Computational Design of Materials Lab 5

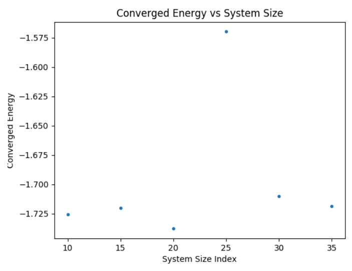
Andrew Sullivan

May 2, 2019

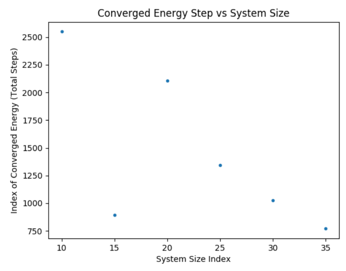
1 Problem 1: Monte Carlo and the Ising Model

In this section, we employ the classic Ising model with two spin states (up and down, or +1 and -1) in a 2D square lattice and simulate using a Markov chain Monte Carlo (MCMC) implementation of the Metropolis algorithm. We set $J = k_B$ for simplicity, so the energy is given by $E = k_B \Sigma_{i,j} S_i S_j - H \Sigma_i S_i$, where spins are denoted by S_i and S_j, k_B is the Boltzmann constant, and H is the magnetic field, which we set to zero for the duration of this problem.

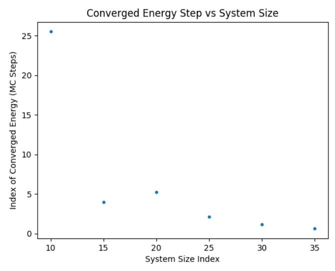
1.1 A: Convergence Calculations



(a) Converged Average Energy.

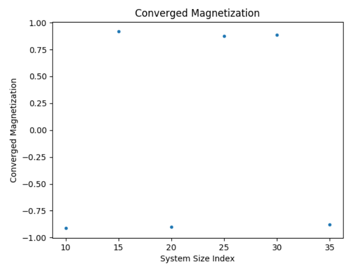


(b) Index of Convergence.

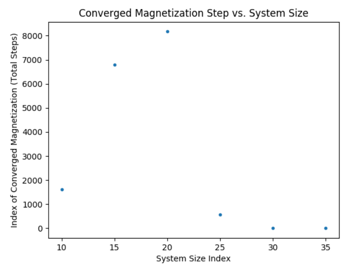


(c) Index of Convergence per Site.

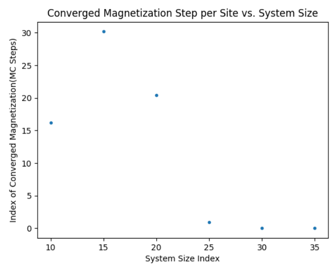
Figure 1: Energy Convergence vs. Size in Ising Model.



(a) Converged Magnetization.



(b) Index of Convergence.



(c) Index of Convergence per Site.

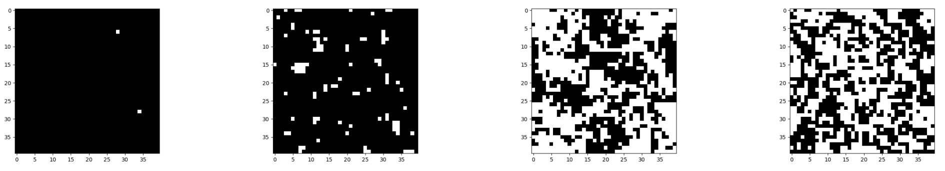
Figure 2: Magnetization Convergence vs. Size in Ising Model.

1.1.1 i)

We begin by exploring the convergence behavior of the system. Here, we consider convergence initially in terms of equilibration of the system energy, and subsequently explore how both the average energy and magnetization equilibrate as a function of system size, which is specified as one dimension of a two-dimensional square lattice. A function was written to identify energy convergence using two criteria. In a previous lab, a convergence paradigm was used in which average energies were calculated by incrementally removing the first energy value in the average, but this proved to be too computationally expensive for the large (several thousand elements) arrays dealt with here. Simulations for convergence were run for 800 MC steps of equilibration and 500 steps of statistics collection. Here, a Monte Carlo step is defined as the equivalent to one sweep over the lattice (N x N spin flip attempts). Instead, the first criterion in convergence is a sufficiently small change in average energy between subsequent values for a specified number of consecutive points. Here, we begin by finding the average energy from step 1 to step i. We then compare this average energy to that obtained from step 1 to step $i+1$. If the difference between these two averages is sufficiently small, a counter is incremented. We set this threshold to be 0.001. If the difference is larger, the counter is set back to zero. Once the counter reaches 100, this criterion is satisfied. However, one hundred consecutive values with differences of 0.001 may instead correspond to a small but persistent change in the energy rather than a truly converged value. Thus, we also require that the average of each of these increments dE over the 100 steps to have an absolute value less than some other threshold (here set to 10^{-6}).

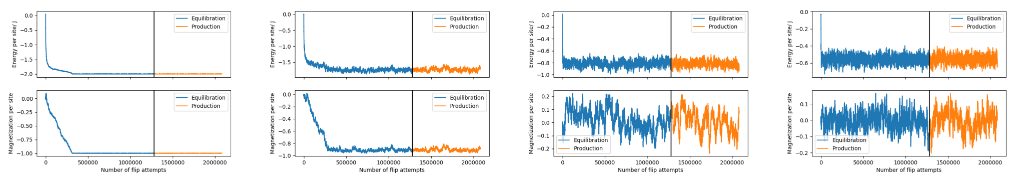
With these criteria, we obtain the trends shown in Figures 1 and 2 for energy and magnetization, respectively. The first plot in each figure shows the converged average energy or average magnetization value obtained from system sizes between 10 and 35. For energy, aside from the outlier at N=25, all energy values are within approximately 0.025, and there is no noticeable trend, as expected if convergence is accurate. A similar trend is observed for magnetization, though we also see the randomness associated with system initialization and spin selection, which could lead to the system converging to either spin-up or spin-down states (note that T=2 for these simulations, below the transition to disorder). The second two plots show the index at which the system reaches convergence, and the final two show the index in terms of the Monte Carlo step, rather than the spin flip attempt number (i.e. the values in the last plot are the values in the second divided by the number of cells, N^2). We see a general downward trend in the converged index, especially in the Monte Carlo step for convergence. As larger systems have more configurational degrees of freedom, it is possible that these systems will trend toward equilibrium more quickly due to the increased ability to explore configurational space. We also note that the system appears to take longer to converge in terms of magnetization, especially for smaller values of N, highlighting the importance of checking the properties of interest rather than any property of the system in convergence checks.

