (1.1)) for a gauge that ensures $\Psi$ is exponentially localized in real space. Not only do Wannier functions give insight into the localized nature of electrons within materials, but Wannier functions have also been used to explain physical phenomena such as polarization [12]. In addition, there are also many practical benefits for calculating Wannier functions. For instance, highly-efficient material properties calculations can be performed when using Wannier functions to build tight-binding models [13] due to their exponentially localized character. All of these applications rely on the exponential localization of Wannier functions for establishing a natural cut-off in real space which reduces memory and the number of operations for numerical calculations.

The importance of Wannier functions has led researchers to develop efficient and automated methods for obtaining them. The methods of maximally localized Wannier functions (MLWFs) and selected column of density matrix (SCDM) have matured into a com-
bined high-throughput approach for reducing human involvement [14]. However, one of the main limitations of this hybrid approach is its inability to enforce symmetrical Wannier character. An alternative variational approach utilizes sparse optimization through an additional term to the objective of Eq. (1.1) to find a sparse localized basis [7, 8]. This method has been extended to enforce Wannier symmetry through an additional constraint [15] which allows for more efficient subsequent calculations by taking advantage of the irreducible wedge of the Brillouin zone.

In chapter 2 we introduce a method that utilizes dictionary learning to obtain Wannier features from a database of previously calculated Wannier functions and then variationally obtains new Wannier functions using learned Wannier features. To do this we optimize Eq. (1.2) with respect to the Wannier variable $W$ and sparse coefficient vector $\alpha$. Optimization of this form has a main advantage over [7, 8] because the objective function is convex and differentiable (with respect to $W$) which allows for the use of fast numerical algorithms such as Nesterov’s accelerated gradient method on the Stiefel manifold (NSM) [16, 17] and basis pursuit [18]. Here NSM ensures orthonormality is preserved by stepping along the Stiefel manifold during optimization (See Figure 6.6).

### 1.2 Calculating Wannier functions using adaptive dictionary learning

One of the main limitations of a dictionary based approach is that it relies on a large and diverse database of Wannier functions to learn the necessary Wannier features needed to be considered a general technique. This problem could be addressed by creating an open source database of Wannier functions similar to how the Materials Project [3] catalogs band structures and other properties for a wide range of materials.

We can understand the need to have such a large dataset in a more systematic way
by analyzing Eq. (1.2). Optimization over $\alpha$ will select a finite number of dictionary functions of $D$ to ensure the term $\mu \|W - D\alpha\|^2_F$ is smallest. However, when solving Eq. (1.2) for a specific physical system, significant error due to $\mu \|W - D\alpha\|^2_F$ will occur if $D$ does not contain features relevant to the physical system regardless of our choice for $\alpha$. In some cases decreasing $\mu$ will not solve this problem either. Instead we must gather new training data that contains the missing features and retrain $D$. Because WFs across different potentials and crystal symmetries have similar features, it is not unrealistic to think that this problem could be addressed through an open source database of WFs. While plausible, realistically this strategy would take considerable time and resources to implement due to the tools that would need to be built (websites, databases) in order to allow researchers to collaborate. Instead, it would be advantageous to develop a technique that avoids needing to gather training data while still learning a Wannier features using dictionary learning.

In chapter 3 we suggest a self-contained method for obtaining any Wannier function and systematically improving the features of $D$ without the need for a prior database of Wannier functions. Our method adapts $D$ as it is exposed to more physical systems forcing it to learn new Wannier features while retaining previously learned features. To do this we use Eq. (1.2) but instead optimize over $W$, $D$, and $\alpha$ by alternating between two steps. This can be understood by looking at the $W$ specific problem of Eq. (1.2);

$$\min_{W} \text{Tr} \ W^\dagger \hat{H} W + \mu \|W - L\|^2_F \quad \text{s.t.} \quad W^\dagger W = I, \quad (1.3)$$

where $L = D\alpha$ is the localized target. Optimizing over $W$ reduces the total energy while being influenced by $L$ and constrained to be orthonormal. In the case that $L$ is a poor choice, we would like to reduce the error in the objective function due to $\|W - L\|^2_F$ by changing $L$. Our method does so by using
to adapt $D$ and adjust $\alpha$, where $D_m$ represents a single dictionary function. Here $W$ is fixed and obtained from Eq. (1.3), which helps to guide our adaption of $D$ in Eq. (1.4) to a more physical form. Because our goal is to reduce the error due to $D\alpha$ it makes sense to optimize over both variables simultaneously within the same alternating step. Coincidentally, Eq. (1.4) is the dictionary learning problem [11] which has a well documented stable online algorithm suitable for our application [10].

The full optimization problem for our adapted Dictionary approach is

$$
\min_{W, D, \alpha} \text{Tr} W^\dagger \tilde{H} W + \mu \| W - D\alpha \|_F^2 + \gamma \| \alpha \|_1 \quad \text{s.t.} \quad W^\dagger W = I, \| D_m \|_2 \leq 1,
$$

(1.5)

where the last term is a symmetry constraint based on the work of [15]. Constraining symmetry ensures the topology of the band structure will be correct and helps to reduce computations by restricting the Fourier components of the Wannier function to the irreducible wedge of the Brillouin zone. After discussing the details of our approach we provide density functional theory (DFT) results for Si, SnTe, and PbTe to demonstrate the efficacy and practicality of our Wannier method.

Our approach has the following main benefits: the ability to simultaneously build a Wannier database while obtaining a general Wannier dictionary, compatibility with symmetry, calculation of Wannier functions directly from the Hamiltonian, a significant reduction of manual dictionary work, and a strategy for controlling the few remaining algorithm-
mic parameters.

1.3 Deep learning for quantum many-body physics

Quantum many-body systems have an immense number of particles and interactions making it difficult to obtain useful numerical results for large systems. Even in cases where only local interactions are significant, resulting in a sparse Hamiltonian, exact diagonalization is intractable due to the geometric increase in the relevant Hilbert space dimensions.

Variational Monte Carlo (VMC) is a numerical method that relies on a wave function and samples the highest probability particle configurations to reduce complexity [19, 20]. While VMC is effective at approximating the many-body ground-state, it is difficult to determine a flexible and accurate Ansatz while still remaining feasible.

There has been much motivation by the deep learning community to avoid these issues and reduce the complexity of the many-body wave function by creating inexpensive mappings between particle configurations and wave function values. Carleo and Troyer describe this problem from a deep learning perspective within the context of dimensional reduction and feature extraction [21]. In addition, they provide a framework for using artificial neural networks (ANNs) to perform an inexpensive mapping, and utilize an alternating scheme between Monte Carlo (MC) sampling and ANN optimization to obtain an approximate ground state as illustrated by Figure 1.1. Here $x$ represents a many-body basis state, equivalent to Slater-determinants for fermionic systems, $S$ represents a chain of sampled basis states, $\Psi$ represents the neural network representation of the wave function which takes $x$ as input and outputs a wave functions value, $W$ represents the trainable variables corresponding to the network tensor representation, $E$ is the average energy which acts as our network loss function, and backpropagation (chain rule) is used to determine the gradients for $W$. MC Sampling (black arrows) relies on network inference to deter-
mine the probability for each basis state which is used in the acceptance criterion. Network training (red arrows) relies on a MC chain (network input) to obtain wave function values that are needed to define the average energy (network loss). From here backpropagation is used to determine the gradients of the loss with respect to the network weights (trainable tensor variables) with which we use to perform a single stochastic gradient descent (SGD) step to update the network with new weights that minimize the average energy. By alternating between MC sampling and network training our network will eventually become a good approximation of the ground state.

Extensions of [21] and the type of approach outlined in Figure 1.1 have been developed to reuse MC samplings for multiple ANN updates [22], include the sampling within a DNN structure [23], and apply symmetry to reduce the chain complexity in Abelian [24] and non-Abelian SU(n) and O(n) groups [25].

We build upon this work by implementing space group symmetry to study lattice based systems and provide 2D examples for the SU(2) spin model. Our numerical results demonstrate that space group symmetry can be used to reduce the size of DNN training data while achieving excellent energy accuracy. In addition, we introduce some subtleties for reusing MC chains that differ from the importance sampling gradient method used in [22].
Figure 1.1: Diagram demonstrating the connections between Monte Carlo (MC) sampling and neural networks for neural network enhanced MC approaches for finding the many-body ground state. $x$ are many-body basis states and the network acts as an inexpensive mapping from basis states to wave function values. Common approaches alternate between MC sampling (black top left arrows) and network training based on energy minimization (red bottom right arrows).
Chapter 2

Calculating Wannier functions using dictionary learning

2.1 Lattice Hamiltonian

Electronic wave functions in crystals are represented either as functions in real space, \( \psi(r) \), or by their expansion coefficients in terms of a chosen basis such as plane waves. They are found as normalized eigenfunctions of the effective crystal Hamiltonian,

\[
\hat{H}(r) = -\frac{1}{2} \nabla^2 + V(r),
\]

(2.1)

where \( V(r) \) is an effective potential that includes Coulomb interactions with the ions and other electrons, as well as so-called exchange and correlation effects arising from quantum mechanical interactions between the electrons [26, 27]. In this work we study two-dimensional systems for simplicity but our notation is applicable to three-dimensions as well. The ions are arranged periodically on the sites of a lattice, which is usually assumed to be perfect and without a boundary. The periodicity of the lattice imparts a corresponding periodicity onto \( V(r) \). For instance, in two dimensions the lattice is defined by basis vectors \( \mathbf{l}_1 \) and \( \mathbf{l}_2 \), so that any lattice point is given by \( \mathbf{R} = n_1 \mathbf{l}_1 + n_2 \mathbf{l}_2 \), with \( n_1, n_2 \in \mathbb{Z} \)
Figure 2.1: A diagram demonstrating the relations between real and reciprocal space lattices for the case of two-dimensions.

(lattice points are shown as large red dots in Fig. 2.1). The potential obeys

\[ V(r + R) = V(r) \]

for any lattice vector \( R \). It is easy to see that the Fourier coefficients of the potential, \( V(G) \), are nonzero only for vectors \( G \) that satisfy \( e^{iG \cdot R} = 1 \). The vectors \( G \) form a lattice in reciprocal space with basis defined by \( l_i \cdot g_j = 2\pi \delta_{ij} \), and \( G = m_1 g_1 + m_2 g_2 \), \( m_1, m_2 \in \mathbb{Z} \) (large blue dots in Fig. 2.1).

In practice the infinite crystal is approximated by a supercell which consists of a sufficiently large number of the basic periodic unit cells. Periodic boundary conditions are used to wrap the boundaries of the supercell. Figure 2.1 shows the relation between the supercell in real space and the wave vectors in reciprocal space. Any point in the supercell can be expressed as \( r = R + \rho \), where \( \rho \) belongs to the primitive cell (small blue points in the bottom left square of Fig. 2.1) and \( N_r \) corresponds to the total number of discrete
real space lattice positions. Correspondingly, in reciprocal space the wave vectors are expressed as \( q = k + G \), where vectors \( k \) belong to the primitive cell of the reciprocal lattice, the so-called Brillouin zone, and \( N_q \) corresponds to the total number of discrete reciprocal space lattice positions. For the simple setup shown in Fig. 2.1 the latter are expressed as \( k = \frac{i_1}{N_k} g_1 + \frac{i_2}{N_k} g_2 \), where \( i_1, i_2 = 0, 1, \ldots, N_k - 1 \) and \( N_k = N_R \).

We choose to depart from the conventional \( \psi(r) \) representation of wave functions and introduce a matrix notation that better conforms to standard numerical practices in the field of machine learning. Our choice is also motivated by the fact that electronic structure calculations are almost always implemented with matrix methods. In both real and reciprocal space we represent wave functions \( \Psi \) and Wannier functions \( W \) as matrices where row indices run over a regular grid of \( r \) and \( q \) values,

\[
\Psi, W \in \mathbb{R}^{N_r \times N_\beta N_{\text{bands}}}
\]

and

\[
\tilde{\Psi}, \tilde{W} \in \mathbb{R}^{N_q \times N_\beta N_{\text{bands}}},
\]

respectively. The columns of \( \Psi \) and \( W \) cover the low-energy eigenspace of the discretized Hamiltonian matrix in Eq. (6.1) located in our Appendix. They are labeled by a set of two indices \( (\beta, n) \), where \( n = 1, \ldots, N_{\text{bands}} \) labels energy bands and \( \beta \) runs either over wave vectors \( k \) in the Brillouin zone or over real-space lattice vectors \( R \) within the supercell. In the former case, the columns represent Bloch states, which are eigenvectors of the Hamiltonian with a wave vector \( k \), while in the latter case they represent Wannier functions associated with the lattice site \( R \).

For simplicity, we assume in our discussion that the dimensions of \( \Psi \) and \( \tilde{\Psi} \) are identical due to the super-cell grid constraints \( N_\beta = N_R = N_k \) and \( N_r = N_q \) as shown in Figure 2.1, while in practical calculations we employ a double-dense real space grid to
avoid wrap-around errors when calculating $V(r)\psi(r)$ via Fast Fourier transform (FFT).

We use the notation $\Psi_{\beta n}$ to refer to the columns of $\Psi$ and adopt the following ordering of rows and columns in $\Psi$ and $\bar{\Psi}$. The rows of $\Psi$ ($\bar{\Psi}$) are grouped in blocks of size $N_\rho$ ($N_G$) corresponding to a unit cell at lattice vector $R$ (Brillouin zone wave vector $k$) and denoted by $\Psi_R$ ($\bar{\Psi}_k$). With a slight abuse of notation, an individual element at row $r$ and column $\beta n$ is referred to as $\Psi_{\beta n}(r)$ (analogously for $\bar{\Psi}_{\beta n}(q)$). Similarly, the columns of both $\Psi$ and $\bar{\Psi}$ are arranged first by $n$ and then by $\beta$ so that a group of $N_{\text{bands}}$ consecutive columns corresponds to a single $\beta$.

The Fourier mapping between real and reciprocal space representations allows us to write

$$\Psi = U_F^\dagger \bar{\Psi},$$

where $U_F \in \mathbb{R}^{N_r \times N_r}$ is a unitary Fourier matrix, $U_F^\dagger = U_F^{-1}$, described in our Appendix with Eq. (6.2). As we transform between real and reciprocal space, the column indices $\beta$ and $n$ do not change meaning.

### 2.2 Wave function properties

The ground state wave functions are found by minimizing the total energy subject to the orthonormality constraint:

$$\Psi = \arg \min_X \text{Tr} X^\dagger H X \quad \text{s.t.} \quad X^\dagger X = I.$$  \hspace{1cm} (2.3)

Using the block-diagonal property of $\bar{H}$, derived in our Appendix with Eq. (6.1), this minimization problem can be split into smaller sub-problems for each $k$ separately,

$$\bar{\Psi}_{[k]} = \arg \min_X \text{Tr} \bar{X}^\dagger \bar{H}_{[k]} \bar{X} \quad \text{s.t.} \quad \bar{X}^\dagger \bar{X} = I,$$  \hspace{1cm} (2.4)
where $|k|$ is a compact notation for the diagonal block corresponding to $k$ in a block diagonal matrix. Note that each $\tilde{\Psi}_{[k]}$ is an $N_G \times N_{\text{bands}}$ sized matrix.

Since the objective function and the constraint in Equations (2.3) and (2.4) are invariant with respect to unitary transformations $\Psi \rightarrow U^\dagger \Psi$ with $U^{-1} = U^\dagger$, there are many possible representations of the wave functions that give the same ground state total energy and charge density. Two choices are widely used. The first and the most straightforward is to associate the index $\beta$ with the wave vector $k$ in the corresponding diagonal block of the eigenvalue problem. In this case, the matrix $\Psi$ also assumes a block-diagonal form where each $k$ block corresponds to Bloch functions $\Psi_{kn}$ with crystal momentum $k$ and band index $n$, satisfying the usual eigenvalue equation

$$\tilde{H}_{[k]} \tilde{\Psi}_{kn} = \epsilon_{kn} \tilde{\Psi}_{kn}, \quad (2.5)$$

as shown by Figure 6.2 of Appendix. We denote these $N_G \times N_{\text{bands}}$ blocks for each $k$ by $\tilde{\Psi}_{[k]}$.

While the periodic Bloch functions extend over the entire crystal, another useful choice consists of wave functions that are spatially localized. These are the so-called Wannier functions, which are related to the Bloch functions by a unitary transformation over the column indices:

$$W = U_L^\dagger \Psi, \quad (2.6)$$

where it is easy to verify that $U_L$, described in our Appendix with Eq. (6.5), is a unitary matrix of dimension $N_R N_{\text{bands}}$. The main advantage of Wannier functions over the Bloch functions is that an appropriate choice of $U_L$ leads to exponentially localized functions in real space that are easier to analyze and enable fast algorithms exploiting sparsity (i.e., short-range). In addition, the localized nature of Wannier functions allow for correlations to be drawn between Wannier character and chemical bonds in a material [5]. This con-
nection to material properties along with their tendency to resemble atomic orbitals [28] and the arbitrariness of Eq. (2.6) suggests that Wannier functions should be transferable to other atomic environments with similar structure. The success of our method is based on this idea of transferability.

The orthogonality condition $W^\dagger W = I$ can be interpreted as a requirement that the Wannier functions are orthogonal to their own translations by a lattice vector, as well as to those with a different band index. This property is often referred to as shift-orthogonality. It is straightforward but somewhat laborious to show that shift-orthogonality in real space is equivalent to the following relation for $k$ components of the Fourier transform:

$$\sum_G \tilde{W}_{Rn}^*(k + G)\tilde{W}_{Rm}(k + G) = \frac{\delta_{nm}}{N_k}, \quad (2.7)$$

the proof of which can be found in Ref. [8]. In other words, shift-orthogonality of $W$ can be ensured by simply orthogonalizing and properly normalizing its Fourier transform at each $k$ in the first Brillouin zone.

