Quantifying Spin Interactions Using Reinforcement Learning

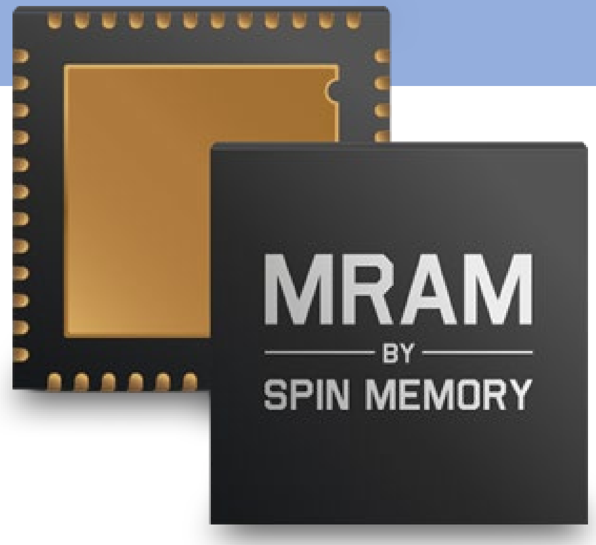
Jessica Opsahl-Ong

Collaborators:
Kate Meuse, Paul Kienzle, William Ratcliff

Magnetism

Many fields with current interest in magnetic structures

- Hard drives, MRAM, quantum computing, etc.



In order to utilize magnetic properties, we must be able to identify the specific interactions that give way to magnetism

Magnetic Properties

The magnetic spin ground state is determined by the minimization of its total energy (represented by the Heisenberg-Hamiltonian)

$$H = -\frac{1}{2}\sum_{i,j} J_{i,j}\mathbf{S}_i \cdot \mathbf{S}_j$$

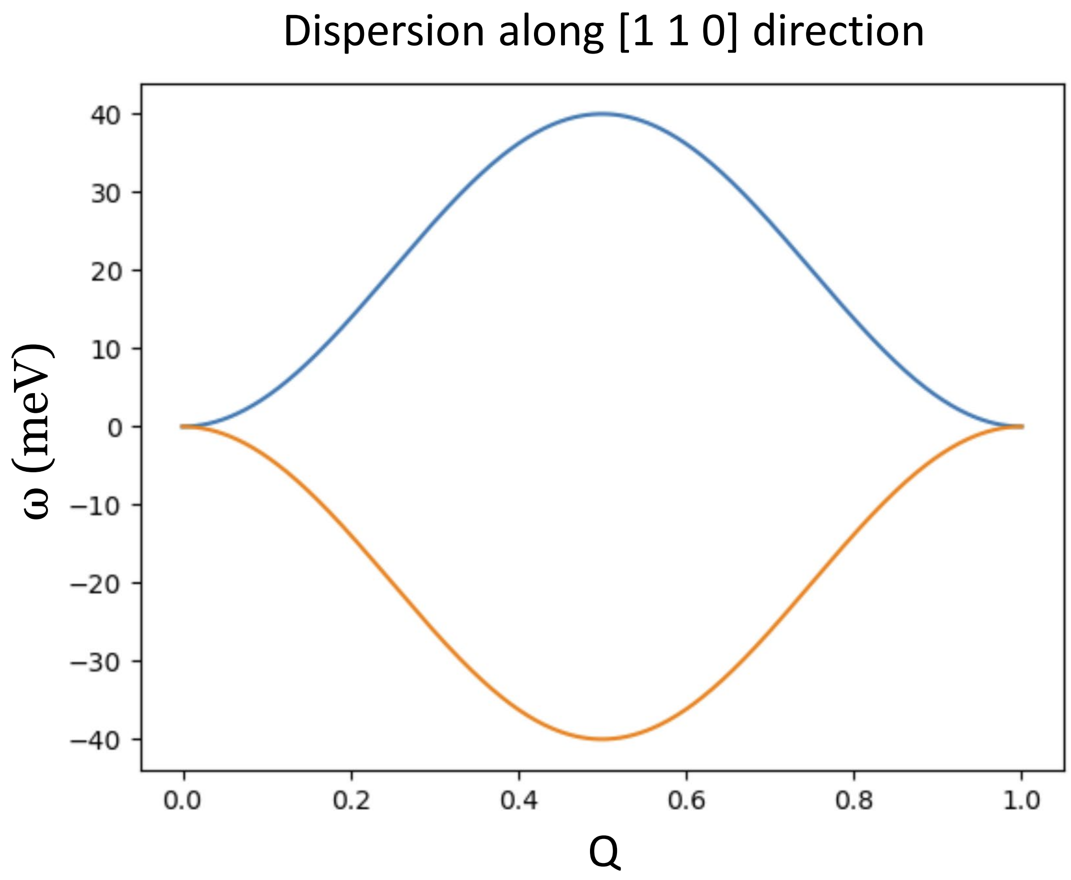
However, knowing the ground state (i.e. the spin vectors) does not uniquely determine the set of interactions (i.e. the coupling constant J) which gave rise to the Hamiltonian

Finding J

- In order to quantify J, we must make a magnetic excitation
  - Simply flipping the direction of one spin is too high-energy
  - Instead, share the single spin reversal among many spins → creates a spin wave → measure the energy of this wave
  - Create and measure spinwave using inelastic neutron scattering