It should also be noted that system autocorrelation is not explored in depth here, though a function was constructed to examine autocorrelation of the equilibrated system. Adjacent values of observables (e.g. energy) are highly correlated, differing by at most one spin, which can increase the variance of the average. Initial tests suggest that autocorrelation is not highly significant here, as energy values obtained by, for example, taking every Nth or N^2th energy value do not differ substantially from those obtained by averaging all energies. Statistical randomness between different simulation trials owing to stochasticity in initialization, spin selection, and drawing of random numbers for acceptance criteria, is far more significant in explaining energy differences between simulations.



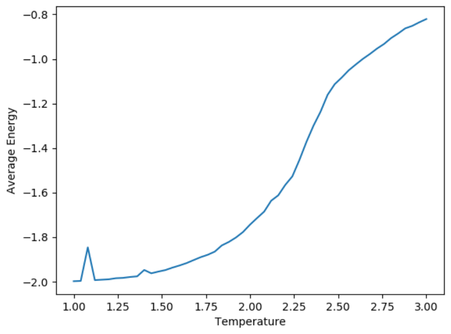
(a) T=1. (b) T=2. (c) T=3. (d) T=4.

Figure 3: Final State of the System at Different Temperatures.

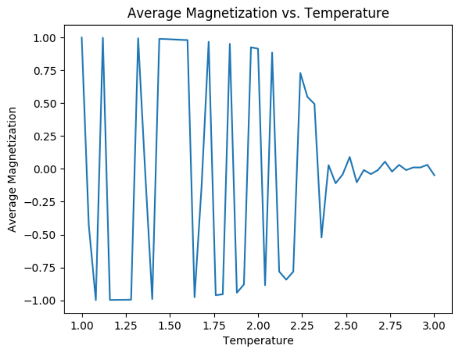


(a) T=1. (b) T=2. (c) T=3. (d) T=4.

Figure 4: Energy and Magnetization at Different Temperatures.

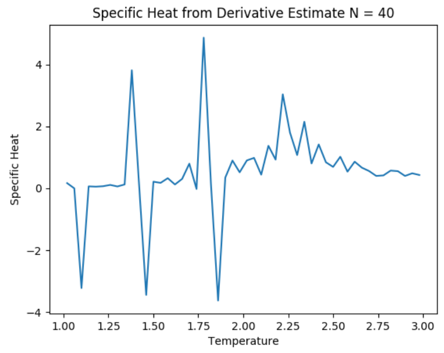


(a) Energy vs. Temperature.

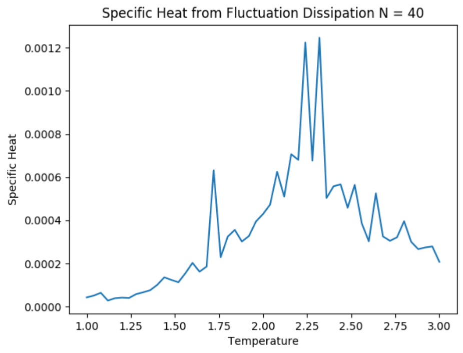


(b) Magnetization vs. Temperature.

Figure 5: Temperature Trends in Observable Properties.



(a) C_V from Derivative Approximation.



(b) C_V from Energy Fluctuations.

Figure 6: Specific Heat for N=40.

1.1.2 ii)

In this part, we explore the temperature dependence of the system with fixed size (N = 40). We initially visualize the final state of the system at four different temperatures (T = 1, 2, 3, 4) after 800 equilibration steps and 500 statistical collection steps. This initial scan clearly suggests that the transition is somewhere between T=2 and T=3, as the system goes from a state with mostly one spin and an average magnetization around either $+1$ or -1 to a system with approximately equal numbers of both spins and an average magnetization that fluctuates largely around 0. Plotting the average magnetization vs. system temperature for N=40, however, does not clearly show where the phase transition is located. This plot in Figure 5 may instead suggest a value closer to 2.50 than 2.25. If we instead plot the average energy as a function of temperature, we can estimate the transition as the point at which the energy changes most rapidly (i.e. the specific heat is maximal). This would yield a value of approximately 2.3 to 2.35, slightly above the ideal value of 2.27.

1.2 B: Specific Heat

1.2.1 i)

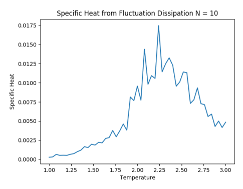
Next, we explore the specific heat, calculated in two ways, as a function of system temperature for fixed size, and then consider the size dependence. We initially compute the specific heat in the canonical ensemble by approximating the derivative $C_V = (\frac{\partial E}{\partial T})_V$ using a first-order central difference approximation by calculating the energy difference between systems separated by a fixed temperature increment, ΔT, which we set to

0.05. The derivative is then approximated as $\frac{\Delta E}{\Delta T} = \frac{E(T+\Delta T/2)-E(T-\Delta T/2)}{\Delta T}$. Therefore, each derivative we estimate using this formula is the approximation for the specific heat at a temperature halfway between the points at which the energy is evaluated. A plot of these values shows a number of large fluctuations at low temperatures, which may correspond to states which have not completely equilibrated rather than a large change in the energy. The phase transition, however, corresponds to the largest peak surrounded by a number of smaller peaks, i.e. a gradual increase in the specific heat vs. T rather than one sharp peak. From this method, we obtain a value of 2.28, quite close to the ideal value. However, we also note that using an automated algorithm that finds the maximum in the specific heat curve would not be ideal for this method, due to the large fluctuations at low temperatures.