In practice, calculating Wannier functions requires finding an appropriate unitary transformation matrix $U_L$ in Eq. (2.6). This is often done explicitly by first obtaining the Bloch eigenfunctions via diagonalization of the block-diagonal Hamiltonian, Appendix Eq. (6.1), on a regular grid of $k$ vectors, followed by minimization of some suitably chosen localization function, such as real-space spread [28]. While being conceptually straightforward, this approach however suffers from the fact that the objective function and the unitary constraint on $U_L$ are both non-convex, which leads to the existence of local minima and yields results that may vary with the initial guess. Recently, an alternative method was proposed which forms a localized Wannier basis from selected columns of the density matrix and avoids the need to solve a non-convex optimization problem [29].
2.3 Constructing a Wannier function dictionary

Dictionary learning within the context of sparse coding [11, 30] is a successful method for constructing a general localized basis. In this work we follow the terminology of the machine learning community and refer to the learned basis obtained from sparse coding as a dictionary. We distill a set of dictionary functions $D$ from a database of known Wannier functions $W_D$ via the following minimization problem:

$$
\min_{D, \alpha} \|W_D - D\alpha\|_F^2 + \gamma \|\alpha\|_1 \quad \text{s.t.} \quad \|D_m\|_2 \leq 1,
$$

(2.8)

where the lattice site index $R$ has been dropped. Here $W_D \in \mathbb{R}^{N_R N_p \times N_W}$, $D \in \mathbb{R}^{N_R N_p \times N_D}$, and $\alpha \in \mathbb{R}^{N_D \times N_W}$ for $N_W$ Wannier functions and $N_D$ dictionary functions in real space.

We use the online method for determining $D$ described in Ref. [10], which alternates between minimizing $D$ and $\alpha$ while cycling through all Wannier functions sequentially. $D$ is updated so that the difference between $W_D$ and the sparse reconstructions $D\alpha$ is minimized. The sub-problem for finding the expansion coefficients $\alpha$ (basis pursuit) uses the $\ell_1$-norm to obtain the closest sparse reconstruction $D\alpha$.

It is important to choose a combination of $\gamma$ and $N_D$ that produces desired dictionary function character as demonstrated by Ref. [11]. Having a sparse $\alpha$ and over-complete $D$, when $N_D$ is greater than the effective dimensionality (localized support region) of $W_D$, results in dictionary functions that are highly specialized to the training dataset. If $\gamma$ is chosen so that $\alpha$ is not sparse and $N_D$ is much smaller than the effective dimensionality then the dictionary functions will not generalize and important Wannier features will not be learned. In this work we choose $N_D \approx 200$ (for an effective dimensionality of approximately 100).

Figure 2.3 demonstrates the base Wannier dictionary $D$ we will use in this work learned
Figure 2.2: A Wannier database $W_D$ obtained using the compressed mode algorithm of Ref. [7] for a $C_{4v}$ supercell. Four single Gaussian centered potentials were used forming the set $V_D = \left\{ v_i \exp \left[ -\frac{(r-R_i)^2}{2\Delta_i} \right] \right\}_{i=1}^{NW}$ for $v = [60, 60, 100, 100]$ and $\Delta = [0.2, 0.4, 0.2, 0.4]$. 
Figure 2.3: We display some of the $N_B \approx 200$ dictionary functions calculated using the online dictionary learning algorithm in Ref. [10]. The dictionary functions extract general features from an input data set; in our case $W_D$ from Figure 2.2.

Figure 2.4: Using standard formulas of group theory [31] we project $D$ from Figure 2.3 into square $C_{4v}$ symmetry.
Figure 2.5: Using standard formulas of group theory [31] we project D from Figure 2.3 into square $C_{6v}$ symmetry.

from the Wannier database $W_D$ shown in Figure 2.2; the latter was obtained using the compressed mode algorithm of Ref. [7] over a small set of potentials defined in the caption. To ensure compatibility with both high-symmetry and perturbed off-symmetry systems we align our dataset so that the Wannier centers coincide with an anchor position of our dictionary function grid (typically the middle position). Doing so also ensures the greatest generalization of Wannier features to the dictionary.

Our results will show that the dictionary functions have general Wannier character making them a capable basis for potentials outside the scope of $W_D$. Another important property of the dictionary is the ability to transfer functions to any desired lattice symmetry. We do this by projecting each function to the closest irreducible representation of site symmetry group using standard formulas of group theory [31]. Figures 2.4 and 2.5 demonstrate the projection of D into square $C_{4v}$ and hexagonal $C_{6v}$ lattice symmetries, respectively. While the radial character is unaffected by symmetrization, the angular character is improved by making the functions conform to the local environment. This, in addition to the flexibility of basis pursuit, allows us to produce quality results for poten-
tials that are different from those used to generate the input data $W_D$. We isolate a specific dictionary function symmetry transformations in Figures 2.3 and 2.4 using red and blue outlines. Note that the blue outlined transformation results in an additional partner dictionary function which is required to preserve symmetry.

### 2.4 Optimization problem for Wannier functions

We now have everything we need to state the optimization problem that finds Wannier functions using dictionary learning by including the total energy and basis pursuit related terms:

$$\{W, \alpha_W\} = \arg \min_{\{X, \alpha\}} \text{Tr} X^\dagger HX + \mu \|X - D\alpha\|_F^2 + \gamma \|\alpha\|_1$$

s.t. $X^\dagger X = I$, \hspace{1cm} (2.9)

where $\alpha_W$ is the resulting approximate expansion of the Wannier function in terms of the learned dictionary $D$. This is a modification of the energy minimization (Eq. (2.3)) by adding terms that favor spatially localized solutions and similarity to dictionary. Localization is promoted via the second term which minimizes the distance between $W$ and the localized function space covered by the dictionary $D$, while the third term ensures that only a finite number of important terms are kept in $\alpha_W$.

Our proposed approach differs from the method of compressed modes [7] which is a variational approach that solves for compactly supported Wannier functions via $\ell_1$ regularization using

$$\min_W \text{Tr} W^\dagger HW + \mu \|W\|_1 \hspace{1cm} \text{s.t.} \hspace{0.5cm} W^\dagger W = I.$$ \hspace{1cm} (2.10)

The $\ell_1$-norm introduces a non-differentiable term into the objective function, while Eq. (2.9) promotes localization by incorporating basis pursuit to a learned dictionary of Wannier
basis. As a result the objective function with respect to the $W$ variable is differentiable, promising greater stability and faster convergence when used with fast numerical solvers, such as Nesterov’s accelerated gradient method on the Stiefel manifold described in the next section. Furthermore, this is a more data-driven approach to electronic structure that in principle allows re-use and learning from previous calculations in line with the outlook of [4], while compressed modes and other direct minimization approaches are experience-agnostic.

For the case of multiple Wannier centers, we ensure that the dictionary functions are aligned with each Wannier center to ensure compatibility with both high-symmetry and perturbed off-symmetry systems. The positions of the Wannier centers may be chosen based on physical intuition (e.g., chosen as atomic positions) or determined from symmetry analysis of the Bloch states using the induced representation theory methods [32].

Minimization over the variables $X$ and $\alpha$ is performed by alternating between

$$X^{(i+1)} = \arg\min_X \operatorname{Tr} X^\dagger H X + \mu \|X - D\alpha^{(i)}\|^2_F \quad \text{s.t. } X^\dagger X = I, \quad (2.11)$$

and

$$\alpha^{(i+1)} = \arg\min_\alpha \|X^{(i+1)} - D\alpha\|^2_F + \gamma' \|\alpha\|_1, \quad (2.12)$$

where $\gamma' = \gamma/\mu$. Minimization in Eq. (2.12) is a standard basis pursuit problem, which for each iteration dynamically establishes a best-match localized function using a dictionary, while the constrained quadratic problem in Eq. (2.11) updates $X$ to minimize energy while collared to remain close to its dictionary representation. The accuracy of the energy minimization by the resulting Wannier functions depends on the value of the parameters $\mu$ and $\gamma$, and on the quality of the dictionary $D$. In the case of a ”perfect” dictionary, i.e., one that can reproduce every physically reasonable Wannier function with a finite number
of terms, the energy becomes exact and the dependence on the value of $\mu$ disappears.

We can use the $k$ separability of the Hamiltonian and the $k$-space shift-orthogonality constraint, Eq. (2.7), to reformulate the minimization in Eq. (2.11) as a set of $N_k$ independent problems:

$$
\tilde{X}^{(i+1)}_{[k]} = \arg \min_{\tilde{X}} \text{Tr} \tilde{X}^\dagger \tilde{H}_{[k]} \tilde{X} + \mu \| \tilde{X} - \tilde{L}^{(i)}_{[k]} \|_F^2 \quad \text{s.t.} \quad \tilde{X}^\dagger \tilde{X} = I / N_k, \quad (2.13)
$$

where $\tilde{L}^{(i)}_{[k]}$ is the $k$-th block of the Fourier transform of $L^{(i)} = D \alpha^{(i)}$. $\tilde{X}_{[k]}$ refers to the unique column block of the full matrix. When we alternate between solving Equations (2.13) and (2.12), we use FFT to transform between reciprocal ($\tilde{X}$) and real ($X$) space representation.

### 2.5 Optimization algorithm

General theory of orthogonality constrained optimization problems is outlined in Ref. [33]. When applied to solving Eq. (2.13), the relevant geometric space of “tall skinny” matrices with orthonormal columns (orthogonal $k$-frames) forms a Stiefel manifold. Several gradient based minimization algorithms have been developed, some of which are analyzed in Ref. [33]. In this work, we use Nesterov’s accelerated gradient method on the Stiefel manifold (NSM), which was developed in Refs. [16, 17] and demonstrated to outperform existing state-of-the-art quasi-Newton methods on some large, ill-conditioned problems.

Nesterov’s method for a convex function $f(x)$ in Euclidean space has the following two-step iterative form:

$$
x^{(i+1)} = y^{(i)} - s \nabla f(y^{(i)}), \\
y^{(i+1)} = x^{(i+1)} + \frac{i - 1}{i + 2} (x^{(i+1)} - x^{(i)}), \quad (2.14)
$$
which can be viewed [34] as a discretization of a second-order differential equation with a
time-dependent damping term: \( \ddot{X} + \frac{3}{7} \dot{X} + \nabla f(X) = 0 \), where \( \dot{X} \) and \( \ddot{X} \) refer to velocity
and acceleration, respectively. Setting \( x^{(0)} = y^{(0)} \) initially, the iterations step away from
\( y^{(i+1)} \) along the gradient with step size \( s \), and then adds a momentum term with \( \frac{i-1}{i+2} \) acting
as mass and \( x^{(i+1)} - x^{(i)} \) being net velocity.

Adapting the Euclidean form of Nesterov’s method to the Stiefel manifold requires two
key modifications: (1) generalization of the gradient step to conserve orthogonality and (2)
smooth extrapolation along a geodesic line through two points on a Stiefel manifold. Both
of these modifications are described in more detail in our Appendix 6.3. For now we will
compactly notate the two steps needed to complete NSM as \( \text{GradStep} \) and \( \text{Extrapolate} \), as
is done in Ref. [16, 17].

Theoretical results and numerical tests of the convergence rate of the described NSM
algorithm [16, 17] show that the iteration count to achieve a predetermined accuracy varies
as the square root of the condition number of the Hamiltonian matrix. The condition
number is a reflection of the sensitivity of a function to changes to its input. Our problem
is naturally ill-conditioned, where high energy \( G \) can have a large impact on solution
error. To avoid this we use the Teter-Allan-Payne preconditioner [35] to reduce the impact
of high energy \( G \) on the gradient described by Eq. 4.22.

Our Algorithm 1 is based on Algorithms 1 and 2 of Ref. [16] with a few alterations
incorporating basis pursuit. Every time we perform basis pursuit we obtain a new set of
localized target functions which in turn alters our objective function given by Eq. (6.13) in
our Appendix.

Because Wannier functions are exponentially localized they can be "cut-off" or set
to zero outside some localized sub-region of the full lattice without significantly distur-
bning the physics of the studied system. As a result, calculations involving cut-off Wannier
functions will be faster due to their sparse character. Inspired by this, we implement basis
pursuit in Algorithm 1 within some localized sub-region of the full lattice. In our results we typically choose this region to span two primitive cells to ensure we capture all essentially Wannier information. Other problems that produce more or less localized Wannier functions will require smaller and larger localized cut-off regions respectively.
Algorithm 1: Algorithm for calculating Wannier functions via basis pursuit using the Nesterov accelerated gradient method on the Stiefel manifold.

Input: $\tilde{H}$, $D$, $\mu$, $\epsilon_{\text{tol}}$, $\epsilon_L$, $s_{\text{in}}$, $n_L$;

Initialize: set $X^{(0)} = Y^{(0)}$ as random symmetrized Gaussians, $i = j = \beta = 0$, $s = s_{\text{in}}$;

while $(\tilde{X}^{(i+1)} - \tilde{X}^{(i)}) < \epsilon_{\text{tol}}$, $i++$ do

<table>
<thead>
<tr>
<th>Basis pursuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>if $i % n_L = 0$ and $(L^{(i-1)} - L^{(i-2)}) &gt; \epsilon_L$ then</td>
</tr>
<tr>
<td>$L^{(i)} = \text{BasisPursuit}(X^{(i)})$, $j = \beta = 0$, $s = s_{\text{in}}$;</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>$L^{(i)} = L^{(i-1)}$;</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nesterov on Stiefel manifold</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $k \in \text{BZ}$ do</td>
</tr>
<tr>
<td>$X^{(i+1)}<em>{[k]} = \text{GradStep}(\tilde{Y}^{(i)}</em>{[k]}, \tilde{L}^{(i)}<em>{[k]}, \tilde{H}</em>{[k]}, s)$;</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

if $F(\tilde{X}^{(i+1)}) < F(\tilde{X}^{(i)})$ then

| $j += 1$, $\beta = \frac{j}{j+3}$; |
| for $k \in \text{BZ}$ do |
| $Y^{(i+1)}_{[k]} = \text{Extrapolate}(X^{(i+1)}_{[k]}, \tilde{X}^{(i)}_{[k]}, \beta)$; |
| end |

else

| $j = \beta = 0$, $\tilde{X}^{(i+1)} = \tilde{Y}^{(i+1)} = \tilde{X}^{(i)}$; |
| end |

end
2.6 Numerical results

We demonstrate the efficacy of our Wannier method by using the symmetrized dictionaries shown in Figures 2.4 and 2.5, which were learned from the Wannier database $W_D$ shown in Figure 2.2. $W_D$ was constructed by solving the $\ell_1$-regularized compressed mode algorithm \[7, 8\] for a set of periodic arrangements of inverted Gaussian potential wells

$$V_D = \left\{ v_i \exp \left[ -\frac{(r - \frac{a}{2})^2}{2\Delta_i} \right] \right\}_{i=1}^{NW}$$

with depths $v_i$ and widths $\Delta_i$ described in the caption of Figure 2.2.

Using these dictionaries we highlight our Wannier method for three key lattice Hamiltonian scenarios: 1) $V(r)$ with same character as $V_D$ but of different parameter values, 2) $V(r)$ with different character from $V_D$, and 3) $V(r)$ of same character and parameters as $V_D$ but for different lattice symmetries. These scenarios demonstrate the robustness of our technique for calculating Wannier functions for many potentials and lattices.

2.6.1 Implementation

In Algorithm 1, we only perform basis pursuit every $n_L$ iterations and in the case that $n_L$ is very large we allow NSM to fully converge before changing the objective function. This is in line with the exact procedure of alternating between solving Equations (2.11) and (2.12). However, in practice a finite value of $n_L$ produces very accurate Wannier functions while still being very efficient. Also, once our target localized functions have converged to the tolerance $\epsilon_L$ we stop performing basis pursuit because the change in objective function is now negligible.

Practically, while NSM has no tuning parameters we must provide two stopping tolerances ($\epsilon_{\text{tol}}$ and $\epsilon_L$) which control the desired convergence of our learned Wannier functions and target localized functions respectively. In addition, we find that a sufficient choice for the frequency of which we perform basis pursuit to be every $n_L = 10$ algorithm iterations.
Our initial guesses for $W$ are devised by choosing Gaussians with random character, symmetrizing them to the correct symmetry derived from theory of induced representations [31], and enforcing SOR via Eq. (2.7). It is important to make sure that our dictionary is centered at each lattice site with a Wannier function before performing basis pursuit.

For all numerical results shown above the physical dimensions of our lattice correspond to the lattice constant $a = 1$. This determines the length of lattice vectors $l$ and $g$. We also set an energy cutoff to $E_{\text{max}} = 300$ and exclude all $G > E_{\text{max}}$.

### 2.6.2 Interpolating Wannier functions for different potential widths and depths

We now show that a Wannier dictionary $D$ (Figure 2.4) learned from a small database $W_D$ (Figure 2.2) for a two-dimensional lattice with $C_{4v}$ symmetry contains enough general information to find Wannier functions for potential configurations outside the scope of $W_D$. Here we keep the character of $V(r)$ similar to $V_D$ but study different depths and widths. This example is analogous to attempting to use Wannier functions from carbon and oxygen to predict Wannier functions for nitrogen.