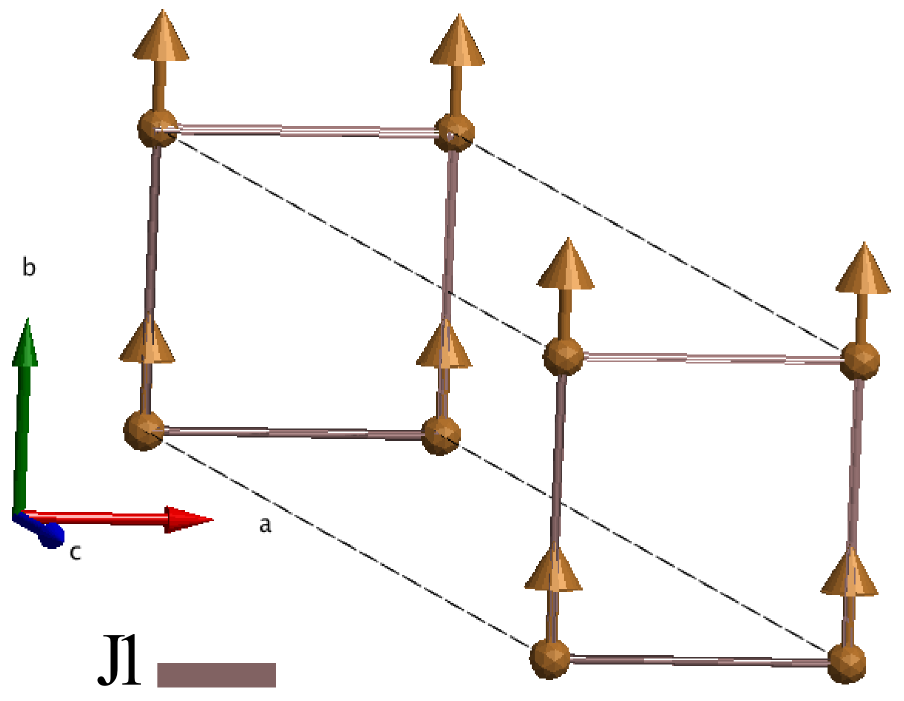
Simulating Spinwaves with SpinW

- SpinW is a MATLAB library that can numerically simulate magnetic structures and their spinwave dispersions
- We used pySpinW which binds SpinW to Python
- Tested how accurate this interface is by comparing pySpinW and SpinW results