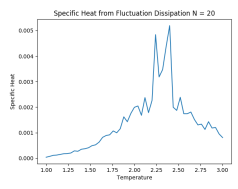
1.2.2 ii)

Rather than directly approximating the derivative, which is subject to statistical fluctuations in the energy, we can consider an alternate formula for the specific heat, which can be derived from fluctuation dissipation theorem. We can expand the definition of the heat capacity using the identity $\beta = \frac{1}{k_B T}$ as $C_V = (\frac{\partial \langle E \rangle}{\partial T}) = -\frac{\beta}{T}\frac{\partial \langle E \rangle}{\partial \beta} = \frac{\beta}{T}\frac{\partial^2 lnZ}{\partial \beta^2}$, where Z is the canonical partition function and we have used the identity $\langle E \rangle = -\frac{\partial lnZ}{\partial \beta}$. The second derivative can then be altered by noting that $\frac{\partial lnZ}{\partial \beta} = \frac{\partial lnZ}{\partial Z}\frac{\partial Z}{\partial \beta}$ and that $\frac{\partial^2 lnZ}{\partial \beta^2} = \frac{\partial}{\partial \beta}\frac{\partial lnZ}{\partial \beta}$. Therefore, we have $C_V = \frac{\beta}{T}\frac{\partial}{\partial \beta}(\frac{1}{Z}\frac{\partial Z}{\partial \beta}) = \frac{\beta}{T}[\frac{1}{Z}\frac{\partial^2 Z}{\partial \beta^2} - \frac{1}{Z^2}(\frac{\partial Z}{\partial \beta})^2]$. Finally, we note that the second term in the brackets is simply the square of the expected value of energy, $\langle E \rangle^2$, and that the first term in the brackets is the expected value of the square of energy, $\langle E^2 \rangle$. Using the definition of the variance, $\sigma^2(E) = \langle E^2 \rangle - \langle E \rangle^2$, we can thus write the heat capacity as $C_V = \frac{\beta}{T}\sigma^2(E) = \frac{\sigma^2(E)}{k_B T^2}$.

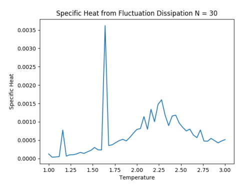
We do not consider the k_B in the denominator in our calculations, and therefore determine specific heat simply as $\sigma^2(E)/T^2$. As a result, the magnitudes of our specific heat are not directly comparable to those obtained from the previous method. However, it is clear that using this method produces a much more noticeable peak around the transition temperature. Smaller temperature increments may be beneficial in determining which of the two adjacent peaks is truly larger, but in either case the peak location is close to the ideal value of 2.27 (averaging the two peak locations gives a value of 2.28). Given that this latter method relies on calculating the variance, which itself requires calculation of the mean, and the other method only requires calculation of the mean, it seems likely that the former method is more numerically efficient. However, in order to obtain more accurate estimates, smaller temperature increments (and perhaps averaging over many simulations) are required for the derivative estimate, so this method may actually be more expensive in certain cases to achieve comparable accuracy and reduce low-temperature fluctuations.



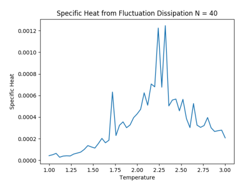
(a) N=10.



(b) N=20.



(c) N=30.



(d) N=40.

Figure 7: Specific Heat vs. T for Different System Sizes.

1.2.3 iii)

Using the method based on the variance of the equilibrium energy at different temperatures, we next explore the influence of system size on the trend of specific heat vs. temperature. We plot the specific heat vs. temperature plot at four different system sizes: 100, 400, 900, and 1600 spins. In general, the major trend in these plots is a sharpening of the peak near the transition temperature. Neglecting the anomalous sharp peak in the N=30 plot, we can see that the peaks become smaller as the system size increases, improving the accuracy of transition identification. We can also see, however, that the magnitude of the specific heat appears to decrease as a function of system size, suggesting that fluctuations become smaller (on a per-atom basis) as more atoms are present.

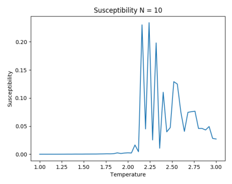
1.3 C: Magnetic Susceptibility

1.3.1 i)

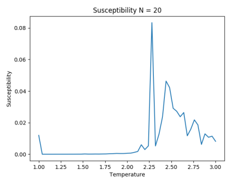
The derivation for magnetic susceptibility, χ, proceeds similarly to that of heat capacity above. The susceptibility is defined as $\chi = \frac{\partial \langle M \rangle}{\partial H}$, where M is the magnetization and H is the applied magnetic field. We begin by noting that the partition function in the canonical ensemble is $Z = \Sigma_{\{S_i\}} e^{-\beta E(S_i)}$, and using the definition of the Hamiltonian for the Ising model, $E = k_B \Sigma_{i,j} S_i S_j - H \Sigma_i S_i$, we have the Ising partition function $Z = \Sigma_{\{S_i\}} e^{-\beta(k_B \Sigma_{i,j} S_i S_j - H \Sigma_i S_i)}$. Note that the sum here is over all possible combinations of spin, i.e. over all the configurational degrees of freedom, rather than over all spins within a given configuration, as indicated by the sums in the exponential. We also note that the magnetization is $M = \Sigma_i S_i$, and that the average magnetization is then $\langle M \rangle = \frac{1}{Z} \Sigma_{\{S_i\}} M e^{-\beta(k_B \Sigma_{i,j} S_i S_j - HM)}$.

We can differentiate this expression with respect to the field, H, and using the product rule we obtain $\frac{\partial \langle M \rangle}{\partial H} = \frac{\partial (1/Z)}{\partial H} \Sigma_{\{S_i\}} M e^{-\beta(k_B \Sigma_{i,j} S_i S_j - HM)} + \frac{1}{Z} \frac{\partial}{\partial H} \Sigma_{\{S_i\}} M e^{-\beta(k_B \Sigma_{i,j} S_i S_j - HM)}$. In the first term, we note that $\frac{\partial (1/Z)}{\partial H} = \frac{-1}{Z^2} \frac{\partial Z}{\partial H}$. In the second term, only the second term in the exponential is dependent on the magnetic field, so we can bring the partial derivative into the summation as $\Sigma_{\{S_i\}} M \frac{\partial}{\partial H} e^{-\beta(k_B \Sigma_{i,j} S_i S_j - HM)}$. Evaluating this derivative, we obtain the final form for the second term, $\frac{\beta}{Z} \Sigma_{\{S_i\}} M^2 e^{-\beta(k_B \Sigma_{i,j} S_i S_j - HM)}$. By our previous definition of the expected value of M, we note that this is β times the expected value of M^2, or $\beta \langle M^2 \rangle$.