We examine the accuracy of our learned Wannier functions by comparing the Wannier interpolated band structure (blue dots) to the eigenvalue dispersion $\epsilon_{km}$ (black lines) of our Hamiltonian. Using Wannier functions we interpolate a real-space effective Hamiltonian

$$h_{nm}(R) = W^\dagger_{Rn} HW_{0m},$$

where $1 \leq n, m \leq N_{\text{bands}}$. These $C_{N_{\text{bands}}} \times N_{\text{bands}}$ matrices are then Fourier transformed by summing over the lattice vectors within the supercell,

$$h_{nm}(k) = \sum_R e^{i k R} h_{nm}(R),$$

28
Figure 2.6: Exact (black lines) and interpolated (blue dots) band structures for Wannier functions obtained from Algorithm 1. The top and bottom plots use $V(\mathbf{r}) \in V_D$ (potentials used to create $W_D$ shown in Figure 2.2) with parameters of $(v, \Delta, \mu) = (60, 0.2, 5)$ and $(v, \Delta, \mu) = (100, 0.4, 5)$, respectively. The middle plot uses a potential with parameter values $(v, \Delta, \mu) = (80, 0.3, 10)$ that was not included in the dictionary set $W_D$. 
Figure 2.7: We inspect more carefully the Wannier functions from the middle case of Figure 2.6. The top plots demonstrate the localized character of these Wannier functions while the middle plot verifies that the localization is exponential. Here for each Wannier function (line) we plot the difference between the full norm and the norm of a sub-region with radius $R$ centered at the Wannier center. Given a log-scale y-axis, the linear character of each line demonstrates exponential localization. The bottom plot demonstrates the convergence properties of Algorithm 1.
Figure 2.8: $\mu$-dependence of the energy-localization trade-off. Here the solid curve and left y-axis demonstrates that the average $\ell_1$-norm decreases with increasing $\mu$, which corresponds to an increase in localization. The dashed curve and right y-axis demonstrates that the total energy increases with increasing $\mu$.

and diagonalized to obtain an interpolated band structure along a desired k-path in the first Brillouin zone. This is done in Figure 2.6, which demonstrates that not only can we find energetically accurate Wannier functions for the potentials used in the dictionary (top and bottom plots) but also for intermediate values of $v$ and $\Delta$ (middle plot). We document the Wannier functions for these cases in Figure 2.7 along with a log-scale plot that indicates exponential localization (middle), and the convergence properties of the NSM algorithm (bottom). In order to measure exponential localization we plot difference between the full norm and the norm contained within a sub-region centered about each Wannier function center and expand this sub-region to the lattice boundaries (x-axis). The log-scale y-axis produces linear plots which represents exponential decay and thus exponential localization.

In our method $\mu$ controls the balance between satisfying the total energy and ”fit” with our dictionary. Due to the fact that our dictionary has compact support by construction, the parameter $\mu$ can also be seen to affect the trade-off between localization and energy.
accuracy. Indeed, the term $\mu \| W - D \alpha \|_F^2$ outside of the support region of $D$ reduces to the $\ell_2$ norm taken over the region outside of the support, which imposes a penalty on the integrated weight of the Wannier function at large distances from the center.

Figure 2.8 shows the trends between average $\ell_1$-norm (solid curve and left y-axis), total energy (dashed curve and right y-axis), and $\mu$. With increasing $\mu$ we see a decrease in $\ell_1$-norm and an increase in the total energy, which suggests a $\mu$-dependent energy-localization trade-off. Furthermore, we can investigate how $\mu$ effects localization by studying how $\mu$ effects our exponential localization measure, $\| W \|_2 - \| W(0 \to r) \|_2$. Figure 2.9 plots this measure for the same Wannier function, a $p$-orbital from Figure 2.7, for different $\mu$. Each demonstrates a linear slope indicating exponential localization. However, as $\mu$ increases the linear slopes become more negative which indicates that $\mu$ controls localization and the trade-off demonstrated in Figure 2.8 is between energy accuracy and localization.

At each $n_L$-th step of the NSM implementation of our Wannier method basis pursuit produces an evolving set of target localized functions $L$. We visualize this evolution in Figure 2.10 which documents the target localized function for the lowest band of the middle plot shown in Figure 2.6 of main text. Comparing $i = n_L$ and $i = 5n_L$ shows that basis pursuit enforces both localization and physical characteristics on Wannier functions. Notice how $D \alpha$ (bottom row) guides the iterative solution to the final form (right most column). The difference between the plots in the last column is directly related to the error between the exact and Wannier interpolated band structures in the middle plot of Figure 6 for the lowest band. After 5 basis pursuit steps our target localized function converges to a form that very closely resembles the converged Wannier function. Basis pursuit chooses a target localized function that minimizes the "fit" error between $X$ and $D$ which indirectly enforces localization and characteristics that are both physical and respectful of symmetry. Larger $\mu$ values, allowed due to a high compatibility between $D$ and the studied $V(r)$, will

\[1\text{We use the $\ell_1$ norm of the Wannier function as a measure of localization following the spirit of Ref. [7].}\]
Figure 2.9: The effects of $\mu$ on localization of Wannier functions. Each plot represents the same Wannier function calculated using a different $\mu$ (indicated by color). The linear slopes demonstrate exponential localization for each $\mu$. Here as $\mu$ increases the linear slope becomes more negative suggesting that larger $\mu$ leads to a more rapid decay of the Wannier functions and an increase in localization.

Figure 2.10: We track the evolution of a single function for $D\alpha$ for results shown in the middle plot of Figure 6 in our main text. Our algorithm implements basis pursuit every $n_L = 10$ iterations and thus dynamically updates $D\alpha$. 
Figure 2.11: The Wannier functions and band structure, obtained from Algorithm 1, for a two-atom type potential (two Gaussians centered at the middle and origin lattice sites) are shown for $\mu = 5$. The exact and Wannier interpolated band structures are shown using black lines and blue dots respectively.

lead to more prominent symmetrical characteristics.

2.6.3 Extrapolating Wannier functions for different potential types

A general Wannier method must be able to deal with $V(r)$ of many different characteristics. Here we demonstrate that our model in fact handles this case by using $D$ from Section 2.6.2 for potential types outside the scope of $V_D$. We use two Gaussian potentials per unit cell, with one of them centered at the origin and the other one at the mid-point $x = y = a/2$ with parameter values $v = 80$, and $\Delta = 0.2$. This non-trivial scenario involves us taking information gained by performing dictionary learning on one-atom systems and applying it to calculate Wannier functions for a two atom system.

Following the guidelines for choosing $\mu$ discussed in the previous section, we obtain highly localized and accurate Wannier functions as shown in Figure 2.11. These results
Figure 2.12: The Wannier functions and band structure, obtained from Algorithm 1, for a single Gaussian potential on a hexagonal lattice are shown. The exact and Wannier interpolated band structures are shown using black lines and blue dots respectively.

show that the information gained from dictionary learning is general, suggesting that there exists a finite database $W_D$ that can be used to build a dictionary $D$ that is compatible with any Gaussian potential within some parameter range.

2.6.4 Extrapolating Wannier functions for different symmetries

Our previous two examples have demonstrated the generality of our method with respect to different potentials for a single lattice configuration. We now show that this generality extends to lattices of different crystal symmetry highlighting the robustness of our Wannier method.

We choose to study a lattice with $C_{6v}$ symmetry and use the symmetrized Wannier dictionary $D$ shown in Figure 2.5 to calculate Wannier functions for the origin centered single Gaussian potential $V(r) = v \exp \left[ -\frac{(r)^2}{2\Delta} \right]$ with $v = 80$ and $\Delta = 0.2$. $D$ was formed by projecting dictionary functions learned in a $C_{4v}$ lattice to a new lattice of $C_{6v}$ symmetry.
Figure 2.13: The Wannier functions and band structure, obtained from Algorithm 1, for a single Gaussian potential on a $C_{4v}$ lattice are shown. The exact and Wannier interpolated band structures are shown using black lines and blue dots respectively. Here we chose a Gaussian potential with depth $v = 80$ and width ($\Delta$) and $\mu$ of $(0.3, 20)$ and $(0.35, 10)$ for the top and bottom plot respectively.

Figure 2.12 shows the accuracy of our Wannier interpolated band structure and highly localized form of our Wannier functions. In taking information learned about Wannier functions in a $C_{4v}$ square lattice, projecting it to six-fold $C_{6v}$ symmetry, and then calculating highly localized and energetically accurate Wannier functions we have shown that the generality of our method extends to many lattice types as well as potential types.

### 2.6.5 Entangled bands

Here we demonstrate our ability to calculate Wannier functions for entangled bands. We enrich the dictionary set $W_D$ by adding pre-calculated entangled Wannier functions using the method of [7, 8] for a two-atom type potential (two Gaussians centered at the middle and origin lattice sites) with $(v, \Delta, \mu) = (80, 0.3, 20)$. Next we combine these functions with the Wannier database shown in Figure 2.2 and calculate a new dictionary to ensure features relevant to entangled bands are learned. Using this dictionary, we obtain Wannier functions and the interpolated band structure shown in Figure 2.13. We conclude that we can use the new entangled dictionary to calculate entangled Wannier functions for potentials not included in the training dataset.
Chapter 3

Calculating Wannier functions using adaptive dictionary learning

3.1 Alternating algorithm

Our approach for numerically solving Eq. (1.5) involves three main parts: simultaneous optimization of $D$ and $\alpha$ (Online Dictionary Learning), optimization of $W$ (Nesterov’s method on the Stiefel Manifold), and symmetrization of $W$. We describe this process in more detail in Algorithm 2. Here we notate real and reciprocal space matrices using bold character and bold character with a tilde, respectively, as is described in section Ref. 2.1.
Algorithm 2: Alternating algorithm for calculating Wannier functions by adapting a Wannier dictionary to a physical environment. Symmetrization is performed using the methods described in Ref. [15]

Input: \( \tilde{H}, D, B, A, W_{D_{\text{sub}}}, \mu_{\text{sched}}, \epsilon_{\text{tol}}, \epsilon_{L}, s_{\text{in}} \);

Initialize: set \( W \) as random Gaussians

\begin{algorithmic}
\FOR{\( \mu \in \mu_{\text{sched}} \)}
    \STATE \textit{—— Symmetrize Wannier Functions ———};
    \( \tilde{W} = \text{Symmetrize}(\tilde{W}) \);
    \STATE \textit{—— Adapt Dictionary ————–};
    \( D, B, A, \alpha = \text{OnlineDictionaryLearning}(W, W_{D_{\text{sub}}}, D, B, A) \);
    \STATE \( L = D\alpha \);
    \STATE \textit{—— Minimize Energy ————–};
    \( \tilde{W} = \text{NesterovStiefelManifold}(\tilde{H}, \tilde{W}, \tilde{L}, \mu) \);
\ENDFOR
\RETURN\( W, D, B, A \)
\end{algorithmic}

3.1.1 Online dictionary learning

Online algorithms [36] can be used to solve optimization problems by processing a training dataset "piece-by-piece" in a sequential fashion. Each element of a dataset is treated independently and used to update optimization variables. For applications that gather data over a long period of time, online algorithms can allow trained variables to adapt to new data while still retaining some knowledge of previously seen data.

We choose to solve Eq. 1.4 using an online algorithm for dictionary learning (ODL) so that our dictionary can be systematically improved between each alternating iteration.
of Algorithm 2. In addition, as we calculate new physical systems our dictionary will evolve to contain new Wannier features while preserving previous learned Wannier features. Pseudo code for ODL is provided in Algorithm 3 and described further in Ref. [10].

**Algorithm 3:** Online dictionary learning [10] which solves Eq. (1.4).

Input: $\Phi = \text{RandomShuffle}(W \cup W_{D_{sub}})$, $D^{(0)}$, $B^{(0)}$, $A^{(0)}$;

Define: $\alpha \in \mathbb{R}_{N\Phi \times N_D}$;

for $t = 1$ to $N_{\Phi}$ do

$\alpha_t = \text{BasisPursuit}(\Phi_t, D^{(t)})$;

$A^{(t)} = A^{(t-1)} + \alpha_t \alpha_t^T$;

$B^{(t)} = B^{(t-1)} + \Phi_t \alpha_t^T$;

$D^{(t)} = \text{DictionaryUpdate}(D^{(t-1)}, B^{(t)}, A^{(t)})$;

end

Return: $D^{(N_{\Phi})}$, $B^{(N_{\Phi})}$, $A^{(N_{\Phi})}$, $\alpha$;

Here $\Phi$ represents our training data and is chosen to include the current $W$, the output of symmetrization in Algorithm 2, as well as a random subset of Wannier functions from the database $W_{D_{sub}}$. This ensures that our training data approximates independent and identically distributed (i.i.d.) sampling and as a result our dictionary will retain features of previously calculated Wannier functions, $W_D$, while learning new features. In the limit that $W_D$ is very large, $\Phi$ becomes a good approximation of i.i.d sampling. When $W_D$ is small the dictionary we learn will be highly personalized to $W$.

In Algorithm 3 we sequentially loop through our training data and notate the value of a variable at iteration $t$ as $(t)$ and a matrix column as subscript $t$. For each element of our training dataset we alternate between optimizing $\alpha$ using basis pursuit and $D$ using a dictionary update algorithm.
Both A and B define "prior" information correlating spatial and basis information. If we desire to learn a dictionary from "scratch" we can set $A^{(0)} \leftarrow 0$, $B^{(0)} \leftarrow 0$, and $D^{(0)}$ to Gaussians with random widths and noise. However, if we want to continue a dictionary using new data we will need to initialize $A^{(0)} = A^{(N\Phi)}$, $B^{(0)} = B^{(N\Phi)}$, and $D^{(0)} = D^{(N\Phi)}$ in Algorithm 3.

**Basis pursuit**

Basis pursuit consists of minimizing Eq. (1.4) for $\alpha$ only:

$$
\min_{\alpha} \| \Phi_t - D\alpha \|_F^2 + \frac{\gamma}{\mu} \| \alpha \|_1.
$$

(3.1)

Because the $\ell_1$-norm is not differentiable we cannot use standard gradient based approaches to solve the problem. One method of getting around this is to introduce a splitting term, based on the split Bregman iteration [37, 38], so that the optimization problem becomes

$$
\min_{\alpha, \beta} \| \Phi_t - D\alpha \|_F^2 + \frac{\gamma}{\mu} \| \beta \|_1 + \frac{\lambda}{2} \| \alpha - \beta + b \|_F^2,
$$

(3.2)

which results in three-sub problems;

$$
\min_{\alpha} \| \Phi_t - D\alpha \|_F^2 + \frac{\lambda}{2} \| \alpha - \beta + b \|_F^2,
$$

(3.3)

$$
\min_{\beta} \frac{\gamma}{\mu} \| \beta \|_1 + \frac{\lambda}{2} \| \alpha - \beta + b \|_F^2,
$$

(3.4)

and $b \leftarrow b + \alpha - \beta$. By alternating between solving these sub-problems, while keeping the other variables fixed, we can obtain the solution to Eq. (3.2). Practically we use the alternating approach described in Algorithm 4.
**Algorithm 4:** Split Bregman basis pursuit alternating algorithm

Input: $\Phi_t, D, N_i, N_j$;

Initialize: $\alpha^{(0)} = \beta^{(0)} = b^{(0)} = 0, \lambda = 1$;

for $i = 0$ to $N_i$ do

    for $j = 0$ to $N_j$ do

        $\alpha^{(j+1)} = \text{ConjugateGradient}(\alpha^{(j)}, \beta^{(j)} + b^{(i)}, \Phi_t, D, \lambda)$;

        $\beta^{(j+1)} = \text{SoftThresholding}(\alpha^{(j+1)} + b^{(i)}, \lambda, \mu, \gamma)$;

    end

    $b^{(i+1)} = b^{(i)} + \alpha^{(N_j)} - \beta^{(N_j)}$;

    $\alpha^{(0)} = \alpha^{(N_j)}, \beta^{(0)} = \beta^{(N_j)}$;

end

Return: $\alpha^{(N_j)}$;

Here it is possible to use a gradient based algorithm, such as conjugate gradient descent [39], for solving Eq. (3.3) because the $\ell_1$-norm is isolated to Eq. (3.4), which is solved using soft-thresholding. Soft-thresholding establishes a compact support by ”chopping” all values beneath an absolute value threshold to zero. The residual variable $b$ has the meaning of the Lagrange multiplier and helps convergence by keeping track of the difference between $\alpha$ and $\beta$. 
Dictionary update

We can start to derive the online dictionary update algorithm [10] shown in Algorithm 5 by reformatting Eq. (1.4) to only have $D$ optimization;

$$\min_{D} \frac{1}{2t} \sum_{i=1}^{N_t} \| \Phi_i - D\alpha_i \|_2^2 \quad \text{s.t.} \quad \|D_m\|_2 \leq 1. \quad \text{(3.5)}$$

We can expand the objective function as

$$\| \Phi_i - D\alpha_i \|_2^2 = (\Phi_i - D\alpha_i)^T(\Phi_i - D\alpha_i)$$

$$= \Phi_i^T\Phi_i - \Phi_i^TD\alpha_i - D^T\alpha_i^T\Phi_i + D^T\alpha_i^TD\alpha_i, \quad \text{(3.6)}$$

and rewrite the optimization problem as

$$\min_{D} \frac{1}{t} \sum_{i=1}^{t} \left( \frac{1}{2}(D^T\alpha_i^TD\alpha_i) - D^T\alpha_i^T\Phi_i \right) \quad \text{s.t.} \quad \|D_m\|_2 \leq 1. \quad \text{(3.7)}$$

Using trace relations we can further simplify the optimization problem reaching the final form

$$\min_{D} \frac{1}{t} \sum_{i=1}^{t} \left( \frac{1}{2} \text{Tr} D^TDA - \text{Tr} D^TB \right) \quad \text{s.t.} \quad \|D_m\|_2 \leq 1, \quad \text{(3.8)}$$

where $A = \sum_{i}^{t} \alpha_i\alpha_i^T$ and $B = \sum_{i}^{t} \Phi_i\alpha_i^T$ correlate the spatial and dictionary function information for the first $t$ data elements of $\Phi$. Following the work of [10] we can devise an parameter-free algorithm from Eq. (3.8) by using block-coordinate descent. Here each dictionary function is treated independently and Algorithm 5 is guaranteed to reach global convergence due to the separability of constraints in Eq. (3.8) [40].
Algorithm 5: Dictionary update algorithm

Input: $D^{(0)} = D^{(t-1)}$

while ($|D^{(i+1)} - D^{(i)}| > \epsilon$, $i++$) do

  for $j = 1$ to $N_D$ do
    $d = \frac{1}{A_{jj}}(B_j - D^{(i)}A_j) + D^{(i)}$
    $D^{(i+1)}_j = \frac{1}{\max(||d||_2,1)}d$
  end

end

Return: $D^{(f)}$

The importance of $\alpha$ sparseness

The parameter $\gamma$ controls the sparsity of $\alpha$ and thus how many functions of the dictionary, $D$, are used to enforce Wannier features on $W$. When $\gamma \ll 1$, $||\alpha||_0 \approx N_D$ resulting in a non-sparse $\alpha$. In this case every dictionary function is used to establish the localized target $L = D\alpha$ for any $W$ (although the coefficient values of $\alpha$ will vary). Here our dictionary functions will not specialize to the dataset and no meaningful Wannier character will be learned.