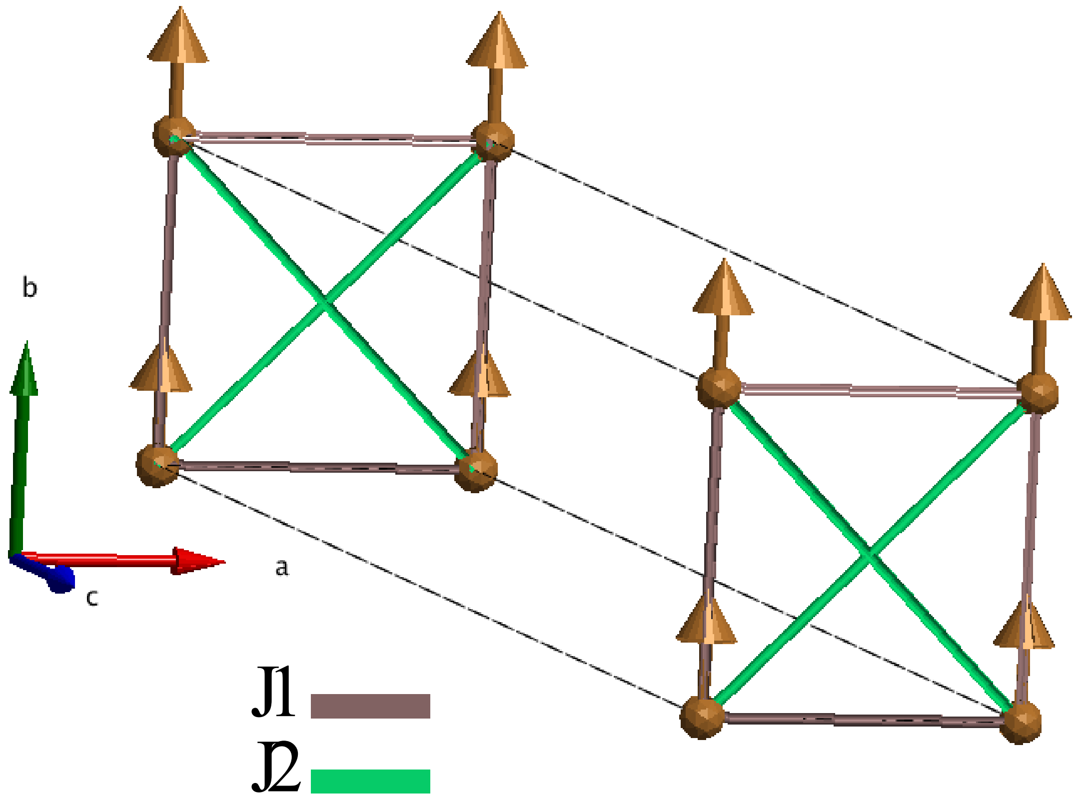


Square Lattice Ferromagnet

Nearest neighbor model



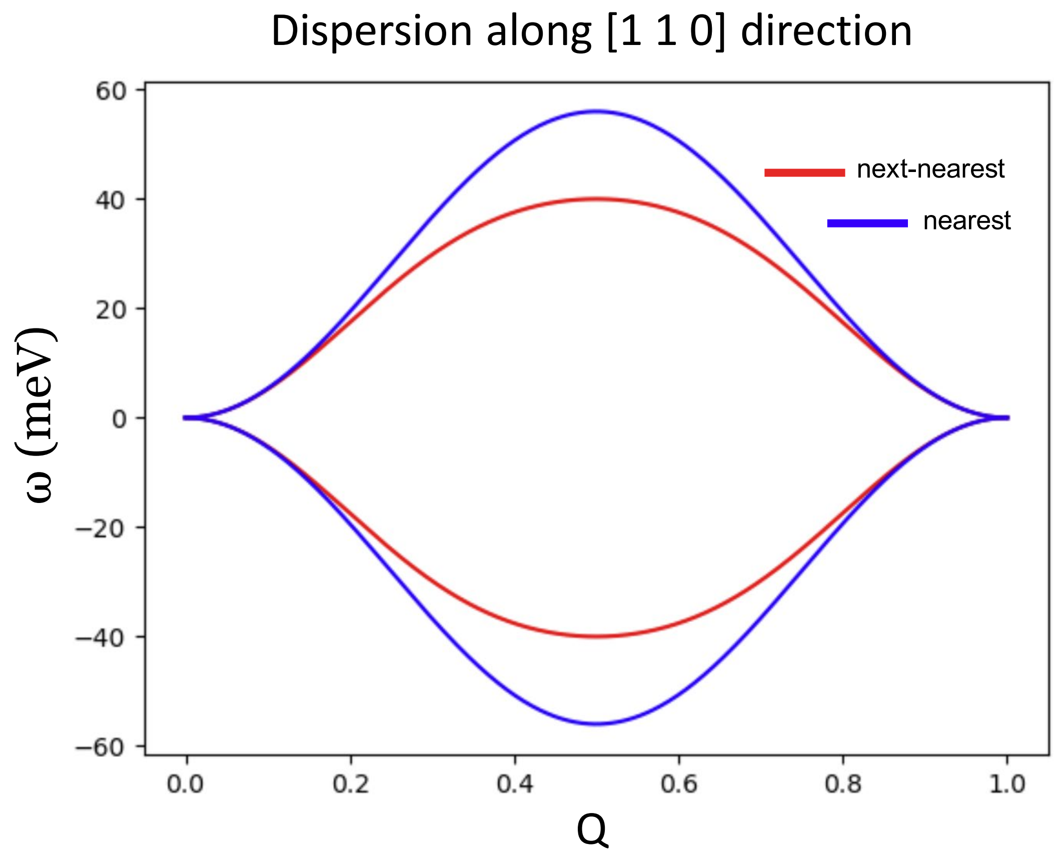
Next-nearest neighbor model



Distinguishing Models

- Can fit for J1 and J2 using BUMPS
- Use Bayesian Information Criterion (BIC) to distinguish between models
  - Based on fit + number of parameters

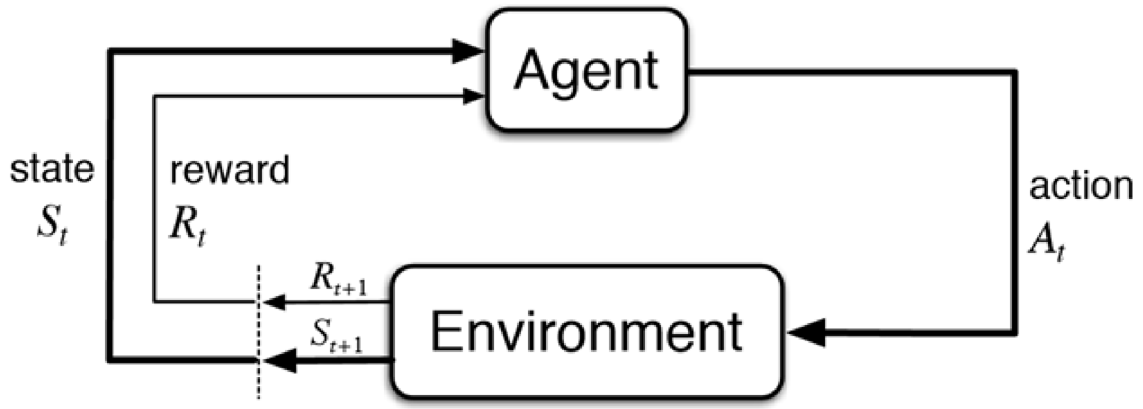
Question: what are the most informative measurements to fit for Js and distinguish between models?



Reinforcement Learning

Defined:

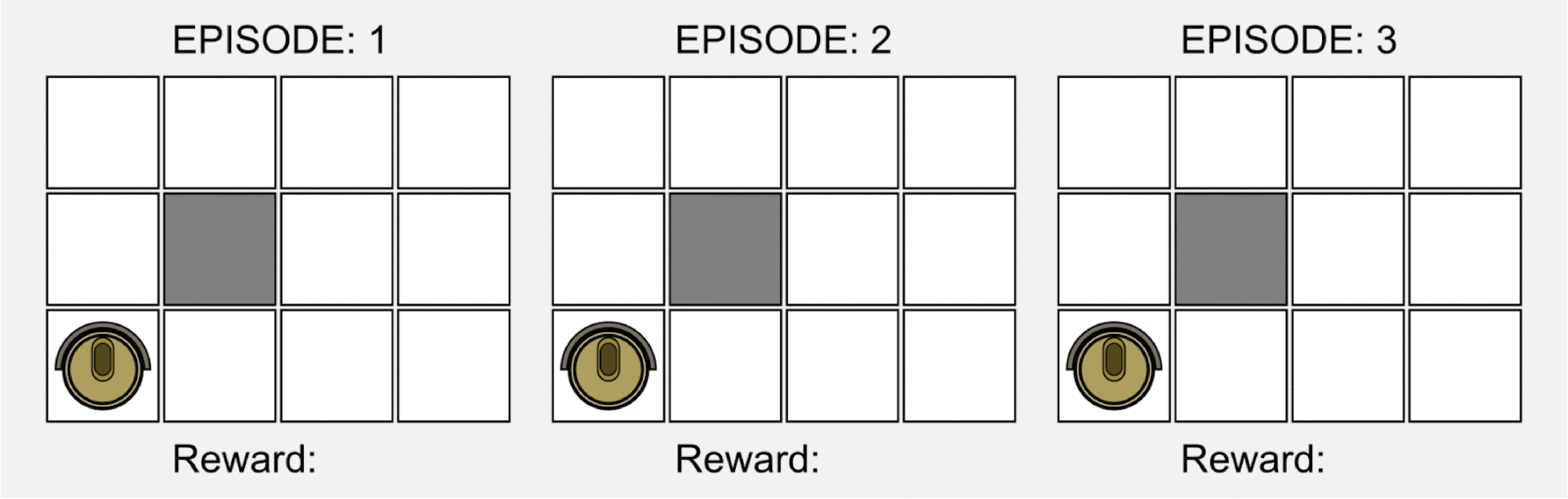
Teaching a computer to make optimal decisions using rewards