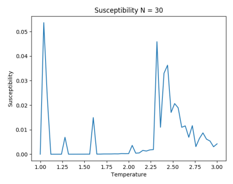
To evaluate the first term, we must determine the derivative of the partition function with respect to H. As before, we note that only the second term in the expontential is dependent on H, and thus $\frac{\partial Z}{\partial H} = \beta \Sigma_{\{S_i\}} M e^{-\beta(k_B \Sigma_{i,j} S_i S_j - HM)}$. Thus, our first term becomes $\frac{-\beta}{Z^2} (\Sigma_{\{S_i\}} M e^{-\beta(k_B \Sigma_{i,j} S_i S_j - HM)})^2 = -\beta \langle M \rangle^2$. Thus, we can write the magnetic susceptibility, $\chi = \frac{\partial \langle M \rangle}{\partial H} = \beta(\langle M^2 \rangle - \langle M \rangle^2) = \beta \sigma^2(M) = \frac{\sigma^2(M)}{k_B T}$. Therefore, just as the specific heat can be obtained from the equilibrium fluctuations of energy, the magnetic susceptibility can be obtained from the equilibrium fluctuations of the magnetization.



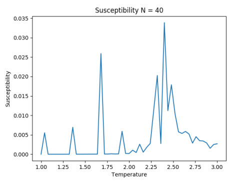
(a) N=10.



(b) N=20.



(c) N=30.



(d) N=40.

Figure 8: Magnetic Susceptibility vs. T for Different System Sizes.

1.3.2 ii)

As with the specific heat, we may use the above method based on equilibrium statistical fluctuations to explore the temperature dependence of the susceptibility and how this relationship changes for different system sizes. In general, we see similar trends to those obtained in the energy plots, i.e. that the peak region broadens around the phase transition for smaller system sizes but the susceptibility magnitude increases. However, it also appears that the susceptibility is a less reliable estimate for detecting the phase transition than the energy, since the largest peaks in the N=30 and N=40 cases, for example, are closer to 2.4 than 2.3. This is consistent with the magnetization vs. temperature plot presented earlier, in which the phase transition was difficult to identify due to large fluctuations throughout the temperature range below the transition. As mentioned previously, the system seems to take longer to converge in terms of magnetization, so it is possible that the transition would become clear at longer simulation times. Additionally, it should be noted that system sizes above N=40 were not explored rigorously due to high computational cost. For further improvements in accuracy, increasing system size is one option, in addition to longer times.

2 Problem 2: Nudged Elastic Band and Machine Learned Potentials

In this problem, we employ the nudged elastic band (NEB) method to explore activation energy barriers for adatom diffusion between adjacent lattice sites on an Al (111) surface. This method is based on providing two endpoint configurations and constructing a number of ”images” spaced between the two configurations connected by elastic spring forces that allow the system to explore the energy at different points during the transition. We use this method in conjunction with three different models to evaluate the energy (and forces on atoms).

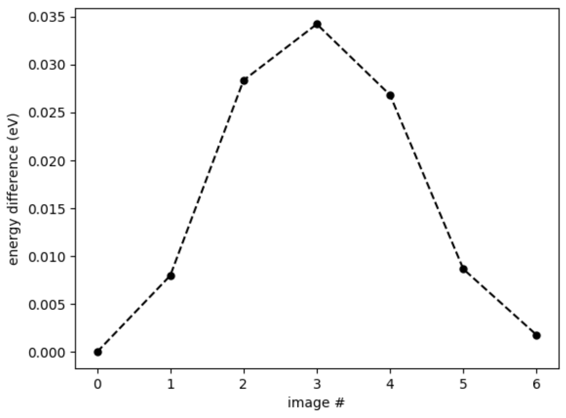


Figure 9: Energy Profile of HCP-FCC Site Transition from EAM .

2.1 A: EAM Activation Barrier

We begin by using the Zhou embedded atom method (EAM) potential for Al and construct a 2x2x3 Al (111) slab using the atomic simulation environment (ASE) package in Python. An adatom is placed initially in an HCP site with a 5 Å vacuum on either side and the default Al lattice parameter, and subsequently in an adjacent FCC site with the same parameters. We then use NEB to determine the energies at different points along the transition between these two states.

2.1.1 i)

Using the Zhou EAM potential for Al, we obtain a relaxed potential energy of -42.791422 eV/atom for the HCP site. Though the energy units are not explicitly specified in the output, eV is consistent with the example found in the EAM calculator in the ASE documentation.

2.1.2 ii)

If we instead use an FCC site, the energy is found to be -42.789640 eV/atom. This energy is slightly larger (more positive) than that found in the previous part, suggesting that the HCP site is slightly more stable for adatoms than the FCC site.

2.1.3 iii)

Using the nudged elastic band method, we find the potential energy of each interpolated configuration along the transition path with a total of 7 images. The forces are also saved for subsequent comparison with DFT and GP methods. The above figure shows the potential energy difference along the transition path (i.e. the difference between the energy of the current image and of the starting configuration).

2.2 B: Validation of the EAM Barrier

We now employ the density functional theory (DFT) method to compute the energies based on the electronic ground state of the system. This method is more computationally intensive but gives more accurate results. Here, we use a k-point grid of 4x4x1, energy cutoff of 29 Ry, and charge density cutoff of 143 Ry. These values are specified in a data input variable. Given that Al is a metal, we also include smoothing with a coefficient of 0.02. The pseudopotential was obtained by installing the PSlibrary for Python and using the ld1.x function to install pseudopotentials for Al. The Al.pde-n-rrjkus-psl.1.0.0.UPF USPP was chosen, and the output from ld1.x was modified to be consistent with the compiler being used in PyCharm (the initial output was an XML file containing CDATA sections which were not read correctly by the compiler).

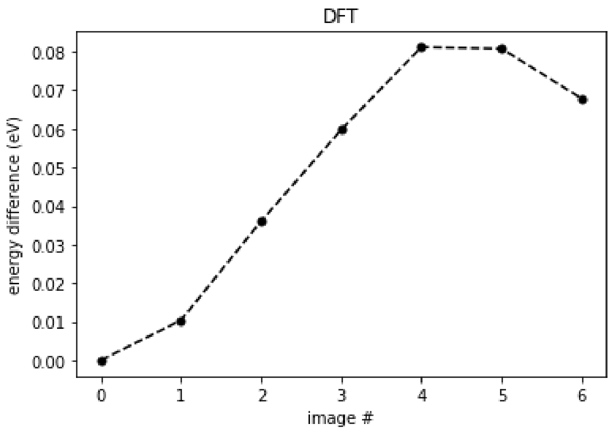


Figure 10: Energy Profile Obtained by Applying DFT to Relaxed Images from EAM.