In the case that $\gamma$ is very large then only one dictionary function will be used to fit $W$ meaning each dictionary function will become approximately a training WF. As a result, our dictionary will not learn general features and our algorithm will simply try to project a known solution as a target for a new Wannier problem.

In early work on sparse coding, Olshausen and Field describe the importance of sparseness based on the observation that natural images may generally be described as a linear combination of ”structural primitive” objects [41]. Thus the optimum case is when $\gamma$ is
finite which allows the dictionary to generalize to the dataset and the individual dictionary functions to have specialized character. A finite $\gamma$ forces $\alpha$ to have only a finite number of non-zero coefficient values, sparseness, which as a result means only a small subset of the dictionary can be used to fit a given WF. This forces dictionary functions to act like structural primitives objects and specialize at reconstructing a portion of each WF. Dictionaries with specialized functions in this way are more capable at transferring useful physical information to WFs outside the training dataset.

Furthermore, Olshausen and Field demonstrate that sparseness of $\alpha$ is only an appropriate strategy when the training data is in fact localized [41]. Sparse coding, or dictionary learning, is not a general principle for finding statistically independent components of data and should not be used for data types that are multi-modal in behaviour with heavy peaks around non-zero values.

**Dictionary initialization**

There are a few general principles that we follow when initializing dictionaries to be used in Algorithm 2.

We have noticed a sensitivity due to the initialized localization of the dictionary. If the dictionary is not sufficiently localized, $L^{(0)}$ can force the $W^{(1)}$ to delocalize to the localized bounds of $L^{(0)}$. Because our goal is to calculate highly localized Wannier functions we adopt the practice of ensuring the dictionary is initialized with very localized functions and allow feedback from the total energy to gradually delocalize the dictionary until we reach balance between localization and energy accuracy. During this process the Wannier functions will gradually become more localized as well.

When learning from ”scratch”, we initialize our dictionaries as Gaussians with added random character and varying spreads so that they are still well localized within our dictionary grid. We have not noticed a sensitivity to the degree of randomness, or range of
spreads but each dictionary function should be unique.

3.1.2 Nesterov’s method on the Stiefel manifold

Here we perform minimization of the total energy subject to our orthonormal constraint by using Nesterov’s accelerated gradient method on the Stiefel manifold, described in detail in section 6.3 and [16, 17]. Although Algorithm 6 is similar to a portion of Algorithm 1, we list it here to highlight the fact that in this case no $\alpha$ minimization occurs. Here for a fixed and symmetrized $\tilde{\mathbf{L}}$, we alternate between performing $\text{GradStep}$ and $\text{Extrapolate}$, discussed extensively in section 6.3, until $\mathbf{X}$ has converged. Though $\mathbf{X}$ is the "Wannier" variable in this algorithm we refrain from calling it such until Algorithm 2 has converged.
Algorithm 6: Nesterov’s method on the Stiefel manifold (NSM) using a static localized target, $\mathbf{L}$.

Input: $\mathbf{H}$, $\mathbf{Y}^{(0)}$, $\mathbf{L}$, $\epsilon_{\text{tol}}$, $\epsilon_{L}$, $s_{\text{in}}$, $\mu$;

Initialize: $i = j = \beta = 0$, $s = s_{\text{in}}$;

while $(\mathbf{X}^{(i+1)} - \mathbf{X}^{(i)}) < \epsilon_{\text{tol}}$, $i++$ do

for $k \in \text{BZ}$ do

$\mathbf{X}_{[k]}^{(i+1)} = \text{GradStep}(\mathbf{Y}^{(i)}_{[k]}, \mathbf{L}^{(i)}_{[k]}, \mathbf{H}^{(i)}_{[k]}, s, \mu)$;

end

if $F(\mathbf{X}^{(i+1)}) < F(\mathbf{X}^{(i)})$ then

$j += 1$, $\beta = \frac{j}{j+3}$;

for $k \in \text{BZ}$ do

$\mathbf{Y}^{(i+1)}_{[k]} = \text{Extrapolate}(\mathbf{X}_{[k]}^{(i+1)}, \mathbf{X}^{(i)}_{[k]}, \beta)$;

end

else

$j = \beta = 0$, $\mathbf{X}^{(i+1)} = \mathbf{Y}^{(i+1)} = \mathbf{X}^{(i)}$;

end

end

Return: $\mathbf{Y}^{(f)}$;

3.1.3 Practical considerations

In practice we solve Algorithm 2 for lattice symmetry specific discrete meshes. This complicates creating a general $\mathbf{D}$ because it will need to be learned over data with different spatial meanings for each mesh value, or matrix element. To deal with this we store the spherical harmonic (SH) decomposition of $\mathbf{D}$ and $\mathbf{B}$. From here, interpolation can be
used to reconfigure $D_{SH}$ and $B_{SH}$ for any real space lattice which is done prior to running Algorithm 2. Details about how to do this can be found in the last two chapters of [42].

Another complication occurs when preparing our training data $\Phi$ for dictionary learning. We must ensure that each function of $\Phi$ is centered about the same real space point which establishes a center for the dictionary functions. To do this we again use spherical harmonic interpolation which increases the cost of each alternating iteration. While it may be possible to learn a dictionary without aligning the centers of the training data doing so would not take advantage of the significant reduction the amount of training data that aligning $\Phi$ centers offers. This reduction in training data will make it easier for dictionary learning to learn more general features.

### 3.2 Material simulations

Here we apply Algorithm 2 in three dimensions (3D) to study semiconductors silicon (Si), tin telluride (SnTe), and lead telluride (PbTe). These calculations used a density functional theory (DFT) package with local density approximation (LDA) exchange-correlation functionals, a planewave basis set, and norm-conserving pseudopotentials.

For each material Wannier initial guesses are generated using hydrogen orbitals (for radial character) and angular momentum (for angular character). The components of the angular momentum are chosen so the initial guesses have the correct symmetry required of the Wannier function. In each example we use either a dictionary initialization from "scratch" using the scheme described in Section 3.1.1 or a warm restart. We choose $\gamma = 100$, a dictionary grid of $1/3$ of the full super-cell grid (see section 2.5 for more discussion), and $N_D = 50$. For three dimensions this $N_D$ is not significantly over-complete, however our choice of $\gamma$ still allows us to learn specialized features. When implementing this approach for larger systems and for more bands $N_D$ should be increased to account for
the additional Wannier character that the dictionary needs to learn. We do not anticipate
Wannier results being sensitive to a choice of $N_D$ and $\gamma$ as long as our discussion in section
3.1.1 is respected.

### 3.3 Silicon

We study the cubic system of silicon with primitive lattice vectors $l_1 = a[1/2, 1/2, 0]$, $l_2 = a[1/2, 0, 1/2]$, and $l_3 = a[0, 1/2, 1/2]$ with Si atoms positioned at the primitive
lattice sites $r_1 = [0, 0, 0]$ and $r_2 = a[1/4, 1/4, 1/4]$. We chose an energy cut-off of 10
Hartrees, a lattice constant of $a = 10.26 \ a_0$, and a $12 \times 12 \times 12$ super-cell. We followed

In our first example we calculate Wannier functions for the six lowest energy bands
using $\mu_{\text{sched}} = [50, 2.5, 0.125, 0.00625]$, for 5 alternating iterations per $\mu$, to illustrate the
effects of $\mu$ on Algorithm 2. Figure 3.1 documents the progression of our alternating
approach by displaying Wannier interpolated band structures for $\mu = 50$ (top plot - 5 alternating iterations), $\mu = 2.5$ (middle plot - 10 alternating iterations), and $\mu = 0.00625$
(bottom plot - 20 alternating iterations). Notice that as $\mu$ decreases the Wannier interpo-
lated bands become more accurate.

We re-plot the Wannier interpolated band structure for the final alternating iteration
(bottom plot) along with the symmetrical $s$-like and $p$-like Wannier functions for the low-
est six bands. The Wannier functions are real-valued and multiple amplitudes were used to
construct 3D renderings. Figure 3.3 displays this $p$-like orbital ($W$) and the corresponding
localized target ($D\alpha$) for a specific amplitude value. We display the dictionary functions
with the three largest $\alpha$ coefficient values. Notice that they appear specialized to the train-
ing data ($W$), which makes sense because $\alpha$ is strongly sparse. More $s$-like and $p$-like
specialized dictionary functions are shown in Figure 3.4.
Figure 3.1: Algorithm 2 progress for Si. Here we display Wannier interpolated band structures (dashed lines) for $\mu = 50$ (top plot - 5 alternating iterations), $\mu = 2.5$ (middle plot - 10 alternating iterations), and $\mu = 0.00625$ (bottom plot - 20 alternating iterations).
Figure 3.2: The final alternating iteration of the implementation of Algorithm 2 in Figure 3.1 ($\mu = 0.00625$). Symmetrical $s$-like and $p$-like Wannier functions are displayed using multiple amplitudes values.
Figure 3.3: The $p$-like orbital ($W$) from Figure 3.2 and the corresponding localized target ($D\alpha$) are plotted for a specific amplitude value. We display the dictionary functions with the three largest $\alpha$ coefficient values and the sparse coefficient vector $\alpha$. 

$D_i \ |\alpha_i| = 1.15$  
$D_j \ |\alpha_j| = 0.32$  
$D_k \ |\alpha_k| = 0.29$
3.3.1 Convergence and Stability

We investigate the total energy convergence of Algorithm 2 using both a dictionary initialized from scratch (\( D_{\text{Random}} \)) and a warm restart corresponding to the results shown in Figures 3.2, 3.3 and 3.4 (\( D_{\text{Si}} \)). These results are displayed in Figure 3.5 where the total energy is continuously minimized with each iteration and each change in \( \mu \). The difference between the red (top) and blue (bottom) plots at \( \mu = 50 \) demonstrates that Wannier
features were learned by $D_{Si}$ and transferred more effectively. However, while $D_{Si}$ allows for slightly a faster progression to the ground state it is slowed down by the adapting to the un-converged Wannier character for smaller $\mu$. In practice if we knew ahead of time that we had a quality dictionary for Si we would use the more efficient fixed dictionary method described by Algorithm 1 to obtain Wannier functions.

Both NSM and ODL are stable methods that converge to stationary points of their respective optimization problems [16, 17, 43]. We now investigate the stability of alternating between these two independently stable problems (Algorithm 2). Here we will neglect the effect of symmetry because it should not significantly increase the non-convexity of the problem. Constraining symmetry involves obtaining solutions for each $k$ within the irreducible wedge of the Brillouin zone (independent variables) and copying these solutions for symmetry equivalent points in the entire Brillouin zone.

In our approach the first step is to use the initial guesses for the Wannier functions $W^{(0)}$ to calculate $D^{(1)}$. From here $D^{(1)}$ can be used to calculate the localized target function $L^{(1)}$ which along with NSM allows us to obtain the Wannier variable at the next iteration, $W^{(1)}$. In the case that $\mu >> 1$, $W^{(1)} \approx W^{(0)}$ and due to the total energy term $W^{(1)} \hat{H}W^{(1)} \approx W^{(0)} \hat{H}W^{(0)}$. This suggests that the manifolds for NSM and ODL would not change significantly after the first alternating iteration and we would expect $D^{(2)} \approx D^{(1)}$. However, the subtle differences between the two dictionaries are directly related to $W$ character that produces a lower energy state. This suggests that through alternation we can systematically achieve a $W$ that approximate the ground state by adapting our dictionary. We support this numerically in Figure 3.5 where we compare the energy convergence of two different dictionary initialization for Si. Notice the stability of the curves for the same $\mu$. As we relax $\gamma$, $(W^{(i+1)} - W^{(i)})$ and $(W^{(i+1)} \hat{H}W^{(i+1)} - W^{(i)} \hat{H}W^{(i)})$ will increase which is demonstrated by comparing the slope between $t = 4, 5$ and $t = 5, 6$. Notice that the energy continuously decreases even with large changes in $\mu$ demonstrating
the numerical stability of our implementation.

Our $\mu$-schedule acts as a "freezing" process which starts with large $\mu$ values to enforce localization and as a result only slightly alters the dictionary to be more physical. We do this to ensure that any relevant dictionary character is transferred before it is affected by the first few iterations of $W$. From here we decrease $\mu$ as is shown in Figure 3.5 to allow the total energy to have more influence on $W$, which indirectly removes the error due to the $D$. As $\mu$ decreases, the ability of the dictionary to enforce character decreases. In practice however we have not experienced an inability to localize the Wannier functions or significant delocalization for smaller ending $\mu$ values. However, if significant delocalization does occur for small $\mu$ we would suggest fixing $D$, adjusting $\mu$, and using Algorithm 1 (static approach).
3.4 Tellurium materials

While it may be possible to have a single Wannier dictionary, it also is logical to have a dictionary for a material family. Doing so would allow for a smaller $N_D$ because less unique features are needed to be learned. Here we demonstrate implementing our approach for two telluride compounds (SnTe and PbTe).

For SnTe we use most of the primitive-cell, super-cell, and dictionary parameters that were used in our Si example, when the dictionary was initialized from scratch. The only parameter differences are choosing a lattice constant of $a = 11.9 \ a_0$ and atom placements of $r_{Sn} = [0, 0, 0]$ and $r_{Te} = a[1/2, 0, 0]$. In addition we use the same algorithm parameters, including the $\mu$-schedule, that were used for Si.

Figure 3.6 displays the symmetrized $s$-like and $p$-like Wannier functions calculated for SnTe. The interpolated band structure (red dashed) is provided for $\mu = 0.00625$ at alternating iteration 20. The only significant deviation of the interpolated from the exact band structure occurs at k-point W which can be attributed to the exclusion of $d$-type Wannier functions that only contribute to higher bands.

The dictionary $D_{SnTe}$ obtained from our previous calculation will have features relevant to compounds containing tellurium. Because of this we rename this dictionary $D_{Te}$, and demonstrate that it can be used as a warm restart to calculating Wannier functions for PbTe. For this case, all other dictionary, primitive-cell, super-cell, and algorithmic parameters are the same as in our SnTe example besides using a lattice constant of $a = 12.2 \ a_0$, replacing the Sn atom with a Pb atom at $r_{Pb} = [0, 0, 0]$, a $16 \times 16 \times 16$ super-cell, and $\mu_{sched} = [50, 5, 0.5, 0.05]$ for 10 alternating iterations per $\mu$.

Figure 3.7 displays the symmetrized $s$-like, $p$-like, and $d$-like Wannier functions corresponding to the interpolated band structure (red dashed) for $\mu = 0.05$ and alternating iteration 40. The only significant deviation again occurs in the top most bands which can
Figure 3.6: Symmetrized $s$-like and $p$-like Wannier functions calculated for SnTe. The interpolated band structure (red dashed) is provided for $\mu = 0.00625$ at alternating iteration 20.
Figure 3.7: Symmetrized $s$-like, $p$-like, and $d$-like Wannier functions calculated for PbTe corresponding to the interpolated band structure (red dashed) for $\mu = 0.05$ and alternating iteration 40.

be attributed to the exclusion of Wannier functions of $d$ character for Te that only occur in higher bands.
Chapter 4

Deep learning for quantum many-body physics

4.1 Background

We use many-body basis states detailing the occupation of particle spin configurations on a lattice. Basis states for a lattice of $L$ sites are represented as

$$x = |s_1, ..., s_L\rangle,$$  \hspace{1cm} (4.1)

where the spin state on each site is $s_i = -n, -n + 1, ..., n$ for $n$-spins. We choose to notate a single basis state as $x$ for convenience and the corresponding wave function value as $\Psi(x)$. For $n$ possible spin states per lattice site, the complete set of $n^L$ basis states, $S^T$, is needed to evaluate the wave function fully. Unfortunately, the astronomical size of $S_T$ makes the direct calculation of the many-body ground state intractable in the general case. To see this we can look at the definition of the average energy,

$$E = \frac{\langle \Psi | \hat{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle}.$$  \hspace{1cm} (4.2)
Not only do we need to evaluate the wave function for $n^L$ basis states, but we also need to calculate the Hamiltonian for this enormous basis set. Just the calculation of the average energy alone would have the complexity of $O(n^{2L})$. Even in the case of a small lattice, say SU(2) and $L = 16$, the basis set contains $\sim 6.5 \times 10^4$ states and the full complexity of the average energy is $\sim 4.3 \times 10^9$. Fortunately it is common for the Hamiltonian to be sparse meaning only local interactions between particles are significant. This makes it easier to evaluate Eq. (4.2), but it is often still too expensive to do so exactly.

Deep learning enhanced Monte Carlo approaches are practical methods that have been used to obtain the many-body ground state [22]. These approaches create a flexible representation of the wave function using a deep neural network (DNN) and are optimized by alternating between MC sampling and neural network training as is illustrated in Figure 1.1. First, Monte Carlo (MC) is used to sample a set of $N_{\text{samples}}$ basis states, $S \subset S_T$. $S$ is obtained in such a way that ensures that the probability to encounter a state $x$ is $P(x) \approx |\Psi(x)|^2$ for $x \in S$. The network is trained to approximate the wave function by minimizing Eq. (4.2) over the set $\{x\} = S$.