How does it work?

1. The agent is in an environment
2. The environment returns a state
3. Agent makes action based on state
4. Agent is rewarded after action
5. Algorithm learns how to best make actions based on rewards

Reinforcement Learning



https://mpatacchiola.github.io/blog/2017/01/15/dissecting-reinforcement-learning-2.html

Applying Reinforcement Learning

- Action: choosing measurement
- State: all measurements chosen thus far
- Ends episode when chi squared & uncertainty is low

Applying Reinforcement Learning

- Action: choosing measurement
- State: all measurements chosen thus far
- Ends episode when chi squared & uncertainty is low

----------Reward function --

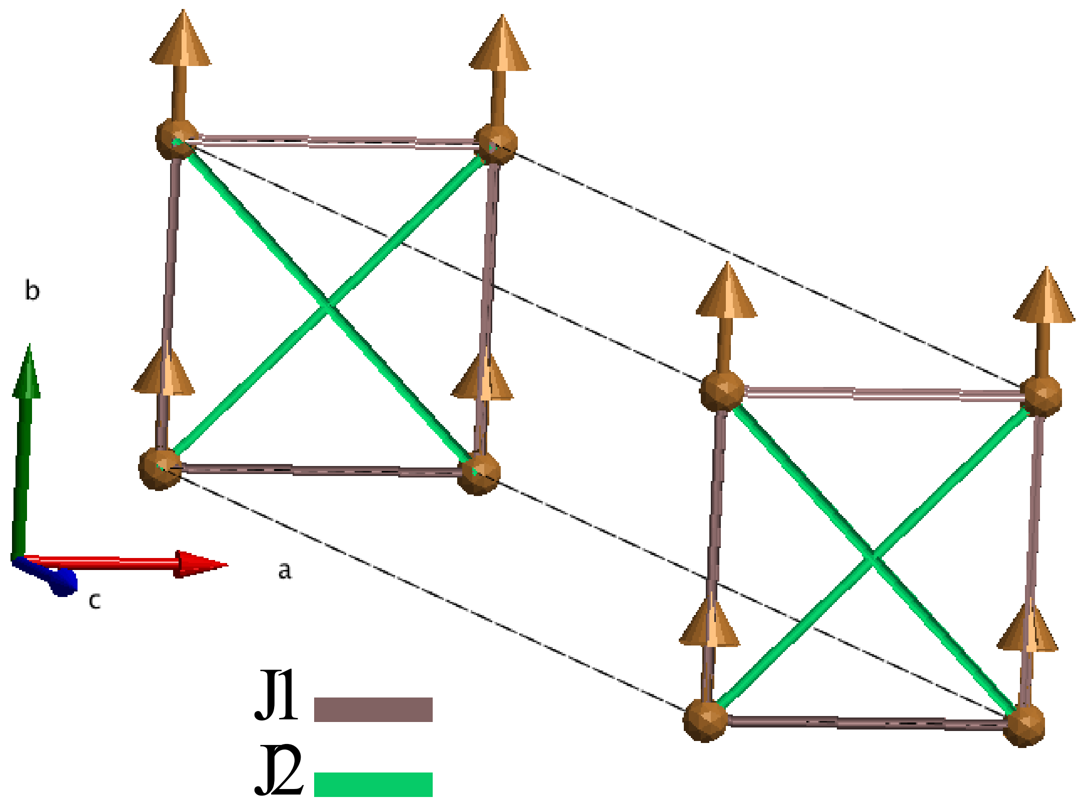
-100 per measurement taken

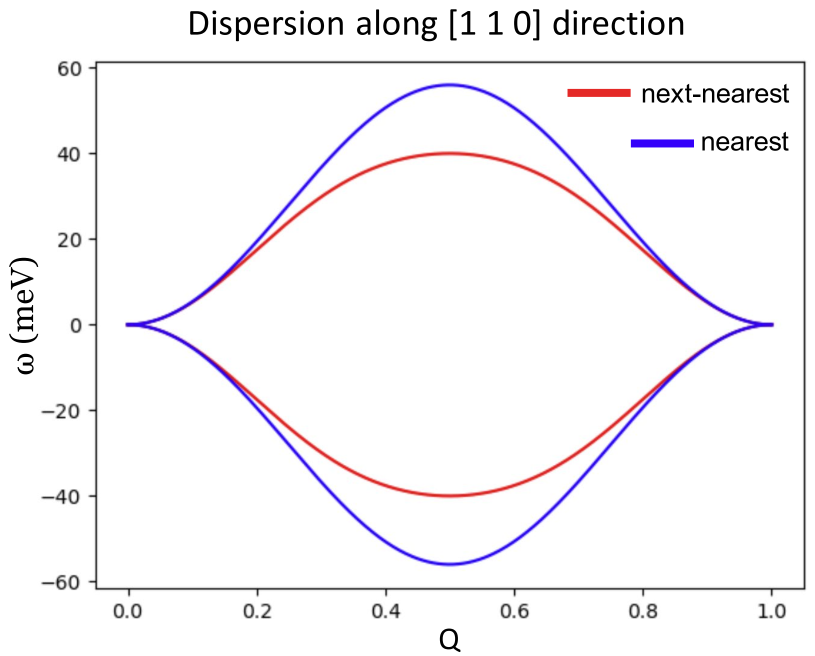
+150 when BIC difference > 10, otherwise 10 * (BIC difference)