2.2.1 i)

In order to validate the EAM activation barrier, we take each of the seven images obtained from BFGS NEB relaxation and set their calculators to the Espresso DFT calculator with the specified pseudopotential. We then re-calculate the energies and forces. There are two prominent differences between the two energy profiles. The most noticeable is that the profile is no longer symmetrical. Instead of the central configuration (3) having the largest energy, the two configurations closer to the FCC site have the largest differences. Consistent with EAM, the FCC site has a higher energy than the HCP site. The second major difference is the magnitude of the energy. It should be noted that the quantities being plotted between EAM and DFT are not truly the same, as EAM returns the energy in eV/atom. DFT, instead, returns the total energy in eV, and thus the energy differences above should be in eV/13 atoms. This was not converted in order to compare with the energies obtained for the GP model below, as discussed later. In the DFT output file, the total energy is specified as approximately -65 Ry, and the get potential energy function from the calculator returns a value of -887, consistent with the conversion between Ry and eV of 13.6. On a per-atom basis, this yields energies of approximately -68 eV/atom, more negative than those obtained from EAM. Performing this conversion to eV/atom, we find that the transition state energy differences are actually much smaller than those obtained from EAM (around 0.006 for the largest).

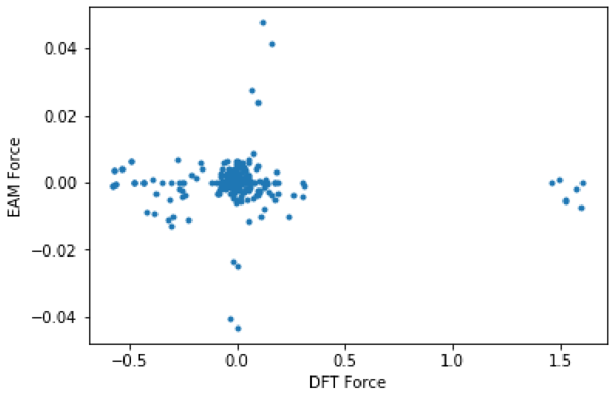


Figure 11: Force Relationship Between DFT and EAM.

2.2.2 ii)

Comparing the forces, we find that the range of forces is substantially larger for DFT than EAM potentials, and that large forces (around 1.5 eV/Å) in the DFT evaluation are nearly zero using the EAM method. It seems likely that most of these large forces are those in the z-direction, as the BFGS relaxation was performed on the initial structure using the EAM potential, and so the EAM-relaxed configuration is likely not the same as that which would have been obtained from relaxing with DFT. We find a mean absolute error of 0.1313 eV/Å using the EAM method relative to DFT.

2.3 C: Machine Learned Activation Barrier

Finally, we explore the development of a machine-learned potential model based on the above DFT force results. The model is trained using a Gaussian process with cutoffs of 4.9 Å, length scales of 1 Å, and standard deviations of 1.0 ev/Å and 0.1 eV/Å for the two- and three-body signals, respectively. The noise standard deviation is set to 1 meV/Å.

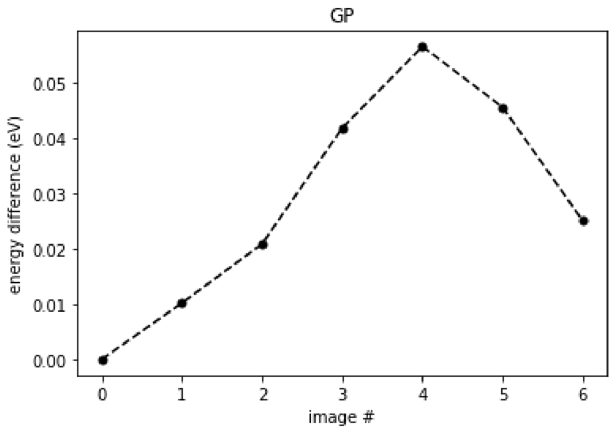
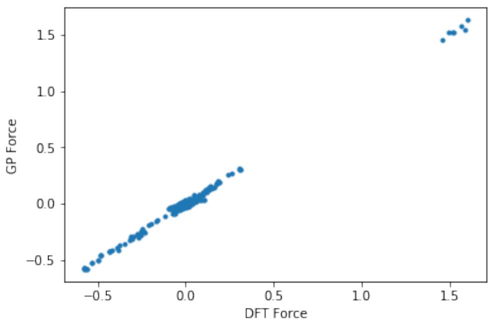


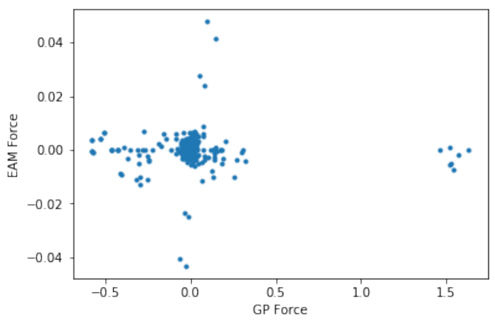
Figure 12: Energy Profile Obtained from Gaussian Process Model Trained on DFT Forces.

2.3.1 i)

Despite the GP model only being trained on forces and not energies, we find that the model does quite a good job of recapitulating the results from DFT, i.e. that images 4 and 5 have the largest energies and the



(a) Forces from GP and DFT.



(b) Forces from GP and EAM.

Figure 13: Forces from Gaussian Process Model.

profile is not symmetric about the central image. It can be observed that the maximal difference is smaller than that obtained from DFT, but is closer on a per-atom basis (around 0.005) than EAM.

2.3.2 ii)

Finally, we compare the forces obtained from the Gaussian Process model to those obtained from both Density Functional Theory and the Embedded Atom Method. Unsurprisingly, the forces from GP and DFT methods match almost exactly (the mean absolute error is now an order of magnitude smaller, at 0.013 eV/Å), as these are the forces on which the model was trained. Similarly, the pattern in forces vs. those from EAM is consistent with that observed from DFT vs. EAM. Thus, the GP model, though initially requiring DFT calculations for training, performs much better at producing DFT-accurate results in a comparable time to EAM.