The DNN relies on MC for its input, but MC also relies on the DNN to provide wave function predictions for $x \in S$, which it then uses to accept or reject basis states in a way that favors high probability. We highlight that without accurate predictions for the wave function, MC cannot identify a high quality $S$, and without a high quality $S$ the DNN cannot learn network parameters (weights, $w$) that perform an accurate mapping to the ground state wave function. This issue is dealt with by alternating between MC and DNN training, as illustrated by Figure 1.1, where the DNN is improved by using a loss function based on a variational minimization of Eq. (4.2).
4.2 DNN training

Training a DNN to predict accurate wave function values given a set of basis states relies on establishing a loss function based on the average energy. We can expand the average energy as

\[ E = \sum_{x,x'} \Psi^*(x) H_{xx'} \Psi(x') = \sum_x |\Psi(x)|^2 \sum_{x'} H_{xx'} \frac{\Psi(x')}{\Psi(x)}, \]  

which can be approximated using our MC sampling \( S \) as

\[ E \approx \frac{1}{|S|} \sum_{x \in S} \sum_{x'} H_{xx'} \frac{\Psi(x')}{\Psi(x)}, \]  

where \( H_{xx'} = \langle x | \hat{H} | x' \rangle \). This approximation is allowed because \( x \in S \) is distributed according to \( |\Psi(x)|^2 \). Note that \( |x'\rangle \) are not sampled basis states but rather a finite set of "ket" basis states specific to each "bra" \( \langle x | \) for which \( H_{xx'} \neq 0 \). An important part of Eq. (4.4) is the local energy,

\[ E_x = \sum_{x'} H_{xx'} \frac{\Psi(x')}{\Psi(x)}, \]  

which allows us to think about our approximation of the average energy as an average of local energies for \( S \);

\[ E \approx \frac{1}{|S|} \sum_{x \in S} E_x. \]  

The error of this approximation is correlated to the number of high probability basis states of \( S_T \) that are left out from \( S \). In addition some of the error is purely statistical, such that the counts of each state deviate from \( |\Psi(x)|^2 \). Thus, it is the role of MC to learn these significant basis states and include them in our sampling.
Because a standard feed-forward network is comprised of a series of nested mappings of an input, in order to optimize the network we must establish a loss based on the output of the network. Then using backpropagation we use the gradient of the loss to establish the gradient for all parameters in the network, which are commonly referred to as weights \(w\), and then use a gradient based algorithm such as stochastic gradient descent [44] to change \(w\) in such a way that the loss is decreased. We now derive our loss function for the problem at hand. The gradient of (4.2) is

\[
\partial_w E = \left\langle \frac{\partial \Psi}{\partial w} \right| \frac{\partial E}{\partial \Psi^*} \right\rangle + \left\langle \frac{\partial E}{\partial \Psi} \right| \frac{\partial \Psi}{\partial w} \right\rangle \\
= \frac{\left\langle \partial_w \Psi \right| \hat{H} \left| \Psi \right\rangle + c.c}{\left\langle \Psi \right| \Psi \rangle} - \left( \left\langle \partial_w \Psi \left| \Psi \right\rangle + c.c \right) \frac{\left\langle \Psi \right| \hat{H} \left| \Psi \right\rangle}{\left\langle \Psi \right| \Psi \rangle^2} \quad (4.7)
\]

\[
= \frac{2 \text{Re} \left[ \left\langle \partial_w \Psi \right| \hat{H} \left| \Psi \right\rangle - E \left\langle \partial_w \Psi \right| \Psi \rangle \right]}{\left\langle \Psi \right| \Psi \rangle},
\]

whose first term can be expanded to as

\[
\left\langle \partial_w \Psi \right| \hat{H} \left| \Psi \right\rangle = \sum_{x,x'} \left\langle \partial_w \Psi \right| x \left\rangle \left( x \right| \hat{H} \left| x' \right\rangle \right\langle x' \right| \Psi \right\rangle \\
= \sum_{x,x'} \partial_w \Psi^*(x) H_{xx'} \Psi(x') \\
= \sum_x \left| \Psi(x) \right|^2 \partial_w \ln \Psi^*(x) \sum_{x'} H_{xx'} \frac{\Psi(x')}{\Psi(x)} \\
= \sum_x \left| \Psi(x) \right|^2 E_x \partial_w \ln \Psi^*(x), \quad (4.8)
\]

where we used the fact that \(\partial_w \ln \Psi^*(x) = \frac{\partial_w \Psi^*(x)}{\Psi^*(x)}\) and multiplied by one \(\left( \frac{\Psi(x)}{\Psi^*(x)} \right)\) within the third line. Using the same trick as above, we can approximate Eq. (4.8) by only summing over \(S\) resulting in

61
\[
\frac{\langle \partial_w \Psi | \hat{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle} \approx \frac{1}{|S|} \sum_{x \in S} E_x \partial_w \ln \Psi^*(x).
\] (4.9)

The second term of Eq. (4.7) can be expanded as

\[
\langle \partial_w \Psi | \Psi \rangle = \sum_x \langle \partial_w \Psi | x \rangle \langle x | \Psi \rangle = \sum_x \partial_w \Psi^*(x) \Psi(x) = \sum_x \Psi(x)^2 \partial_w \ln \Psi^*(x)
\] (4.10)

which can be approximated using $S$ as

\[
\frac{\langle \partial_w \Psi | \Psi \rangle}{\langle \Psi | \Psi \rangle} \approx \frac{1}{|S|} \sum_{x \in S} \partial_w \ln \Psi^*(x).
\] (4.11)

The full approximation of the energy gradient is

\[
\partial_w E \approx 2 \frac{\sum_{x \in S} \text{Re} \left[ \left( E_x - E \right) \frac{\Psi(x)}{\Psi(x)^2} \partial_w \Psi^*(x) \right]}{|S|}
\] (4.12)

where $E$ is defined by Eq. (4.12). In practice most coding libraries such as Tensorflow [45] only require the definition of a loss function from which automatic calculation of the gradients for the network variables will be carried out. In this case we can use a gradient trick where we pre-calculate the constant terms in Eq. (4.7),

\[
g(x) = \frac{2}{|S|} \text{Re} \left[ \left( E_x - E \right) \frac{\Psi(x)}{\Psi(x)^2} \right]
\] (4.13)
and then define the loss function for the network to be

\[ \mathcal{L} = \sum_{x \in S} g(x)\Psi^*(x). \]  \hspace{1cm} (4.14)

Note that \( g(x) \) needs to be recalculated after each update of the DNN weights \( w \).

### 4.2.1 Monte Carlo

Pseudo code for a basic MC algorithm that uses the wave function in the acceptance criterion is shown in Algorithm 7. While MC is effective at identifying the important ground state basis states from \( S_T \), standard MC-DNN approaches require us to sample a new \( S \) for each DNN training step. This can be very expensive limiting the practicality of the alternating approach.
Algorithm 7: Monte Carlo (MC) based on the Metropolis Hastings algorithm [46]. Here we generate a set of unique basis states $S_\neq$ drawn from the distribution $P(x) \propto |\Psi(x)|^2$. We may choose to only select every $N_{\text{skip}}$ basis states to obtain a statistically independent chain. The trial state is obtained making a single exchange from the current state, $x$.

Input: $N_{\text{sites}}, N^\uparrow, N^\downarrow, N_{\text{sample}}$;

Initialize: $S_\neq = \emptyset, x = \text{GenerateState}(N_{\text{sites}}, N^\uparrow, N^\downarrow)$;

for $i = 1 \ldots N_{\text{sample}}$ do
  $x_{\text{trial}} = \text{GenerateTrial}(x)$;
  $r_{\text{trial}} = \frac{|\Psi(x_{\text{trial}})|^2}{|\Psi(x)|^2}$;
  $r = \text{Random}(0, 1)$;
  if $r \leq r_{\text{trial}}$ then
    $x = x_{\text{trial}}$;
    $S_\neq = S_\neq \cup \{x\}$;
  end

4.2.2 Neural network optimization via multi-use MC

If we simply reused $S$ for multiple network updates, the average energy (4.6) and gradient (4.12) will be incorrect because the sum over $x$ pertains to state of the network at the time of sampling. Fortunately, the authors of [22] have shown that importance sampling gradient optimization (ISGO) allows us to reuse $S$ for multiple DNN training steps if we alter our expression for the average energy,

$$E \approx \frac{C}{|S_0|} \sum_{x \in S_0} \frac{|\Psi(x)|^2}{|\Psi_0(x)|^2} E_x,$$

(4.15)

where the normalization constant,
\[ C = \left( \frac{1}{|S_0|} \sum_{x \in S_0} \frac{|\Psi(x)|^2}{|\Psi_0(x)|^2} \right)^{-1}, \quad (4.16) \]

ensures the weights add up to one. Here \( S_0 \) and \( \Psi_0 \) are the basis sampling and network at the time of sampling respectively. \( \Psi \) represents a network that is optimized \( n \) times after being initialized as \( \Psi_0 \). Notice that in Eq. (4.15) we scale the local energies \( E_x \) using a ratio of the probabilities for the network at different training steps for each \( x \in S_0 \). This change to the average energy, which is analogous to non-Boltzman sampling in literature on classical MC, adapts the \( S_0 \) to new networks \( \Psi \) so long as \( \Psi \) and \( \Psi_0 \) do not differ considerably. This is required because high-probability regions of \( \Psi(x) \) and \( \Psi_0(x) \) are usually localized with respect to \( x \) which means that when the two networks differ substantially their high-probability regions will not overlap and potentially important states will be missed.

In practice we adopt a simpler approach which avoids the need to keep track of \( \Psi_0(x) \) and frequencies of basis states, \( f_x \), in the chain. The distinct set of basis states in the chain, \( S_\neq \), are utilized to reduce the size of the sums of basis configurations in the above equations. We can rewrite the average energy in terms of frequencies and distinct basis states as

\[ E \approx C \frac{1}{|S_0|} \sum_{x \in S_\neq} f_x \frac{|\Psi(x)|^2}{|\Psi_0(x)|^2} E_x, \quad (4.17) \]

where the average of frequency \( f_x \) is defined as

\[ \bar{f}_x = \frac{|S_0|}{\sum_{x' \in S_0} |\Psi_0(x')|^2} \frac{|\Psi_0(x)|^2}{\sum_{x' \in S_0} |\Psi_0(x')|^2}. \quad (4.18) \]

Because changes to the network should be small \( \Psi \approx \Psi_0 \), which allows us to rewrite the
approximation of the average energy as

\[
E \approx \left( \sum_{x' \in S_0} |\Psi(x')|^2 \right)^{-1} \sum_{x \in S_{\neq}} \sum_{x' \in S_{\neq}} |\Psi(x)|^2 E_x. \tag{4.19}
\]

Now the gradient can be written as

\[
\partial_w E \approx \sum_{x \in S_{\neq}} |\Psi(x)|^2 \text{Re} \left[ (E_x - E) \Psi(x) \partial_w \Psi^*(x) \right], \tag{4.20}
\]

where our gradient trick of pre-calculating constant terms turns into

\[
g(x) = |\Psi(x)|^2 \text{Re} \left[ (E_x - E) \Psi(x) \right] \tag{4.21}
\]

and our loss is still defined by Eq. (4.14). Our approach still preserves detailed balance which is the necessary requirement of Markov chain Monte Carlo methods.

Our multi-use MC training method is defined in Algorithm 8. Unless we are close to convergence, \(S_{\neq}\) can only be used a finite number of times. Still, this avoids redundant calculations of Algorithm 7 and makes for more efficient training of the neural network. While Algorithm 8 uses a single \(N_{\text{optimize}}\) for all iterations, using a smaller value in the earlier stages of training and larger value as we approach convergence can be implemented to make training even more efficient.
Algorithm 8: Neural network optimization via multi-use Monte Carlo (MC)

Input: $\epsilon_E, N_{\text{optimize}}$;

Initialize: $\Psi$ network with random weights using Eq. (4.14) as loss;

\[
\text{while } (E - E_0) > \epsilon_E \text{ do }
\]

\[
S_{\neq} \leftarrow \text{MC from Algorithm 7};
\]

\[
\text{for } i = 1,...,N_{\text{optimize}} \text{ do}
\]

\[
g(x) \leftarrow \text{Eq. (4.21)};
\]

\[
\Psi \leftarrow \text{Backpropagation and loss minimization};
\]

end

end

4.3 Space group symmetry for MC-DNN applications

Out of the $n^L$ basis states possible on a crystal lattice some will be related to each other by operations of crystal symmetry. Then, the wave function values for these basis states are also related by symmetry. This suggests that crystal symmetry could be exploited to make neural network training, similar to Algorithm 8, more efficient. Conceptually, this is done by expanding $S$ to include all basis states that are related to those in $S$ by symmetry and ensuring that network training leverages the information provided by this expanded set.

To our knowledge no one has leveraged non-Abelian crystal symmetries to make methods similar to Algorithm 8 more efficient. A method to treat Abelian symmetries due to translation was introduced by [24]. Also, an approach to non-Abelian $\text{SU}(n)$ and $\text{O}(n)$ symmetries has been introduced [25]. The latter transforms individual spin basis to coupled basis of total angular momenta via the $3j$-symbols. While this approach is efficient for lattice spin models, it becomes cumbersome in solid state calculations when multiple spin-orbitals per site are occupied and it misses translational and rotational symmetries.
of crystals. Here we first provide a brief background into space groups and then discuss information more relevant to symmetrized MC-DNN applications.

4.3.1 Background

We start with the left coset decomposition of the space group $G$,

$$G = \{R_1|\tau_1\}T + ... + \{R_h|\tau_h\}T$$

(4.22)

where $R_j$ are point operations, $\tau_j$ are fractional translations, and $T = \sum_n \{E|t_n\}$ is the subgroup of lattice translations. $x(r)$, a configuration, is a function where each lattice site $r$ defines a distribution of electrons among atomic spin-orbitals on that site. Note that $x(r)$ are not the same thing as many-body basis states $|x\rangle$ which corresponds to a configuration by $x = \{x(r)\}_{r \in \mathbf{L}}$, where $\mathbf{L}$ is the complete set of lattice vectors. A space group operation, $\hat{g} = \{R|\tau + t\}$, acts on a configuration as

$$\hat{g}x(r) = x(\hat{g}^{-1}r) = x(R^{-1}(r - \tau - t)).$$

(4.23)

Here the transformed state, $\hat{g}x(r)$, is obtained by rotating and translating the initial. Note that we choose to manipulate the configuration (state) itself instead of the coordinate system. Transformation of our basis states occurs in the same way,

$$\hat{g}|x(r)\rangle = |x(\hat{g}^{-1}r)\rangle,$$

(4.24)

and a many-body state, $|\Psi\rangle = \sum_x \Psi(x)|x\rangle$, transforms as

$$\hat{g}|\Psi\rangle = \sum_x \Psi(x)\hat{g}|x\rangle = \sum_x \Psi(\hat{g}^{-1}x)|x\rangle.$$
This suggests that there are two ways to obtain $\hat{g} |\Psi\rangle$: transform the basis states with fixed coefficients, or transform the coefficients with fixed basis.

The Bloch theorem for a single particle state is written as

$$\{E|t\} \psi_k(r_0) = \psi_k(r_0 - t) = e^{-ikt} \psi_k(x) = \psi_k(\{E|t\}^{-1}x), \quad (4.26)$$

where for the last part we made a simplification using Eq. (4.25). The Bloch theorem holds for the many-body state,

$$\{E|t\} \Psi_k(\{x(r)\}_{r \in L}) = \Psi_k(\{x(r + t)\}_{r \in L}), \quad (4.27)$$

which demonstrates that representations of the lattice translations group $T$ are labeled by a wave vector $k$ from the first Brillouin zone (1BZ). Note that the many-body Bloch function, $\Psi_k$, is a function of a set of configurations on a lattice, $\{x(r)\}_{r \in L}$. An important symmetry property of $\Psi_k$ is that a symmetry element $\{S|w\}$ transforms a Bloch function at $k$ into another at wave vector $Sk$;

$$\{S|w\} \Psi_k(\{x(r)\}_{r \in L}) = \Psi_{Sk}(\{x(r + t)\}_{r \in L}), \quad (4.28)$$

as discussed in Section 3.6 of [47]. We can use this to help obtain a set of distinct wave vectors, obtained from $k_1 \in 1BZ$, by applying point group operations from the isogonal point group of $G$, denoted by $F$, which is a point group containing only point operations of $G$. We call this set the *star* of $k_1$ and define it as

$$^*k = \{k_1, \ldots, k_q\} = \bigcup_{R \in F} Rk_1. \quad (4.29)$$

In the union, $Rk_1 \equiv k_1$ if $Rk_1 = k_1 + g$, because two wave vectors that differ by a reciprocal lattice vector $g$ are considered equivalent. A subset of operations $S \in F$ satisfy
\( R k_1 \equiv k_1 \). We call this subset the \textit{little co-group} of \( k_1 \) and denote it by \( \bar{G}^{k_1} \). Every two vectors \((k_i, k_j \in \ast k)\) are related to one another via some operation \( Q \in F \) by \( k_j = Q k_i \).