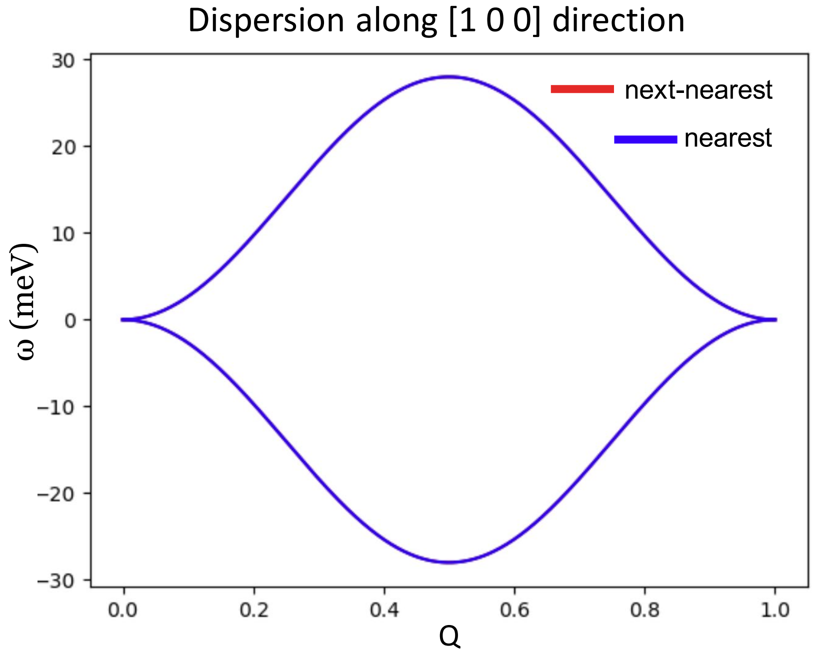
+50 when chi-squared < 1 and uncertainty < 100