3 Appendix

The first code provided is that used for the Markov Chain Monte Carlo (MCMC) Ising model. There are four subsections that can be used to compute different system properties. For Q = 1, the system is simulated at a fixed temperature for a specified number of Monte Carlo steps. The array of energy values is fed into the converge function, which iteratively calculates the average energy and finds the first energy value at which the average changes by less than some threshold (valthresh) for a fixed number of consecutive points (countthresh). An additional requirement is imposed to ensure that the average energy is relatively constant over this range and is not just changing by small increments at each step. This is confirmed by checking the average incremental change over the countthresh time points and ensuring that its absolute value is below some value (Ethresh). Then, the trends of the converged energy and the index at which convergence is achieved are plotted as a function of system size. For Q = 2, the system is simulated at fixed system size over the specified temperature range. From the energy array for a given temperature, the average energy is obtained by averaging over the E vector, corresponding to energy values after equilibration. The standard deviation is obtained from the same array. The specific heat vs. temperature relationship is calculated using an approximation to the derivative by taking the difference of consecutive energy values and dividing by the temperature increment and by using the fluctuation-dissipation relationship in which the specific heat is proportional to the variance of energy over the squared temperature. Q=3 and Q=4 are used to perform the same calculations for the magnetization. A function was also written to explore system autocorrelation but was not studied intensively due to time constraints.

```python
from labutil.src.plugins.ising import *
import matplotlib.pyplot as plt
import numpy as np

def run_mcmcising(N,n_eq,n_mc,T):
    #### RUN MC CODE ####
    E, M, E_eq, M_eq, imagelist = mc_run(N=N,n_eq=n_eq,n_mc=n_mc,T=T)

    x_eq = np.arange(n_eq*N*N)
    x_mc = np.arange(n_mc*N*N)+(n_eq*N*N)

    return(E,M,E_eq,M_eq,x_eq,x_mc,imagelist)

def autocorr(E):
    l = len(E)
    n = 1                              # spacing of dt to test
    N = 500                            # total number of dt to test
    acorr = []
    for dt in range(1,N,n):
        m = l - max(range(1,N,n))
        s1=0
        s2=0
        for t in range(0,m):
            s1 = s1 + E[t]
            s2 = s2 + E[t]*E[t+dt]
        E1 = s1/m
        E2 = s2/m
        acorr.append((E2 - E1**2))
    # plt.plot(range(1,N,n),acorr)
    # plt.xlabel('Separation Between Points')
    # plt.ylabel('Autocorrelation')
    # plt.title('Autocorrelation Function')
    # plt.show()
    return(acorr)

def converge(E):
    Eavg = []
    countthresh = 100
    valthresh = 0.001
    Ethresh = 10**(-6)
    Eavg1 = np.mean(E)
    count = 0
    for i in range(1, len(E)):
        Eavg2 = np.mean(E[i:len(E)])
        Eavg.append(Eavg2)
        dE = []
        if abs(Eavg2 - Eavg1) < valthresh:
            count = count + 1
            dE.append(Eavg2 - Eavg1)
            if count > countthresh and abs(np.mean(dE)) < Ethresh:
                ind = i-100
                print(np.mean(dE))
                break
        else:
            count = 0
            dE = []
        Eavg1 = Eavg2
    conv = np.mean(E[ind:len(E)])
    print(ind, conv)
    print('ind, conv')
    return conv, ind


if __name__ == '__main__':
    # put here the function that you actually want to run
    Q = 3

    if Q == 1:
        Econv = []
        Eind = []
        Eindpersite = []
        Corrtime = []
        for N in range (10,40,5):
            T=2
            n_eq=800          # Average number of equilibration steps (flips) per site
            n_mc=200         # Average number of Monte Carlo steps (flips) per site
            print(N)

            [E,M,E_eq,M_eq,x_eq,x_mc,imagelist]=run_mcmcising(N=N,n_eq=n_eq,n_mc=n_mc,T=T)
            # acorr = autocorr(E)
            # thresh = next(i for i in acorr if i < 0.1)
            # Corrtime.append(thresh)

            Etot = E_eq + E
            [ec, eind] = converge(Etot)
            Econv.append(ec)
            Eind.append(eind)
            Eindpersite.append(eind / (N * N))
            # plt.plot(Etot)
            # plt.show()

            # fig, (ax_energy, ax_mag) = plt.subplots(2, 1, sharex=True, sharey=False)
            #
            # ax_energy.plot(x_eq, E_eq, label='Equilibration')
            # ax_energy.plot(x_mc, E, label='Production')
            # ax_energy.axvline(n_eq * (N ** 2), color='black')
            # ax_energy.legend()
            # ax_energy.set_ylabel('Energy per site/ J')
            #
            # ax_mag.plot(x_eq, M_eq, label='Equilibration')
            # ax_mag.plot(x_mc, M, label='Production')
            # ax_mag.axvline(n_eq * (N ** 2), color='black')
            # ax_mag.legend()
            # ax_mag.set_ylabel('Magnetization per site')
            # ax_mag.set_xlabel('Number of flip attempts')
            #
            # animator(imagelist)
            # plt.show()

        plt.plot(range(10,40,5),Eind, '.')