In the case that \( i = 1 \) and \( j = 2 \) we can define the little co-group of \( k_2 \) to be

\[
\bar{G}^{k_2} = Q \bar{G}^{k_1} Q^{-1}.
\]  

(4.30)

All little co-groups of \( \ast k \) form a set of \textit{conjugate subgroups} of \( F \) [47]. This allows us to express \( F \) as a sum, or union, of left cosets with respect to \( \bar{G}^{k_1} \):

\[
F = \sum_{i=1}^{q} Q_i \bar{G}^{k_1},
\]  

(4.31)

where \( k_i = Q_i k_1 \) and \( Q_1 = E \). Keep in mind that \( \bar{G}^{k_1} \) is a set of symmetry operations, and each \( Q_i \) will transform this set. We can finally relate what we have discussed so far to space groups. We introduce the \textit{little group} \( G^{k_1} \) which is a space group and subgroup of \( G \) in Eq. (4.22). It is formed by left cosets with the rotational operations, \( S_i' \), from the \textit{little co-group} \( \bar{G}^{k_1} \):

\[
G^{k_1} = \sum_{i=1}^{b} \{ S_i' | \tau_i \} T
\]  

(4.32)

with \( S_i' = E \) and \( \tau_1 = 0 \). \( G^{k_1} \) is the space group containing all symmetry operations of the wave vector \( k_1 \). From here we can define the full space group as a sum, or union, of left cosets with the little groups of \( \ast k \):

\[
G = \sum_{i=1}^{q} \{ Q_i | \zeta_i \} G^{k_i}.
\]  

(4.33)

Important to the application of symmetrized networks, Bloch function symmetries are classified according to the \textit{small representations} of the little group \( G^{k_1} \). The basis functions of small representations must also satisfy the Bloch function property described in

70
Eq. (4.27). In general, the matrices of a small representation are defined as

$$\Gamma^k_\beta(\{R|\mathbf{v}\}) = e^{-ik\cdot\mathbf{v}}D^k_\beta(R),$$  \hspace{1cm} (4.34)

where $\mathbf{v} = \tau + t$ and $D^k_\beta(R)$ are the same for each coset of Eq. (4.33). $D^k_\beta(R)$ can be computed using the method of projective representations [47]. We define the basis functions for the small representation $\beta$ of $G^{k_1}$ to be

$$\left\{ \psi^\beta_{k_1,i}(x) \right\}_{i=1}^r,$$  \hspace{1cm} (4.35)

where $r$ is the dimensionality of $\beta$. An irreducible representation of the space group $G$ can be defined by forming a linear closure of the vector space spanned by the functions

$$\{Q_j|\zeta_j\}\psi^\beta_{k_1,j}(x)$$  \hspace{1cm} (4.36)

where $j = 1, ..., q$ and $i = 1, ..., r$ and $\{Q_j|\zeta_j\}$ are the left coset representatives in Eq. (4.33). The full dimension of the representation is $d = qr$.

### 4.3.2 Implementing space group symmetry

In practice we study a physical model by simulating it on a supercell with lattice vectors $A_i$, which are combinations of primitive lattice vectors, $a_j$;

$$A_i = \sum_j M_{ij}a_j,$$  \hspace{1cm} (4.37)

where $M$ is a matrix of integers. A supercell has $N_S = |M|$ lattice sites and we define $N_S$ lattice vectors, $t_n$, that point to each lattice site from the origin, 0. By imposing periodic boundary conditions (PBCs) we can force $t_n$ to become the finite group $T_S$. 

71
Figure 4.1: A simple illustration of orbits. Each dot represents a basis state. Applying $G_S$ to any basis state $(x_i, x_j)$ of the orbit, say $x_j$, yields the same orbit $O(x)$. $x_E$ is a reference configuration (canonical) which should be the same regardless of the starting basis state $(x_i, x_j)$. Red dots, $g'_m x_E$, represent an invertible subspace of the full orbit.

The full symmetry group of the supercell, $G_S$, consists of a set of symmetry operations, $g_n = \{R|\tau + t_n\}$, that transform supercell sites into themselves for $t_n \in T_S$.

Given normalized and unsymmetrized basis states, $x$, we define an orbit as

$$O(x) = \bigcup_{g_n \in G_S} \tilde{g}_n x,$$  \hspace{1cm} (4.38)

which creates a set of symmetry related basis states; in fact, the entire set of basis states, $S_T$, can be divided into non-overlapping orbits as demonstrated by Figure 4.1. This means that applying $G_S$ to any $x_n \in O$ will obtain the same orbit.
In order to take advantage of orbits we must be able to differentiate between them. To do this we must choose a reference basis state in each orbit, called a "canonical" basis state \( x_E \), which can be defined arbitrarily but must be reproducible. Practically, \( O \) is a set of symmetry transformed basis states, each being an array of integers. From this viewpoint a simple way of defining \( x_E \) is to sort \( O \) and then take the array at index-\( n \) as \( x_E \). Any index will work so long as it is the same for all calculations of \( x_E \). We can also use \( x_E \) to notate the other elements of \( O \) by the symmetry operation needed to transform \( x_E \) into each element;

\[
x_{g_n} = \hat{g}_n x_E.
\]

An important property of the orbit is that \( \Psi(x') \) and \( \Psi(x'') \) are related for any \( x', x'' \in O \), which reduces the number of unique wave function values by a factor of \( |O| \). In theory, this may allow for network training to become easier and require less trainable parameters to approximate the many-body wave function. To take advantage of the symmetrical properties of the wave function we first obtain the normalized and symmetrized basis functions of the orbit,

\[
| \phi_{m \ell}^j(x_E) \rangle = \sqrt{\frac{d_j}{|G_S|}} \sum_{g \in G_S} \Gamma_j^g (g)^*_{m \ell} | x_g \rangle,
\]

where \( j \) is a collective index for \( k \)-point \( k \) and irrep \( \beta \), \( d \) in the dimensions of the irrep, \( m \) runs over the basis functions of irrep, \( \Gamma \) is the irrep matrix defined in Eq. (4.34), and \( \ell \) labels the columns of the irrep matrices. We have dropped the index on symmetry operations for simplicity. Here each element of the orbit is weighted by the irrep matrix element of \( G_S \) and summed forming a reduced representation of the orbit.

Because our end goal is to calculate the symmetrized wave function simply calculating Eq. (4.40) will not help us because we must use our network to perform the mapping and it can only accept unsymmetrized basis states as input. We can define the wave function
using the normalized and symmetrized basis states as

\[ |\Psi^j_m\rangle = \sum_{x_E,t} \Psi^j(x_E,t) |\phi^j_{mt}(x_E)\rangle, \]  

(4.41)

Our goal is to include our neural network output into this equation by first substituting Eq. (4.40) into Eq. (4.41) resulting in

\[ |\Psi^j\rangle = \sum_{x_E,t} \Psi^j(x_E,t) \sqrt{\frac{d_j}{|G_S|}} \sum_{g \in G_S} \Gamma^j(g)_{1t}^* |x_g\rangle \]

(4.42)

\[ = \sum_{x_E,g \in G_S} \left( \sum_{t} \Psi^j(x_E,t) \sqrt{\frac{d_j}{|G_S|}} \Gamma^j(g)_{1t}^* \right) |x_g\rangle, \]

where we used only \( m = 1 \). Furthermore, the expression in the parenthesis can be identified as the network representation of the wave function

\[ \Psi^{\text{net}}(x_g) = \sum_{t} \Psi^j(x_E,t) \sqrt{\frac{d_j}{|G_S|}} \Gamma^j(g)_{1t}^*, \]  

(4.43)

and the coefficients for the normalized and symmetrized basis states can be obtained through inversion,

\[ \Psi^j(x_E,t) = \sum_{g' \in G'_S} \left[ M^j \right]^{-1}_{g't} \Psi^{\text{net}}(g'x_E). \]  

(4.44)

where the subset \( G'_S \subset G_S \) is selected so that

\[ M^j_{g't} = \sqrt{\frac{d_j}{|G_S|}} \Gamma^j(g'_{1t}). \]  

(4.45)

is invertible.

This shows that the wave function is now parameterized by \( j \) (k-point and irrep) and the canonical basis \( x_E \), demonstrating a significant reduction of the unsymmetrized wave function.

74
Having defined the symmetrized wave function we now define the symmetrized expressions for the average energy, gradient, and loss function needed to training the weights of the network. The normalized weight

$$w^j(x_E, t) = \frac{|\Psi^j(x_E, t)|^2}{\sum_{(x'_E, t') \in S_{E \neq}} |\Psi^j(x'_E, t')|^2} \quad (4.46)$$

approximates the probability of a particular \((x_E, t)\) basis state. Here we define \(S_{E \neq}\) as the unique canonical basis states of \(S\) similar to what was done in section 4.2.2. We define the local energy as

$$E^j(x_E, t) = \left( \frac{1}{\Psi^j(x_E, t)} \right)^{-1} \sum_{y_E} \sum_{s=1}^{n^j_{y_E}} H^j_{ts}(x_E, y_E) \Psi^j(y_E, s), \quad (4.47)$$

where the symmetrized Hamiltonian \(H^j_{ts}(x_E, y_E)\) and states \(y_E\) are described in detail in Section 4.3.2. Both Eq.’s (4.46) and (4.47) can then be used to express the symmetrized average energy as

$$E^j = \sum_{(x_E, t) \in S_{E \neq}} w^j(x_E, t) E^j(x_E, t). \quad (4.48)$$

Following our previous discussions, the gradient of the average energy with respect to the weights of the network is defined by

$$\partial_w E^j = 2 \sum_{(x_E, t) \in S_{E \neq}} w^j(x_E, t) \text{Re} \left[ \left( E^j(x_E, t) - E^j \right) \Psi^j(x_E, t) \partial_w \Psi^j(x_E, t)^\ast \right] \quad (4.49)$$

and the symmetrized gradient trick, where we pre-calculate all constants ahead of time, is defined by

$$g^j(x_E, t) = 2w^j(x_E, t) \text{Re} \left[ \left( E^j(x_E, t) - E^j \right) \Psi^j(x_E, t) \right]. \quad (4.50)$$

Using Eq. (4.50) we can now define the loss function used to guide our network weights.
to the ground state:

\[
L^j = \sum_{(x_E,t) \in S_E, \phi} g^j(x_E, t)\Psi^j(x_E, t)^*.
\] (4.51)

Notice that this loss differs from previous definitions primarily due to the inclusion of the index \( j \). Our network will now find the lowest energy state for a specific \( k \)-point and irrep, and one needs to run over all \( k \) in the irreducible Brillouin zone (IBZ) and all irrep indexes. Often this is not a complex problem seeing as the ground state usually occurs at a high-symmetry \( k \)-point and irrep.

**Symmetrized Hamiltonian**

We need to calculate the Hamiltonian matrix in the symmetrized basis, \( H^j_{ts}(x_E, y_E) = \langle x_E | \hat{H} | \phi^j_{ts}(y_E) \rangle \), so that we can calculate the local energy in Eq. (4.47). Consider a general Hamiltonian matrix element,

\[
\langle \phi^j_{mt}(x_E) | \hat{H} | \phi^j_{ns}(y_E) \rangle = \sqrt{\frac{d_id_j}{|G_S|^2}} \sum_{g, h} \Gamma^j(g)_{mt} \Gamma^i(h)^*_{ns} \langle \hat{g}x_E | \hat{H} | \hat{h}y_E \rangle
\]

\[
\quad \quad = \sqrt{\frac{d_id_j}{|G_S|^2}} \sum_{g, h} \Gamma^j(g)_{mt} \Gamma^i(h)^*_{ns} \langle x_E | \hat{H} | \hat{h}y_E \rangle.
\] (4.52)

We can expand \( \Gamma^i(gh)^*_{ns} \) into a matrix product of \( \Gamma^i(g)^* \) and \( \Gamma^i(h)^* \) by including an additional sum over an intermediate index \( p \) such that our expression becomes

\[
\langle \phi^j_{mt}(x_E) | \hat{H} | \phi^j_{ns}(y_E) \rangle = \sqrt{\frac{d_id_j}{|G_S|^2}} \sum_p \sum_{g, h} \Gamma^j(g)_{mt} \Gamma^i(g)^*_{np} \Gamma^i(h)^*_{ps} \langle x_E | \hat{H} | \hat{h}y_E \rangle.
\] (4.53)
This means we can further reduce our expression to

\[
\langle \phi^j_{mt}(x_E) | \hat{H} | \phi^j_{ns}(y_E) \rangle = \delta^{ij} \delta_{mn} \sum_h \Gamma^i(h)^*_t \langle x_E | \hat{H} | \hat{y}_{E} \rangle
\]

\[
= \sqrt{\frac{|G_S|}{d_i}} \delta^{ij} \delta_{mn} \langle x_E | \hat{H} | \phi^j_{ts}(y_E) \rangle.
\]

(4.54)

The result is that the Hamiltonian becomes block-diagonal with each block indexed by \( j \equiv (\ast k, \beta) \) and irrep row indicies \( m \). Each block matrix elements are indexed by orbit states \( x_E, y_E \) and irrep basis sets \( s, t \). The steps for calculating the symmetrized Hamiltonian and states \( y_E \) are summarized in Algorithm 9.

---

**Algorithm 9: Symmetrized Hamiltonian**

Input: \( x_E \) (canonical basis state), \( t \) (irrep matrix column);

1: Find all \( y \) where \( \langle x_E | \hat{H} | y \rangle \neq 0 \);

2: Determine \( y_E \) for all \( y \) and set \( H_{ts}(x_E, y_E) = 0 \);

3: Calculate the normalization factors for \( z = x_E, y_E \):

\[
N_z = \sqrt{\sum_{g \in G_S} \Gamma^j(g)^*_t}.
\]

4: for each \( y \) do

   a): Find all \( \hat{h} \) such that \( y = \hat{h} y_E \);

   \[
   \langle x_E | \hat{H} | \hat{y}_{E} \rangle = \langle x_E | \hat{H} | y \rangle;
   \]

   b): Update the matrix elements;

\[
H^j_{ts}(x_E, y_E) \rightarrow \frac{1}{N_{x_E}N_{y_E}} \left( \sqrt{\frac{d_j}{|G_S|}} \sum_{h \in \{h|H_{E} = y\}} \Gamma^j(h)^*_t \right) \langle x_E | \hat{H} | y \rangle;
\]

end

Output: All \( y_E \) and \( H^j_{ts} \).
4.4 Symmetrized MC

Algorithm 10 demonstrates how we calculate symmetrized MC. Here we still sample $S$, which is a set of basis states. However, because the acceptance criterion relies on calculating the probability, for each $x \in S$ we must calculate $x_E^m \in \mathcal{G}'_S$, and $M^j$ as indicated by Eq. (4.44). Furthermore, because every state within an orbit has the same canonical basis state, $|S_{E\neq}| \leq |S|$. When it is possible, it is advantageous to store important calculations, such as $\mathcal{G}'_S$ and $M^j$, for each $x_E$ which will reduce redundant calculations in the future. Note that lattice size is directly related to the total number of possible basis states $S_T$. Thus, when the lattice is small, sometimes a single sampling results in $S \approx S_T$. In this case storing symmetry calculations will dramatically speed up later algorithms. However, for larger lattice sizes, storing calculations for each $x_E \in S_{E \neq}$ is unrealistic because there are an enormous amount of canonical basis states possible. Fortunately, as the network improves the sampling will gravitate to only the highest-probability basis states rejecting insignificant ones. It is common for the high-probability basis states to form a small subset of $S_T$ which makes storing calculations more realistic.

An important difference in symmetrized MC is the change to the acceptance criterion. Here probability must be scaled by the dimension of the normalized symmetrized basis (4.40), $N_\beta$, and the dimension of the orbit $N_\mathcal{O}$. 

78
Algorithm 10: Symmetrized Monte Carlo (MC) based on Algorithm 7. Here $N_\beta$ is the dimension of the normalized symmetrized basis in Eq. (4.40) and $\Psi(x_E)$ is defined by Eq. (4.44). We have dropped the $j$ notation (k-point and irrep index) for simplicity.

**Input:** $N_{\text{sites}}, N_\uparrow, N_\downarrow, N_{\text{sample}}$;

**Initialize:** $S_{E,\neq} = \emptyset, x = \text{GenerateState}(N_{\text{sites}}, N_\uparrow, N_\downarrow)$;

$\{x_E, G'_S, M\} \leftarrow \text{Section (4.3.2)}$;

for $i = 1 \ldots N_{\text{sample}}$ do

\begin{align*}
    x_{\text{trial}} &= \text{GenerateTrial}(x); \\
    \{x_E, G'_S, M\}_{\text{trial}} &\leftarrow \text{Section (4.3.2)}; \\
    r &= \text{Random}(0, 1); \\
    \text{if } r \frac{N_\beta}{N_{\Omega(x)}} |\Psi(x_E)|^2 \leq \frac{N_{\text{trial}}}{N_{\Omega(x_{\text{trial}})}} |\Psi(x_{E_{\text{trial}}})|^2 \text{ then} \\
    &\quad x_E = x_{E_{\text{trial}}} \\
    \text{end}
\end{align*}

$S_{E,\neq} = S_{E,\neq} \cup \{x_E\}$;

end

4.5 Symmetrized multi-use MC and DNN training

Finally, we can put everything together and define our approach for performing symmetrized multi-use MC and DNN training. In Algorithm 11 we show how symmetrized MC (Algorithm 10) and the symmetrized gradient trick (Eq. (4.50)) can be used to train a DNN for approximating the ground state wave function. In addition, Figure 4.2 discusses in more detail the steps that are needed to perform Algorithm 10.
Algorithm 11: Neural network optimization via multi-use symmetrized MC. Here we have omitted the irrep matrix column index \( t \) for simplicity. See Figure 4.2 for elaboration.