Run

Next-nearest neighbor model

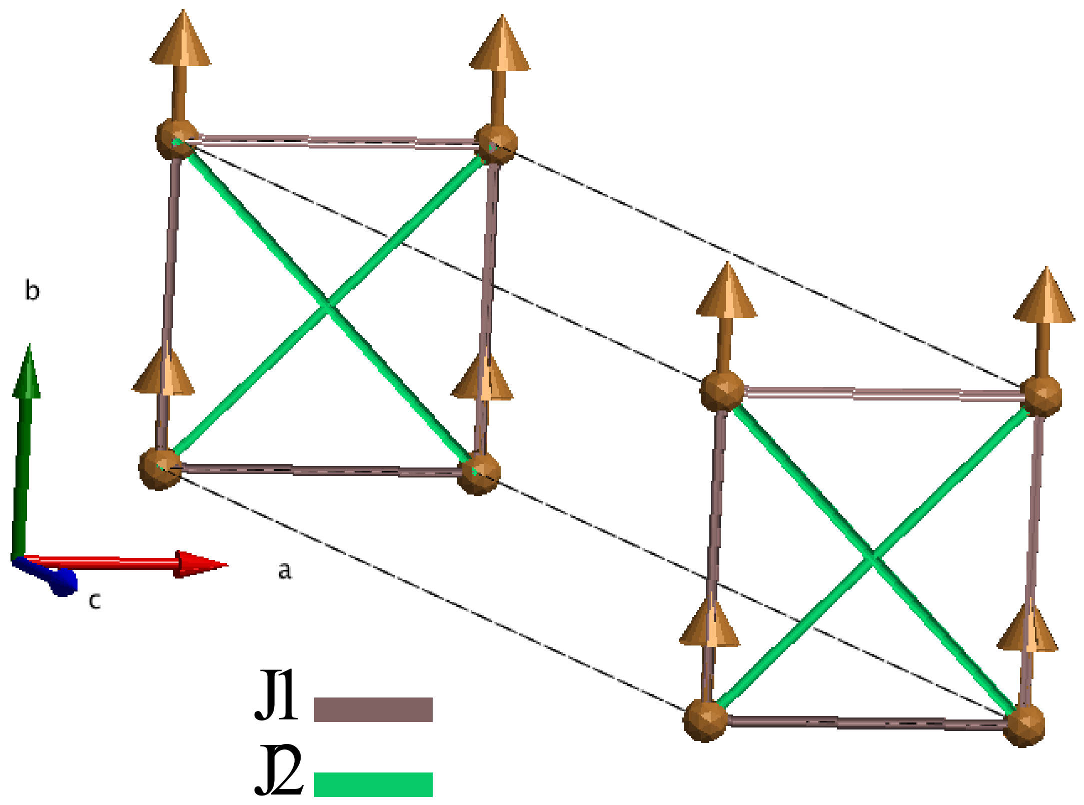


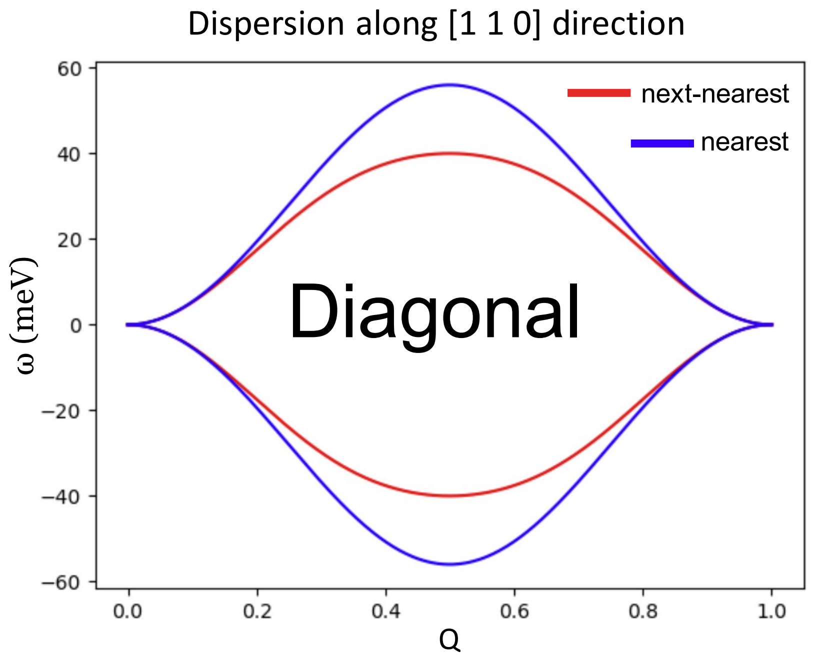


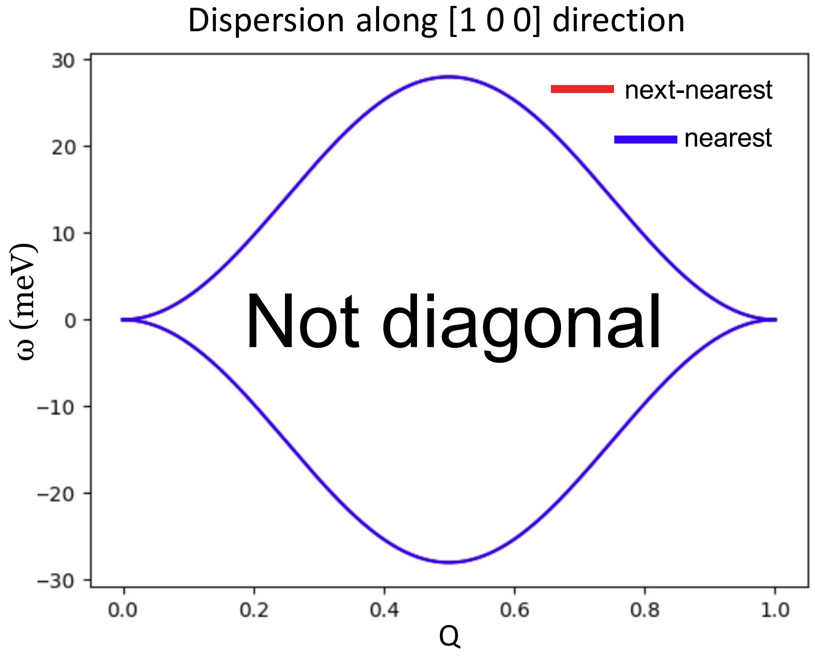


Run

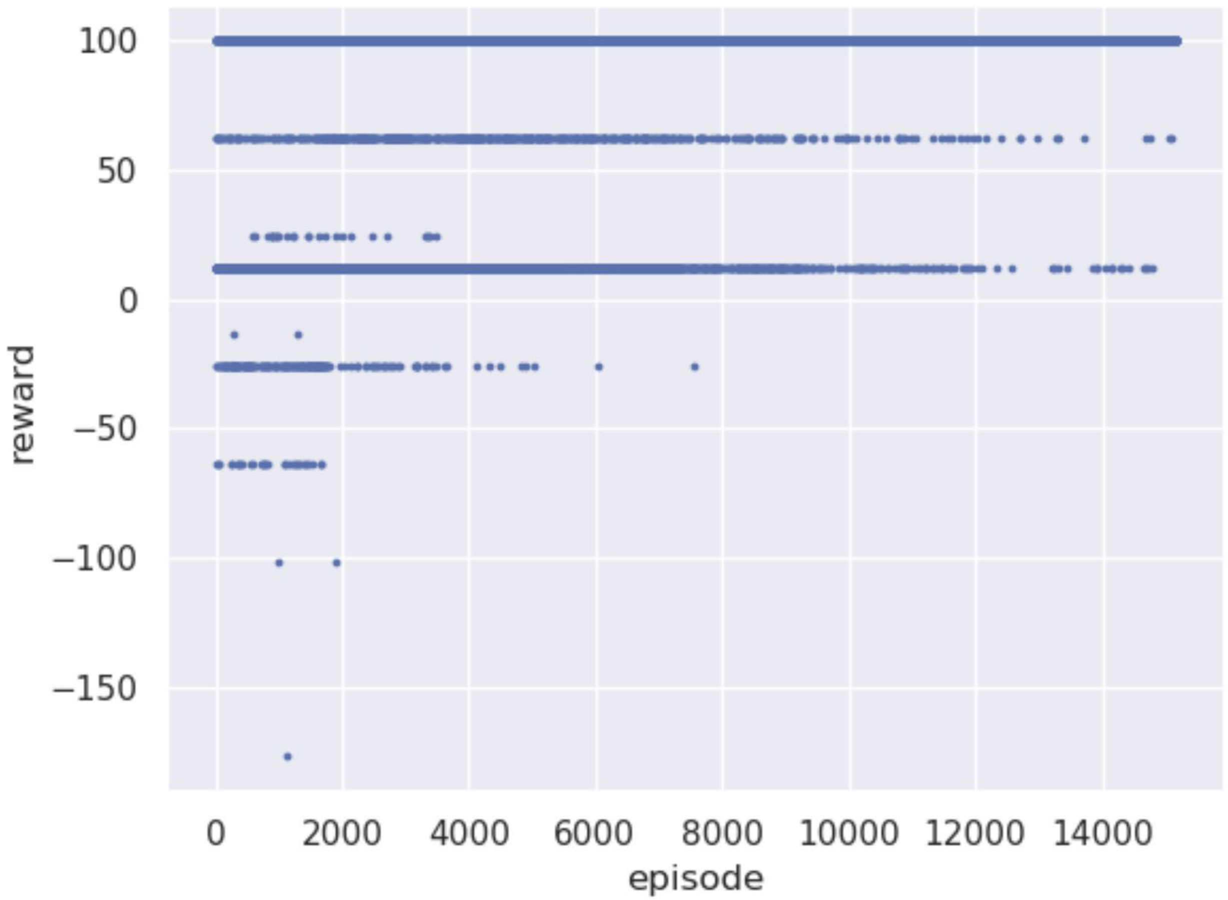