        plt.xlabel('System Size Index')
        plt.ylabel('Index of Converged Energy (Total Steps)')
        plt.title('Converged Energy Step vs System Size')
        plt.show()
        plt.plot(range(10,40,5),Econv, '.')
        plt.xlabel('System Size Index')
        plt.ylabel('Converged Energy')
        plt.title('Converged Energy vs System Size')
        plt.show()
        # plt.plot(range(10,40,5),Corrtime)
        # plt.xlabel('System Size Index')
        # plt.ylabel('Correlation Time')
        # plt.show()
        plt.plot(range(10, 40, 5), Eindpersite, '.')
        plt.xlabel('System Size Index')
        plt.ylabel('Index of Converged Energy (MC Steps)')
        plt.title('Converged Energy Step vs System Size')
        plt.show()

    elif Q == 2:
        Eavg=[]
        Ttest=[]
        Estd=[]
        for T in np.linspace(1,3,51):
            N=40
            n_eq=800
            n_mc=500
            Ttest.append(T)
            print(T)
            [E,M,E_eq,M_eq,x_eq,x_mc,imagelist]=run_mcmcising(N=N,n_eq=n_eq,n_mc=n_mc,T=T)

            Esample = E[0:len(E):N*N]
            print(len(Esample))
            Eavg.append(np.mean(Esample))
            Estd.append(np.std(Esample))
            # plt.plot(E_eq[0:len(E_eq):N*N] + E[0:len(E):N*N])
            # plt.show()
        Cv = []
        Cv2 = []
        dT = Ttest[1] - Ttest[0]
        for i in range(0,len(Eavg)-1):
            dE = Eavg[i+1]-Eavg[i]
            Cv.append(dE/dT)
        for i in range(0,len(Estd)):
            Cv2.append(((Estd[i]**2)/Ttest[i]**2))
        Tnew = Ttest[0:(len(Ttest)-1)]+dT/2
        plt.plot(Tnew,Cv)
        plt.xlabel('Temperature')
        plt.ylabel('Specific Heat')
        plt.title('Specific Heat from Derivative Estimate N = 40')
        plt.show()
        plt.plot(Ttest,Cv2)
        plt.xlabel('Temperature')
        plt.ylabel('Specific Heat')
        plt.title('Specific Heat from Fluctuation Dissipation N = 40')
        plt.show()
        plt.plot(Ttest,Eavg)
        plt.xlabel('Temperature')
        plt.ylabel('Average Energy')
        plt.show()

    elif Q == 3:
        Mconv = []
        Mind = []
        Mindpersite = []
        Corrtime = []
        for N in range(10, 40, 5):
            n_eq = 800  # Average number of equilibration steps (flips) per site
            n_mc = 500  # Average number of Monte Carlo steps (flips) per site
            print(N)

            [E, M, E_eq, M_eq, x_eq, x_mc, imagelist] = run_mcmcising(N=N, n_eq=n_eq, n_mc=n_mc, T=T)
            # acorr = autocorr(M)
            # thresh = next(i for i in acorr if i < 0.1)
            # Corrtime.append(thresh)

            Mtot = M_eq + M
            [mc, mind] = converge(Mtot)
            Mconv.append(mc)
            Mind.append(mind)
            Mindpersite.append(mind / (N * N))

            fig, (ax_energy, ax_mag) = plt.subplots(2, 1, sharex=True, sharey=False)

            ax_energy.plot(x_eq, E_eq, label='Equilibration')
            ax_energy.plot(x_mc, E, label='Production')
            ax_energy.axvline(n_eq * (N ** 2), color='black')
            ax_energy.legend()
            ax_energy.set_ylabel('Energy per site/ J')

            ax_mag.plot(x_eq, M_eq, label='Equilibration')
            ax_mag.plot(x_mc, M, label='Production')
            ax_mag.axvline(n_eq * (N ** 2), color='black')
            ax_mag.legend()
            ax_mag.set_ylabel('Magnetization per site')
            ax_mag.set_xlabel('Number of flip attempts')

            animator(imagelist)
            plt.show()

        plt.plot(range(10, 40, 5), Mind, '.')
        plt.xlabel('System Size Index')
        plt.ylabel('Index of Converged Magnetization (Total Steps)')
        plt.title('Converged Magnetization Step vs. System Size')
        plt.show()
        plt.plot(range(10, 40, 5), Mconv, '.')
        plt.xlabel('System Size Index')
        plt.ylabel('Converged Magnetization')
        plt.title('Converged Magnetization')
        plt.show()
        # plt.plot(range(10, 40, 5), Corrtime)
        # plt.xlabel('System Size Index')
        # plt.ylabel('Correlation Time')
        # plt.show()
        plt.plot(range(10, 40, 5), Mindpersite, '.')
        plt.xlabel('System Size Index')
        plt.ylabel('Index of Converged Magnetization(MC Steps)')
        plt.title('Converged Magnetization Step per Site vs. System Size')
        plt.show()

    elif Q == 4:
        Mavg = []
        Ttest = []
        Mstd = []
        N=40
        for T in np.linspace(1,3,51):
            n_eq = 800
            n_mc = 500
            Ttest.append(T)
            print(T)
            [E, M, E_eq, M_eq, x_eq, x_mc, imagelist] = run_mcmcising(N=N, n_eq=n_eq, n_mc=n_mc, T=T)

            Mavg.append(np.mean(M[0:len(M)]))
            Mstd.append(np.std(M[0:len(M)]))
        X = []
        print(Mavg)
        for i in range(0, len(Mstd)):
            X.append((Mstd[i]**2 / Ttest[i]))

        plt.plot(Ttest, X)
        plt.xlabel('Temperature')
        plt.ylabel('Susceptibility')
        plt.title('Susceptibility')
        plt.show()

        plt.plot(Ttest, Mavg)
        plt.xlabel('Temperature')
        plt.ylabel('Average Magnetization')
        plt.title('Average Magnetization vs. Temperature')
        plt.show()