Input: \( N_{MC}, N_{opt} \);

Initialize: \( \Psi \) network with random uniform weights using Eq. (4.51) as loss;

for \( i = 1 \ldots N_{MC} \) do

\( S_{E_{\phi}} \leftarrow \) Symmetrized MC from Algorithm 10;

for \( j = 1 \ldots N_{opt} \) do

\( g(x_E \in S_{E_{\phi}}) \leftarrow \) Eq. (4.50);

\( \Psi \leftarrow \) Backpropagation and loss minimization;

end

end
Figure 4.2: Diagram illustrating our process for performing symmetrized multi-use MC and DNN training. The outer and inner loop are notated by the black and red arrows and mirror the for loops in Algorithm 11. Here we elaborate on the specific expressions needed at each stage of our process.

\[
\Psi(x, t) = \sum_{m=1}^{n_{st}} [M]^{-1}_{mt} \Psi^{\text{net}}(\hat{g}_m x, t)
\]

Normalized Weight
\[
w^i(x, t) = \frac{|\Psi(x, t)|^2}{\sum_{i,j \in S_{E, \pi}} |\Psi(x', t')|^2}
\]

Local Energy
\[
E^i(x, t) = \sum_{y_s} \sum_{s=1}^{n_s} \frac{H^i(x, y_s) \Psi(y, s)}{\Psi(x, t)}
\]

Average Energy
\[
E^i = \sum_{i,j \in S_{E, \pi}} w^i(x, t) E^i(x, t)
\]

Gradient Trick
\[
g^i(x, t) = 2w^i(x, t) \text{Re} \left[ (E^i(x, t) - E^i) \Psi(x, t) \right]
\]

Symmetrized Loss
\[
L^i = \sum_{i,j \in S_{E, \pi}} g^i(x, t) \Psi(x, t)^* \Psi(x, t)
\]
4.6 DNN architecture

Figure 4.3 describes a symmetrized complex neural network. Object dimensions and elements are notated using parenthesis and brackets respectively. The input to our network is $\hat{g}^m_{x_E}$, a single inverted basis function described in Section 4.3.2. Regardless of the dimension of the Hamiltonian, the input always has two dimensions, the total sites $N_{\text{sites}}$ and number of particle states $N_{\text{in}}$ which is implemented as one-hot. We describe a deep neural network consisting of $N_l = 4$ convolutional layers $C$, an aggregation layer which sums about the spatial dimension of the previous layer, and a single dense layer $D$. Each $C$ has a kernel size $(K_x, K_y)$ and $F$ filter channels. The output of the network are two values; amplitude $A$ and phase $\phi$. We can then use these values to formulate the complex wave function as $\Psi_{\text{net}}(\hat{g}^m_{x_E}) = [A \cos \phi, A \sin \phi]$ (see Eq. (4.44)).

Periodic boundary conditions (PBCs) are enforced by wrapping enough spatial information so that the convolutional filters can reach beyond the boundaries. Practically we enforce this by concatenating $K/2$ elements to each layer input to simulate PBCs.
Figure 4.3: A diagram describing a symmetrized complex neural network. Object dimensions and elements are notated using parenthesis and brackets respectively. As indicated by Eq. (4.44), our input is a single inverted basis function (See Section 4.3.2) and output the complex wave function $\Psi^{\text{net}}(\hat{g_{m}^{t}x_{E}})$. Here the actual output of the network, $A$ and $\phi$, are amplitude and phase respectively. Convolutions and dense layers are defined by $C$ and $D$ respectively. The green objects (widest) represent the feature maps of the convolutional layers. The sum aggregates information along the spatial indices. Periodic boundary conditions (PBCs) are enforced by wrapping enough spatial information so that the convolutional filters can reach beyond the boundaries.
4.7 Spin model

We investigate a simple spin model where each lattice site can have either a spin up or spin down particle. Basis states for this model are notated using Eq. (4.1) but for $s_i = -1/2, 1/2$ (SU(2)). Our Hamiltonian consists of a set of nearest-neighbor exchange operators. In two-dimensions (2D) we formulate it as

$$\hat{H} = \sum_{<i,j>} \hat{P}_{i,j},$$

(4.55)

where $<i,j>$ represents all distinct 2D nearest-neighbor pairs. Eq. (4.1) is related to 2D lattice positions such that $n_i \rightarrow (x_i, y_i)$. For example, if we have a $4 \times 4$ lattice $n_1 \rightarrow (1,1)$ while $n_5 \rightarrow (2,1)$. Algorithm 9 demonstrates how to treat this Hamiltonian within the scope of symmetrization.

4.7.1 Numerical setup

In all of our numerical results we implement Algorithm 11 and use the following parameter values: a lattice size of $4 \times 4$, SU(2), $N_{\text{samples}} = 10000$, $N_{\text{MC}} = 100$, $N_{\text{opt}} = 20$, a convolutional layer kernel size of $K = 3$, and Adam optimization [48] with out-of-box settings from Tensorflow [45]. For more complex systems, due to lattice size or number of spin states, $N_{\text{samples}}$ may need to be increased so that the averages are meaningful. We do not notice a sensitivity between algorithm initialization (network and MC) and average energy.

4.7.2 Numerical results for different k-point and irreps

Although we can use symmetry to reduce sampling and as a result reduce the unique training data needed for network training, we still need to explore possible k-points and
Figure 4.4: Average energy (Eq. (4.48)) shown for different $j = (k, \beta - \text{index})$ notated as $k, \beta - \text{index}$. Both real (a) and imaginary (b) components are possible because our network is complex (See Figure 4.3. Setup includes $N_l = 6$ convolutional layers, $F = 64$ filter channels per layer, and a learning rate of $\gamma = 0.0005$. 

85
irrep indexes \((j)\) for the ground state. Figure 4.4 demonstrates average energy \((\text{Eq. (4.48)})\) for the spin model defined by Eq. (4.55) at different \(j = (k, \beta)\), notated as \(k_{\beta-\text{index}}\). Our setup includes a network architecture as described in Figure 4.3 but with \(N_l = 6\) convolutional layers, \(F = 64\) filter channels per layer, and a learning rate of \(\gamma = 0.0005\).

Because our network is complex and the sampling is not perfect, the total energy can have a non-zero imaginary part. We notice that the non-physical imaginary components become negligible when network training has converged. This is demonstrated by the difference between the real and imaginary plots for \(M_1\) and \(\Gamma_4\) (fully symmetric), where the former has noisy \(\langle E_{\text{Im}} \rangle\) and decreasing \(\langle E_{\text{Re}} \rangle\) at iteration 2000 and the later has negligible \(\langle E_{\text{Im}} \rangle\) and a converged \(\langle E_{\text{Re}} \rangle\). As evidenced by the lowest energy plot in (a), \(\Gamma_4\) corresponds to the ground state with \(\langle E_{\text{Re}} \rangle = -6.456\). Direct diagonalization shows that the exact energy for this case is \(-6.457\), which means our trained network is 99.98% accurate. We expect further accuracy could be obtained with proper tuning of network and algorithm parameters.

After 2000 iterations (weight updates) our network has converged as evidenced by the constant flat values near \(\langle E^{(2000)}_{\text{Re}} \rangle\) and very small gradient norm \(\|G^{(2000)}\|^2\) (Figure 4.5 (a)) (the gradient of our network is given by Eq. (4.49)). Although the norm of the gradient is hard to interpret due to the complexity of the network mapping and corresponding manifold, the fact that the norm significantly decreases for all plots near iteration 2000 suggests that our algorithm is stable with respect to \(j\).

The impact of symmetry is illustrated in Figure 4.5 (b), where the size of the sampled set of symmetrized states from Algorithm 10, \(|S_{E^\pi}|\), is significantly smaller than the number of sampled basis states, \(N_{\text{samples}} = 10000\), for each \(j\). Note that the choice of the irrep \(j\) affects the number of symmetry-unique sampled basis states. Interestingly, for each \(j\), \(|S_{E^\pi}|\) decreases with each iteration suggesting that a trained network will help the sampling algorithm to discover a subset of high-probability basis states necessary for obtaining
the lowest energy state while discarding insignificant basis states. Note that \(|S_{E,\phi}|\) is most noisy when the network is untrained. The highest probability \(x_E \in S_{E,\phi}\) correspond to the checkerboard configuration or near checkerboard which is expected for the ground state of the SU(2) spin model.

### 4.7.3 Numerical results for different networks

Network size affects the complexity of the mapping between basis states and wave function values. We now explore how network size affects the accuracy of our method. Our setup includes \(j = \Gamma_5\) (doubly degenerate), and a learning rate of \(\gamma = 0.001\). Figure 4.6 plots average energy vs network training iteration for the network architecture described in Figure 4.3 but for \(N_l\) and \(F\) values shown in the legends. In this case, direct diagonalization yields an exact average energy of 1.1348. When \(N_l = 1\) and \(F = 1\), our network has only one convolutional weight and two dense weights. Notice that while this small network incurs large error with an average energy plateau near \(\langle E_{Re}\rangle = 11\), training still has a great effect on improving the mapping. We notice a significant improvement of model accuracy for a single layer by increasing the number of filter channels \((F = 64)\). Increasing the number of layers is also shown to lower the average energy with \(N_l = 6\) and \(F = 64\) producing \(\langle E_{Re}\rangle = 1.1395\) after 2000 weights updates which leads to 99.6% accuracy. We believe with careful choice of network and algorithm parameters that the accuracy can be further improved.

After 2000 iterations (weight updates) our network has converged as evidenced by the constant flat values near \(\langle E_{Re}^{(2000)}\rangle\) and the negligible values for \(\langle E_{Im}^{(2000)}\rangle\) and \(\|G^{(2000)}\|^2\) (Figures 4.6 (b) and 4.7 (a), respectively). While hard to interpret, \(\|G\|^2\) is noticeably larger for the smaller models which may be attributed to an insufficient number of trainable parameters for capturing the complexity of the problem. We suspect that in this case the weights have a tendency to change more drastically to adapt to new inputs.
Figure 4.5: Norm of the gradient (Eq. (4.49)) and size of symmetrized MC sampling (Algorithm 10) is shown in subplots (a) and (b) respectively for different $j = (k, \beta - \text{index})$ notated as $k_{\beta - \text{index}}$. Setup described in Figure 4.4.
We notice that for all cases besides $N_\ell = 1$ and $F = 64$ network training decreases the size of sampling and plateaus near $|S| = 110$ (Figure 4.7 (b)). For our smallest model, the size of our sampling varies wildly even near iteration 2000 and results in much smaller sample sizes.

### 4.7.4 Numerical results for different learning rates

The learning rate affects the step-size of a gradient based optimization algorithm. If the step size if too large the optimum network can be overshot whereas having too small of a learning rate can slow down optimization requiring significantly more iterations to achieve a desired accuracy. In Figures 4.8 and 4.9 we demonstrate results for our spin model with a setup of $j = \Gamma_4$ (fully symmetric), $N_\ell = 6$, and $F = 64$. Figure 4.8 plots average energy vs network training iteration for three learning rates. For each learning rate we use different network initializations and start with a different basis state for MC which is why $\langle E_{Re}\rangle$ differs at $i = 0$. We show that regardless of initialization, the average energy behaves as expected with respect to the learning rate. Inspecting plot (a) reveals that the largest learning rate ($\gamma = 0.001$) reaches an $\langle E_{Re}\rangle$ plateau first with the smallest learning rate ($\gamma = 0.0001$) not reaching plateau even after 2000 training iterations as evidenced by the zoomed in subplot inside (a) and the noise at the end of the green plots in (b) and Figure 4.9 (a). At $\langle E_{Re}^{(2000)}\rangle$ we have $-6.450$ and $-6.456$ for $\gamma = 0.001$ and $\gamma = 0.0005$ respectively. This demonstrates that $\gamma = 0.001$ may be slightly too large to reach the optimum. For $\gamma = 0.0005$ we obtain an average energy accuracy of 99.98% as discussed previously.

Our different starting conditions also affect the initial size of $|S|$ as evidenced by Figure 4.9 (b). For $\gamma = 0.001$ and $\gamma = 0.0005$ network training results in similar sampling sizes at iteration 2000 while the unconverged $\gamma = 0.0001$ may still reduce sampling size with more training.
Figure 4.6: Average energy (Eq. (4.48)) shown for different networks parameterized by number of convolutional layers $N_l$ and number of filter channels per layer $F$. Both real (a) and imaginary (b) components are possible because our network is complex (See Figure 4.3. Setup includes k-point-irrep collective index $j = \Gamma_5$, and a learning rate of $\gamma = 0.0005$. 

90
Figure 4.7: Norm of the gradient (Eq. (4.49)) and size of symmetrized MC sampling (Algorithm 10) is shown in subplots (a) and (b) respectively for different networks parameterized by number of convolutional layers $N_l$ and number of filter channels per layer $F$. Setup described in Figure 4.6.
Figure 4.8: Average energy (Eq. (4.48)) shown for different learning rates $\gamma$. Both real (a) and imaginary (b) components are possible because our network is complex (See Figure 4.3. Setup includes $\mathbf{k}$-point-irrep collective index $j = \Gamma_4$, number of convolutional layers $N_l = 6$, and filter channels per layer $F = 64$.}
Figure 4.9: Norm of the gradient (Eq. (4.49)) and size of symmetrized MC sampling (Algorithm 10) is shown in subplots (a) and (b) respectively for different learning rates \( \gamma \). Setup described in Figure 4.8.
Chapter 5

Discussion

5.1 Calculating Wannier functions using dictionary learning

In chapter 2 we considered the problem of finding exponentially localized solutions to a second order eigenvalue PDE with gauge symmetry. Our approach is designed for problems in low spatial dimensions that require highly accurate solutions. We demonstrate that dictionary learning can be used to extract general features from a pre-calculated database of Wannier functions, and that dictionary learning and basis pursuit can leverage past information (general features) for finding Wannier functions in new situations. Our work outlines a path towards obtaining a general dictionary that contains essential features and can be used to calculate and possibly classify Wannier functions in real materials.

Our proposed numerical algorithm is computationally efficient due to its compatibility with gradient based minimization methods on the Stiefel manifold. An important aspect of our approach is that the accuracy of our solutions is controlled via a single parameter $\mu$, which represents a trade-off between localization and energy minimization. The ability to achieve both localized and highly accurate solutions relies on the generality and
representational power of the derived dictionary.

The outlined approach admits several extensions. First, computational efficiency can be increased for systems with spatial symmetry. For the case of Wannier functions, this can be done by restricting the $k$ values in Eq. (2.13) to the irreducible wedge of the Brillouin zone and obtaining Wannier function Fourier components $\tilde{X}_{[k]}$ in the entire zone using induced representation theory [32]. This straightforward development will increase both the speed of the calculations and physical interpretability of the results.

The second line of future work involves calculating changes to the Wannier functions associated with small perturbations within the linear response theory. Suppose that the Hamiltonian is a function of a parameter, $H = H(\lambda)$. The parameter $\lambda$ may represent the amplitude of an atomic displacement pattern in a phonon mode, an external electrical field, or change in electronic occupations [49, 50]. It is then of interest to calculate the first-order derivative of the wave functions which, according the the “2n+1” theorem of perturbation theory [51], completely determine the response functions given by the 2nd and 3rd-order derivatives of the total energies. To see how this can be accomplished, we write down the Euler-Lagrange equation for the minimization problem in Eq. (2.11):

\[
(H + \mu - \epsilon) X = \mu D\alpha. 
\]  
(5.1)

where $\epsilon$ is a diagonal matrix of Lagrange multipliers associated with the orthonormality constraint. We then take the derivative w.r.t. $\lambda$ of both sides of Eq. (5.1):

\[
(H + \mu - \epsilon) \frac{dX}{d\lambda} = \mu \frac{d}{d\lambda} (D\alpha) - \left( \frac{dH}{d\lambda} - \frac{d\epsilon}{d\lambda} \right) X 
\]  
(5.2)

This is a linear equation for $dX/d\lambda$ which, together with the orthogonality condition

\[
\frac{dX^\dagger}{d\lambda} X + X^\dagger \frac{dX}{d\lambda} = 0 
\]  
(5.3)
can be solved with standard iterative techniques already used in DFT linear response codes. An open question concerns the best way to treat \( \frac{\partial \alpha}{\partial \lambda} \), which according to Eq. (2.12) involves a subgradient. Calculation of the linear response directly in terms of the Wannier functions should be more reliable than and comparably efficient to the post-DFT Wannierization employed currently.

More generally, our dictionary learning approach could be of use to work related to Refs. [52, 53] when solutions with localized or specifically structured spatial characteristics are desired. While we have only showed results for a single type of PDE we believe our approach would extend to the PDEs with compressed solutions that are studied in [54].

It may be interesting to study what other machine learning techniques can be applied to extract meaningful features from a Wannier database and then enforce those features on \( W \). We recommend two possible avenues; 1) using a different technique for Wannier feature extraction while still using our algorithm for solving Eq. 1.2, 2) using a different technique for Wannier feature extraction and a different objective function term to enforce learned features on \( W \). Neural networks and auto-encoders may be interesting techniques to explore.

Finally, the calculation of Wannier functions without explicit diagonalization of the Hamiltonian is attractive when the former becomes computationally expensive, such as in many beyond-DFT methods [9]. The approach presented here can be incorporated in the self-consistency cycle, and the localized character of Wannier functions can be used to accelerate the calculation of exchange and correlation contributions.
5.2 Calculating Wannier functions using adaptive dictionary learning

In chapter 3 we proposed a new Wannier method and provided 3D real material simulations demonstrating its effectiveness. From a data science perspective, our work can be viewed as a "bootstrapped" approach for obtaining a Wannier database and general Wannier dictionary. If we had this database, our static dictionary Wannier approach described in Chapter 2 would be more efficient and practical to use. Furthermore, because our adaptive dictionary approach only relies on our choice of \( \mu \) with no other sensitive algorithmic parameters, it also requires less manual involvement.

An essential benefit our approach is the compatibility with constrained symmetry. Having symmetrical exponentially localized Wannier functions ensures their values are related at symmetrically equivalent \( k \)-points which allows numerical implementations to exploit this simplification for maximum efficiency.

Establishing a stopping criteria for our \( \mu \)-schedule would make our approach fully automated. Future work on not only an efficient implementation of our proposed algorithm but also the determination of a stopping criteria for \( \mu \) would be worth pursuing. Such an approach would not only allow for Wannier functions to be easily calculated "on the fly", but also for a Wannier database to be easily obtained. The learned dictionary corresponding to this database would serve as a Wannier basis, which may have a deeper physical meaning that is useful for describing physical phenomena in materials.