Next-nearest neighbor model







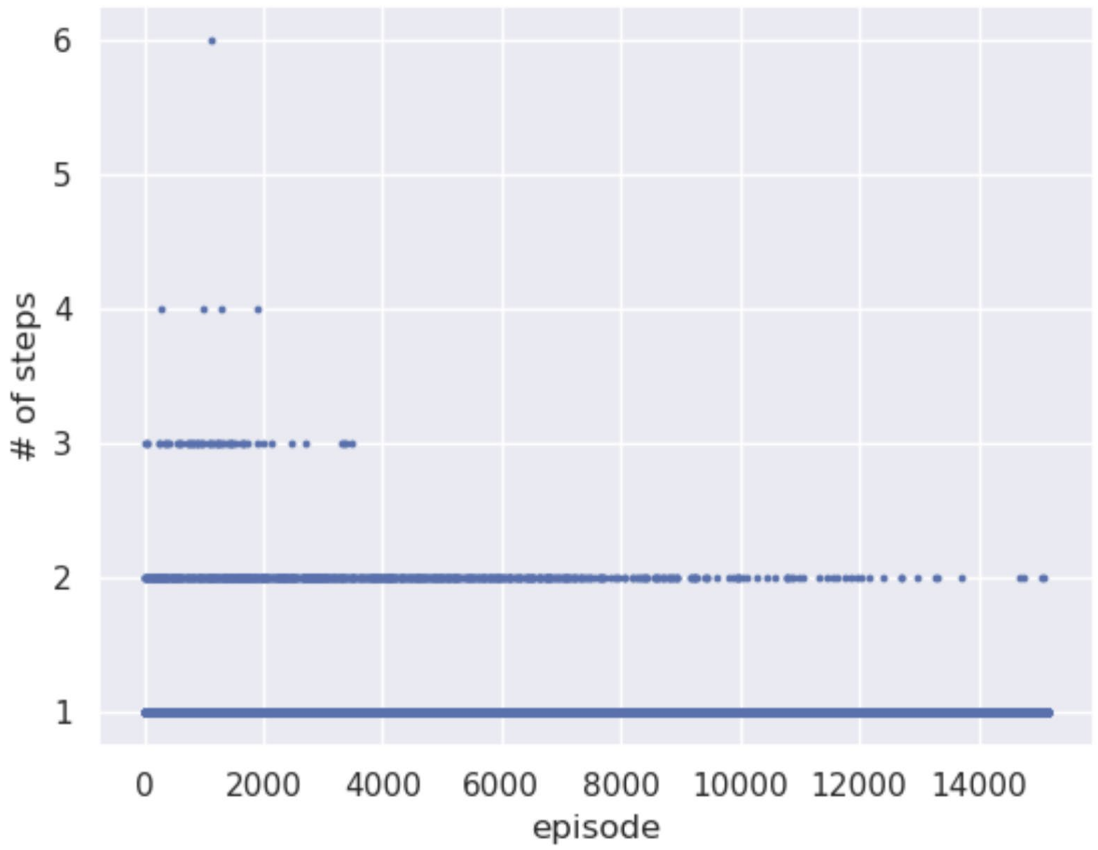
Results - Rewards



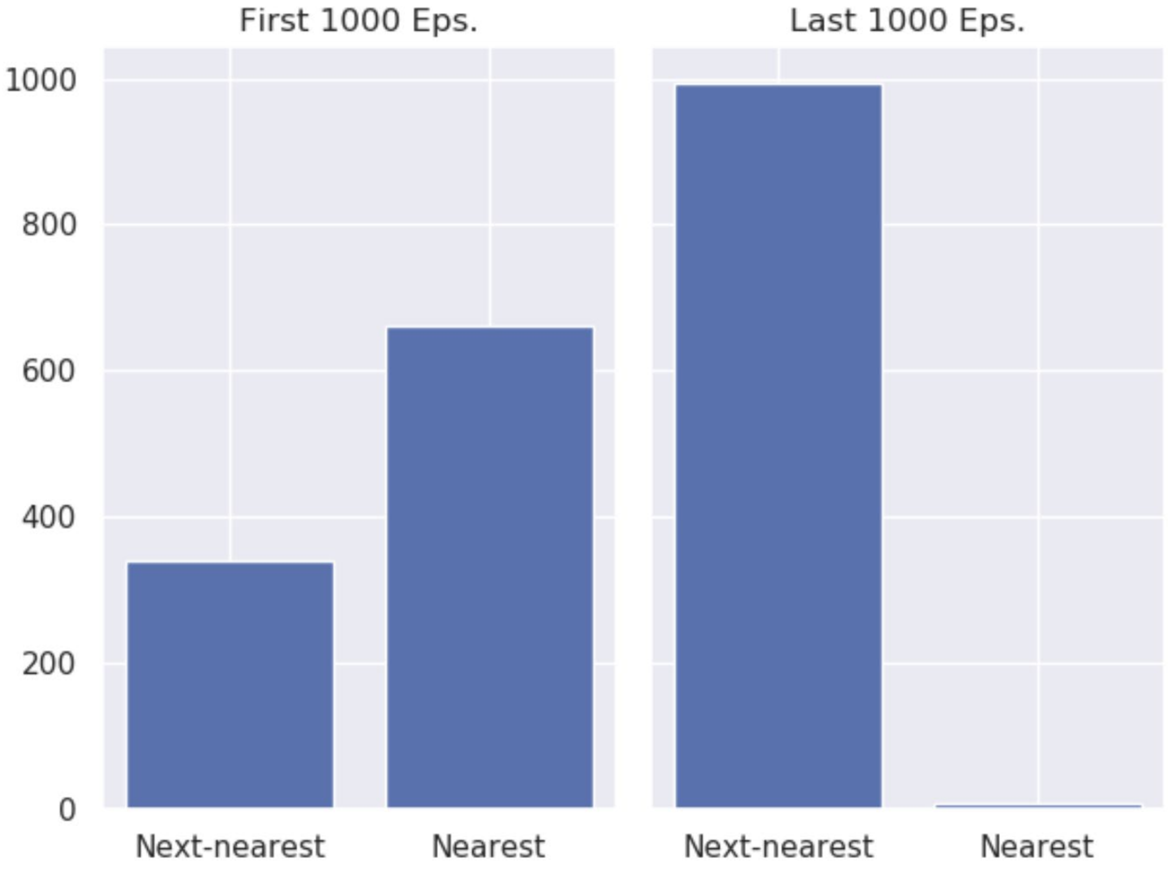
- Run with real model as next-nearest neighbor