```

The second code is adapted from the provided Jupyter Notebook for nudged elastic band calculations. The pseudopotential files for EAM and DFT are first specified. Note here that the PP for DFT is an adapted version of the aluminum pseudopotential returned as an XML file from the PSlibrary. This was modified by removing the CDATA sections which interfered with the PyCharm compiler. Two aluminum 111 FCC slabs are constructed with Al adatoms on HCP and FCC sites, respectively. Both are associated with the EAM calculator and relaxed via BFGS. Images are interpolated using the NEB functions to construct the transition path, which is subsequently relaxed using BFGS with the elastic forces between adjacent images. From the relaxed configurations, the energy difference is obtained as the difference in the energy of each image from the first image, and the forces are recorded for comparison with later methods. The DFT calculation is set up using the Espresso calculator with specific parameters and including smearing to account for the metallic nature of Al. For each image obtained from relaxing the EAM NEB path, the energy and forces are obtained again after setting the calculator to Quantum Espresso. The energy difference vs. image number plotted is reconstructed, and the forces on each atom are compared between the two methods. A two-plus-three body Gaussian Process model is then trained by constructing a training set composed of all of the relaxed images from EAM and the forces obtained from DFT. The calculator for each image is then set to the trained model, and the energy difference vs. image plotted is reconstructed again. Forces are obtained and compared with both DFT (the training forces) and EAM.

```
from ase.calculators.eam import EAM
from ase.calculators.espresso import Espresso
import os
from ase.build import fcc111, add_adsorbate
from ase.visualize import view
from ase.optimize import BFGS
from ase.neb import NEB
import numpy as np
import matplotlib.pyplot as plt
import kernels
import gp
import struc
from gp_calculator import GPCalculator

###EAM Setup, Optimization, and Plotting
 pot_file = os.environ.get('LAMMPS_POTENTIALS') + '/Al_zhou.eam.alloy'
 pot_file2 = 'Al.pbe-n-rrkjus_psl.1.0.1.UPF'
 pseudopotentials = {'Al': pot_file2}
zhou = EAM(potential=pot_file)
slabEAM = fcc111('Al', size=(2, 2, 3))
add_adsorbate(slabEAM, 'Al', 2, 'hcp')
slabEAM.center(vacuum=5.0, axis=2)
slabEAM.set_calculator(zhou)
print(slabEAM.get_potential_energy())
dyn = BFGS(slabEAM)
dyn.run(fmax=0.0001)

slab_2EAM = fcc111('Al', size=(2, 2, 3))
```

```
add_adsorbate(slab_2EAM, 'Al', 2, 'fcc')
slab_2EAM.center(vacuum=5.0, axis=2)
slab_2EAM.set_calculator(EAM(potential=pot_file))
dyn = BFGS(slab_2EAM)
slab_2EAM.get_potential_energy()
print(slab_2EAM.get_potential_energy())
dyn.run(fmax=0.0001)

no_images = 7
imagesEAM = [slabEAM]    #first image is the first slab (endpoint)
imagesEAM += [slabEAM.copy() for i in range(no_images-2)]    #copy first slab for
    intermediate images
imagesEAM += [slab_2EAM]    #final image is the second slab (endpoint 2)
nebEAM = NEB(imagesEAM)
nebEAM.interpolate()
pot_dir = os.environ.get('LAMMPS_POTENTIALS')
for image in imagesEAM[1:no_images-1]:
    image.set_calculator(EAM(potential=pot_file))
optimizer = BFGS(nebEAM)
optimizer.run(fmax=0.01)
pes = np.zeros(no_images)
pos = np.zeros((no_images, len(imagesEAM[0]), 3))
EAMForces = np.zeros((no_images,len(imagesEAM[0]),3))
for n, image in enumerate(imagesEAM):
    pes[n] = image.get_potential_energy()
    pos[n] = image.positions
    EAMForces[n] = image.get_forces()
plt.plot(pes-pes[0], 'k.', markersize=10)  # plot energy difference in eV w.r.t. first image
plt.plot(pes-pes[0], 'k--', markersize=10)
plt.xlabel('image #')
plt.ylabel('energy difference (eV)')
plt.title('EAM')
plt.show()

###DFT Setup, Optimization, and Plotting
pseudopotentials = {'Al': pot_file2}
input_data = {
    'system': {
        'ecutwfc': 29,
        'ecutrho': 143,
        'occupations': 'smearing',
        'smearing': 'mp',
        'degauss': 0.02
    },
    'disk_io': 'low'}
calc = Espresso(pseudopotentials = pseudopotentials, tstress = True, tprnfor = True, kpts =
    (4, 4, 1), input_data = input_data)

for image in imagesEAM:
    image.set_calculator(Espresso(pseudopotentials=pseudopotentials, tstress=True, tprnfor=
    True, kpts=(4, 4, 1), input_data=input_data))

print(imagesEAM[len(imagesEAM)-1].get_potential_energy())
pes = np.zeros(no_images)
pos = np.zeros((no_images, len(imagesEAM[0]), 3))
DFTForces = np.zeros((no_images,len(imagesEAM[0]),3))
for n, image in enumerate(imagesEAM):
    pes[n] = image.get_potential_energy()
    pos[n] = image.positions
    DFTForces[n] = image.get_forces()
plt.plot(pes-pes[0], 'k.', markersize=10)
plt.plot(pes-pes[0], 'k--', markersize=10)
plt.xlabel('image #')
plt.ylabel('energy difference (eV)')
plt.title('DFT')
plt.show()

plt.plot(DFTForces.reshape(-1),EAMForces.reshape(-1), '.')
```

```python
plt.xlabel('DFT Force')
plt.ylabel('EAM Force')
plt.show()

print(np.mean(abs(DFTForces-EAMForces)))

###GP Setup, Optimization, and Plotting

kernel = kernels.two_plus_three_body
kernel_grad = kernels.two_plus_three_body_grad
hyps = np.array([1, 1, 0.1, 1, 1e-3]) # sig2, ls2, sig3, ls3, noise std
cutoffs = np.array([4.9, 4.9]) # (don't need to optimize for lab)
energy_force_kernel = kernels.two_plus_three_force_en
gp_model = gp.GaussianProcess(kernel, kernel_grad, hyps, cutoffs,
                              energy_force_kernel=energy_force_kernel)
for image in imagesEAM:
    training_struc = struc.Structure(cell = image.cell,
                                     species=['Al'] * len(image),
                                     positions=image.positions)
    training_forces = image.get_forces()
    gp_model.update_db(training_struc, training_forces)
gp_model.set_L_alpha()

for image in imagesEAM:
    image.set_calculator(GPCalculator(gp_model))

pes = np.zeros(no_images)
pos = np.zeros((no_images, len(imagesEAM[0]), 3))
GPForces = np.zeros((no_images,len(imagesEAM[0]),3))
for n, image in enumerate(imagesEAM):
    pes[n] = image.get_potential_energy()
    pos[n] = image.positions
    GPForces[n] = image.get_forces()

plt.plot(pes-pes[0], 'k.', markersize=10)  # plot energy difference in eV w.r.t. first image
plt.plot(pes-pes[0], 'k--', markersize=10)
plt.xlabel('image #')
plt.ylabel('energy difference (eV)')
plt.title('GP')
plt.show()

plt.plot(GPForces.reshape(-1),EAMForces.reshape(-1), '.')
plt.xlabel('GP Force')
plt.ylabel('EAM Force')
plt.show()

plt.plot(DFTForces.reshape(-1),GPForces.reshape(-1), '.')
plt.xlabel('DFT Force')
plt.ylabel('GP Force')
plt.show()

```