Having a general dictionary for any physical system requires interpolating information on different lattices which is a costly procedure. However, this process is highly parallelizable and the cost can be reduced with proper numerical implementation.

In the future it may be interesting to explore other methods for solving Eq. (1.5). One
idea worth pursuing is alternating between performing optimization with respect to \( W \) and \( \alpha \) until convergence (the method described in chapter 2), and optimizing over \( D \) and \( \alpha \) until convergence (dictionary learning). Doing so may increase the convergence of Figure 3.5 and allow for better transfer of learned features from \( D \).

### 5.3 Deep learning for quantum many-body physics

In chapter 4 we demonstrated that space group symmetry can be applied to an approach that alternates between MC and neural network training to find the ground state of a many-body system. Here MC samples basis states of a many-body system and the network learns a inexpensive mapping between basis states and wave function values such that Eq. (4.48), the average energy, is at a minimum.

Our numerical results demonstrate that when performing \( 10000 \) sampling steps in symmetrized MC (Algorithm 10) the number of symmetrically unique sampled basis states is on the order of \( 100 \) for \( \Gamma_4 \) (fully symmetric irrep). Here, redundancy occurs due to space group symmetry, where basis states are gathered into non-overlapping groups called orbits. However, we must also account for our use of an invertible subspace of each orbit in Eq. (4.44). Thus, each symmetrized MC chain element, a basis state, points to an invertible subspace which increases the size of the training data that is pushed through the network. In our numerical results the maximum size for an invertible subspace is 2 which means the number of elements that we push through our network is \( \sim 200 \). In comparison, for non-symmetry based alternating approaches, such as [22], there is a one-to-one correspondence between the number of sampled states and the size of the network training data. Thus for our numerical results, we demonstrate \( \sim 50 \times \) reduction in training data size. This fits well with theory that states the reduction of \( S_T \) should be a factor \( N_l \times |G_S|/|M| \), and in our case for \( N_l = 16, |G_S| = 8, \) and \( |M| = 2 \) we should expect a reduction of \( 64 \times \). A
reduction in training data could allow for the use of smaller models while still obtaining accurate mappings for the ground state.

The size of the orbit is defined by $|\mathcal{O}| = L|G^j_S|$, where $L$ is the number of lattice sites and $|G^j_S|$ is the number of group operations at k-irrep-index $j$. In theory the size of the orbit will scale linearly with lattice size which suggests that our approach should work for larger more complex models.

While our symmetrized approach reduces the size of training data there is a cost associated with manipulating the data into this form. However, obtaining symmetry information needed for network training is highly parallelizable which upon implementation would make either MC sampling or network training the algorithmic bottleneck.

We have documented that our approach is capable of obtaining a 99.98% accurate approximation of the ground state wave function (see section 4.7.2) with an error per lattice site

$$\frac{\Delta E}{(C \ast N_l)}$$

of $6.25 \times 10^{-5}$ for $\Delta E = 0.001$, an exchange coupling of $C = 1$, and $N_l = 16$ lattice sites. Improvements to accuracy may be possible with careful choice of network and algorithm parameters such as number of convolutional layers, the number of filter channels per layer, the learning rate, the number of MC sampling steps, the number of steps we reuse a MC sampling, and the number of network training steps. It may also be possible that other network architectures could yield even more accurate results with a comparable number of network parameters.
Chapter 6

Appendix

6.1 Lattice Hamiltonian

Real space periodicity of the potential in \( \tilde{H}(r) = -\frac{1}{2} \nabla^2 + V(r) \) leads to a block-diagonal form for the Hamiltonian matrix in reciprocal space. To see this, we express the Fourier matrix elements of \( \tilde{H} \) as

\[
\tilde{H}_{k+G,k'+G'} = \frac{1}{|\Omega|} \int_{\Omega} e^{-i(k+G)r} \tilde{H}(r) e^{i(k'+G')r} \, dr = \\
\left[ \frac{1}{N_R} \sum_{\mathbf{R}} e^{-i(k-k')\mathbf{R}} \right] \frac{1}{|v_0|} \int_{v_0} e^{-i(k+G)\rho} \tilde{H}(\rho) e^{i(k'+G')\rho} \, d\rho = \delta_{kk'} H_{k+G,k'+G'}
\]

where \( \Omega \) and \( v_0 \) are volumes of the supercell and unit cell, respectively, and the summation extends over all lattice vectors in the supercell. We have also used the fact that \( e^{iG\mathbf{R}} = 1 \) and \( \delta_{kk'} = \frac{1}{N_R} \sum_{\mathbf{R}} e^{-i(k-k')\mathbf{R}} \). The key point here is that the Hamiltonian matrix \( \tilde{H} \) is block diagonal with block dimensions \( N_G \times N_G \) at each \( k \) in the Brillouin zone. This has useful consequences for the structure of the wave function matrices \( A \) and \( \tilde{A} \). This is illustrated in Figures 6.1 along with the specific details about how our notation corresponds to row and column indexes.
We introduce a compact notation for matrices with block form such as the Hamiltonian, e.g., $\tilde{H}_{|k|} \equiv \delta_{k k'} \tilde{H}$, where square brackets index the specific blocks of $\tilde{H}$. The Fourier mapping between real and reciprocal space representations allows us to write

$$A_{\beta n}(r) = \frac{1}{\sqrt{N_q}} \sum_{q} e^{i qr} \tilde{A}_{\beta n}(q). \quad (6.2)$$

This expression can be compactly written as a unitary transformation

$$A = U_F \tilde{A}, \quad (6.3)$$

where $U_F \in \mathbb{R}^{N_r \times N_r}$ is a unitary Fourier matrix, $U_F^\dagger = U_F^{-1}$. As we transform between real and reciprocal space, the column indices $\beta$ and $n$ do not change meaning.
Figure 6.1: The reciprocal Hamiltonian $\tilde{H}$ is block-diagonal with block matrix dimensions of $(N_G \times N_G)$.

Figure 6.1: The reciprocal Hamiltonian $\tilde{H}$ is block-diagonal with block matrix dimensions of $(N_G \times N_G)$. 

[Diagram showing block-diagonal structure with matrices $G_{NG}$, $kNk_i$, $G_{NG}$, $k_1$, $G_{NG}$, $k_1$, and $G_{NG}$, $k_1$.]
6.2 Wave function properties

6.2.1 Bloch function properties

The Bloch functions $\tilde{\Psi}$ also assumes a block-diagonal form where each $k$ block corresponds to Bloch functions $\Psi_{kn}$ with crystal momentum $k$ and band index $n$, satisfying the usual eigenvalue equation $\hat{H}_{|k|} \tilde{\Psi}_{kn} = \epsilon_{kn} \tilde{\Psi}_{kn}$. This is illustrated in Figure 6.2 along with the specific details for how each index corresponds to rows and columns of each matrix.

Since each $\tilde{\Psi}_{|k|}$ only contains Fourier coefficients of plane waves with wave vector $k + G$, the following matrix element property holds in real space:

$$\Psi_{kn}(\rho + R) = \Psi_{kn}(\rho)e^{ikR}. \quad (6.4)$$

This shows that the Bloch functions extend over the whole supercell, and their values in the unit cell at $R$ can be obtained from those at $R = 0$ via multiplications by a phase factor as seen in Figure 6.3.

6.2.2 Wannier function properties

While the periodic Bloch functions extend over the entire crystal, another useful choice consists of wave functions that are spatially localized. These are the so-called Wannier functions, which are related to the Bloch functions by a unitary transformation over the column indices:

$$W_{R\beta}(\mathbf{r}) = \frac{1}{\sqrt{N_k}} \sum_{k,m} e^{-i\mathbf{kR}} U_{[k],mn} \Psi_{km}(\mathbf{r}). \quad (6.5)$$

The $N_{\text{bands}} \times N_{\text{bands}}$ matrices $U_{[k]}$ are unitary and $R$ is a column index $\beta$ labeling Wannier functions.

Even though the columns of $W$ are not eigenfunctions of $\hat{H}$ and the Fourier transform
\( \tilde{W} \) is not block diagonal because it mixes different \( k \)-blocks of \( \tilde{\Psi} \), Wannier functions obey a simple relation that follows directly from Equations (6.4) and (2.7). In real space, this relation is

\[
W_{R_j n}(r - R_i) = W_{(R_j + R_i) n}(r),
\]  
(6.6)

showing that only one set of \( N_{\text{bands}} \) columns of \( \tilde{W} \) is unique because all others can be obtained by wrap-shifting the rows by \((j - i)\) blocks (see Figure 6.4). In other words, this relation means that Wannier functions for the whole supercell can be obtained by translating a set of \( N_{\text{bands}} \) Wannier functions centered in one primitive cell. For the Fourier representation \( \tilde{W} \) this means that columns with the same band index \( n \) are related by a phase factor:

\[
\tilde{W}_{R_j n}(k + G) = \tilde{W}_{R_i n}(k + G)e^{ik(R_i - R_j)},
\]  
(6.7)

as demonstrated by Figure 6.5. Because all block matrices in \( \tilde{W} \) are redundant besides a single column block \((R_0)\), due to Equations (6.6) and (6.7), we can simplify our subsequent notation by denoting the unique row blocks as \( \tilde{W}_{[k]} \in \mathbb{R}^{N_G \times N_{\text{bands}}} \).
The Eigenvectors $\tilde{\Psi}$ are block-diagonal with block matrix dimensions of $\left( \frac{N_G \times N_G}{\text{bands}} \right)$ due to the Hamiltonian in Figure 6.1.

<table>
<thead>
<tr>
<th>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</th>
<th>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</td>
<td>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</td>
</tr>
<tr>
<td>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</td>
<td>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</td>
<td>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</td>
</tr>
<tr>
<td>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</td>
<td>$(\Psi + \Psi \Psi \Psi \Psi \Psi \Psi \Psi \Psi)$</td>
</tr>
</tbody>
</table>

\[
\begin{pmatrix}
q^N & \cdots & I \\
\vdots & \ddots & \vdots \\
q^N & \cdots & I
\end{pmatrix} + \begin{pmatrix}
q^N & \cdots & I \\
\vdots & \ddots & \vdots \\
q^N & \cdots & I
\end{pmatrix} + \begin{pmatrix}
q^N & \cdots & I \\
\vdots & \ddots & \vdots \\
q^N & \cdots & I
\end{pmatrix} + \begin{pmatrix}
q^N & \cdots & I \\
\vdots & \ddots & \vdots \\
q^N & \cdots & I
\end{pmatrix}
\]
Figure 6.3: In real space the Eigenfunctions span the entire matrix. Note however, that all block rows can be obtained from the top block row (highlighted) after multiplication with an exponential.

\[
\begin{array}{c|c|c}
\begin{array}{c}
(\hat{N}d)^{N\times N} \phi \cdots (\hat{N}d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}d)^{N\times N} \phi \cdots (\hat{d}d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}^2d)^{N\times N} \phi \cdots (\hat{d}^2d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}^3d)^{N\times N} \phi \cdots (\hat{d}^3d)^{1\times N} \phi
\end{array}
& \cdots \\
\begin{array}{c}
(\hat{N}d)^{N\times N} \phi \cdots (\hat{N}d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}d)^{N\times N} \phi \cdots (\hat{d}d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}^2d)^{N\times N} \phi \cdots (\hat{d}^2d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}^3d)^{N\times N} \phi \cdots (\hat{d}^3d)^{1\times N} \phi
\end{array}
& \cdots \\
\begin{array}{c}
(\hat{N}d)^{N\times N} \phi \cdots (\hat{N}d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}d)^{N\times N} \phi \cdots (\hat{d}d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}^2d)^{N\times N} \phi \cdots (\hat{d}^2d)^{1\times N} \phi \\
& \vdots \\
(\hat{d}^3d)^{N\times N} \phi \cdots (\hat{d}^3d)^{1\times N} \phi
\end{array}
\end{array}
+ \begin{pmatrix}N & \mathbf{r} \end{pmatrix} \begin{pmatrix}N & \mathbf{r} \end{pmatrix}^T
\]
Figure 6.4: For $W$ each matrix element can be obtained by wrap-shifting the first block column (highlighted).
Figure 6.5: For $\tilde{W}$ each matrix element can be obtained by multiplying elements from the first block column (highlighted) by an exponential of specific $k$, $G$, and $R$ index.
6.3 Nesterov’s accelerated gradient method on the Stiefel manifold

For brevity, we omit subscript [k] on the Wannier function blocks $\tilde{X}[k]$. We also omit the tilde because the following equations are the same for real and reciprocal space functions. Figure 6.6 illustrates the key steps of Nesterov’s accelerated gradient method on the Stiefel manifold (NSM) presented in Refs. [16, 17]. Note that a number of tangent space projections and manifold retractions are needed to ensure a gradient based approach steps along the sphere and as a result preserves orthogonality. For example, a tangent space projection of $-G$ yields $-P_T(G)$. We can then perform the manifold retraction $R(Y^{(i)}, -sP_T(G))$ to obtain the point $X^{(i+1)}$; with $s$ being some finite step size.

The algorithm works with the canonical metric of the Stiefel manifold, which defines a scalar product in the tangent space at point $X$:

$$g_c(V, V) = \text{Tr} \ V^\dagger \left( I - \frac{1}{2}XX^\dagger \right) V,$$

(6.8)

where the tangent space vectors satisfy $V^\dagger X + X^\dagger V = 0$. The main reason for using the canonical Stiefel metric in place of the seemingly more natural Euclidean expression $\text{Tr} \ V^\dagger V$ is that the former correctly counts contributions of independent degrees of freedom, see Ref. [33] for details. This metric is used to project vectors from the full Euclidean space, such as the objective function gradient,

$$G = \frac{\partial}{\partial Y} F(Y)$$

(6.9)
Figure 6.6: Schematic illustrating the method of Refs. [16, 17] on the Stiefel manifold (sphere). The pink plane represents the Stiefel tangent space. Both the blue and red lines represent different Stiefel geodesics pointing in the direction of the optimum solution. The dashed line represents the Euclidean path that can be used to extrapolate from $X^{(i+1)}$ towards $Y^{(i+1)}$. The dots mark the key variables in the Nesterov Stiefel method which all occur on the sphere.
onto the tangent space of the Stiefel manifold at point $X$ via

$$P_T(G) = G - XG^\dagger X.$$  \hfill (6.10)

To perform a finite step along the direction of descent, $-sP_T(G)$, one needs to project a finite vector from the tangent space onto the Stiefel manifold. This is accomplished with the help of retraction $R(X, -sP_T(G))$, which relates vectors in the tangent space to points on the manifold. Loosely speaking, retraction represents a geodesic curve on the manifold which goes through $X$ and is tangent to $P_T(G)$ at $X$. Exact expression for the retraction involves a hard-to-evaluate matrix exponential [33], while simple and accurate formulas can be developed based on Padé approximants [16]; in this work, we use the first-order approximation proposed in Ref. [54]. It yields the following generalization for the top line in Nesterov’s algorithm Eq. (2.14) on the Stiefel manifold [16, 17]:

$$X^{(i+1)} = Y^{(i)} + 2J(I - Z^\dagger J)^{-1}Z^\dagger Y^{(i)}.$$  \hfill (6.11)

$J = [-sG, Y^{(i)}]$ and $Z = [Y^{(i)}, sG]$ represent column-wise concatenations of two matrices; the resulting matrices have dimensions $(2N_{\text{bands}} \times N_G)$. Note that $G$ is the gradient at $Y^{(i)}$. Subsequently, the following 3-step procedure is denoted as GradStep: 1) calculating $G$, 2) performing a finite step using Eq. (6.11), and 3) choosing a step size $s$ which satisfies the Armijo condition [55]:

$$F(X^{(i+1)}) \leq F(X^{(i)}) - s\|G\|^2_g,$$  \hfill (6.12)

where objective function is

$$F(Y) = \text{Tr} Y^\dagger HY + \mu \text{Tr} (Y - L)^\dagger(Y - L).$$  \hfill (6.13)
The Stiefel norm of the gradient is given by \( \|G\|_g^2 = \text{Tr} \left( D^T D \right) \) and

\[
D = G - \frac{1}{\sqrt{2}} X^{(i)} G^\dagger X^{(i)} - \left( 1 - \frac{1}{\sqrt{2}} \right) X^{(i)} X^{\dagger (i)} G.
\]

The second step in Nesterov’s algorithm requires a method to extrapolate through two points on the Stiefel manifold, \( Y^{(i+1)} = (1 - \alpha) X^{(i+1)} + \alpha X^{(i)} \). This is nontrivial because a simple averaging between \( X^{(i)} \) and \( X^{(i+1)} \) would take us off the manifold and violate the orthogonality constraint. One could perform the extrapolation in Euclidean space and project back onto the Stiefel manifold, but this is neither accurate nor efficient. The method introduced in Refs. [16, 17] uses retraction with the following basic idea. To perform extrapolation, we look for a geodesic line that starts at \( X^{(i+1)} \) and goes through \( X^{(i)} \). This geodesic has a tangent at \( X^{(i+1)} \) which we denote by \( V \). The geodesic line near \( X^{(i+1)} \) is given by retraction \( R(X^{(i+1)}, \alpha V) \), and we want to fix \( V \) so that the retraction goes through \( X^{(i)} \). This is simply done by solving \( R(X^{(i+1)}, V) = X^{(i)} \). Then the second (extrapolation) step of the NSM algorithm can be shown to be [16]:

\[
Y^{(i+1)} = X^{(i)} + 2J(I - Z^\dagger J)^{-1}Z^\dagger X^{(i)}, \tag{6.14}
\]

where \( V = 2X^{(i+1)}(I + X^{(i)} X^{(i+1)})^{-1} \), \( J = [(1 + \beta^{(i)}) V, X^{(i)}] \), and \( Z = [X^{(i)}, -(1 + \beta^{(i)}) V] \) with \( \beta^{(i)} = (i - 1)/(i + 2) \). We will refer to Eq. (6.14) as Extrapolate in pseudo code.

112
Bibliography