- General upward trend in rewards – means it's learning!
- Need to run for longer

Results – Overall Goals

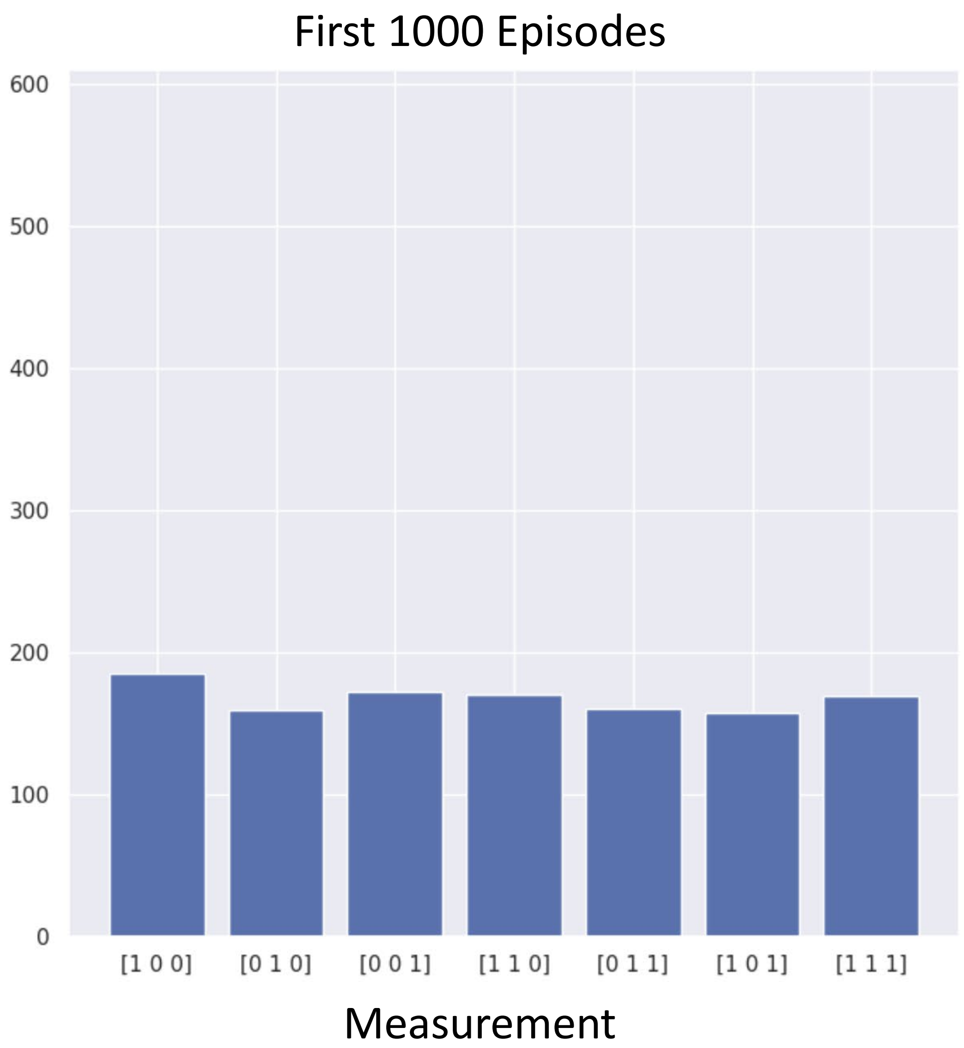
Number of measurements per episode

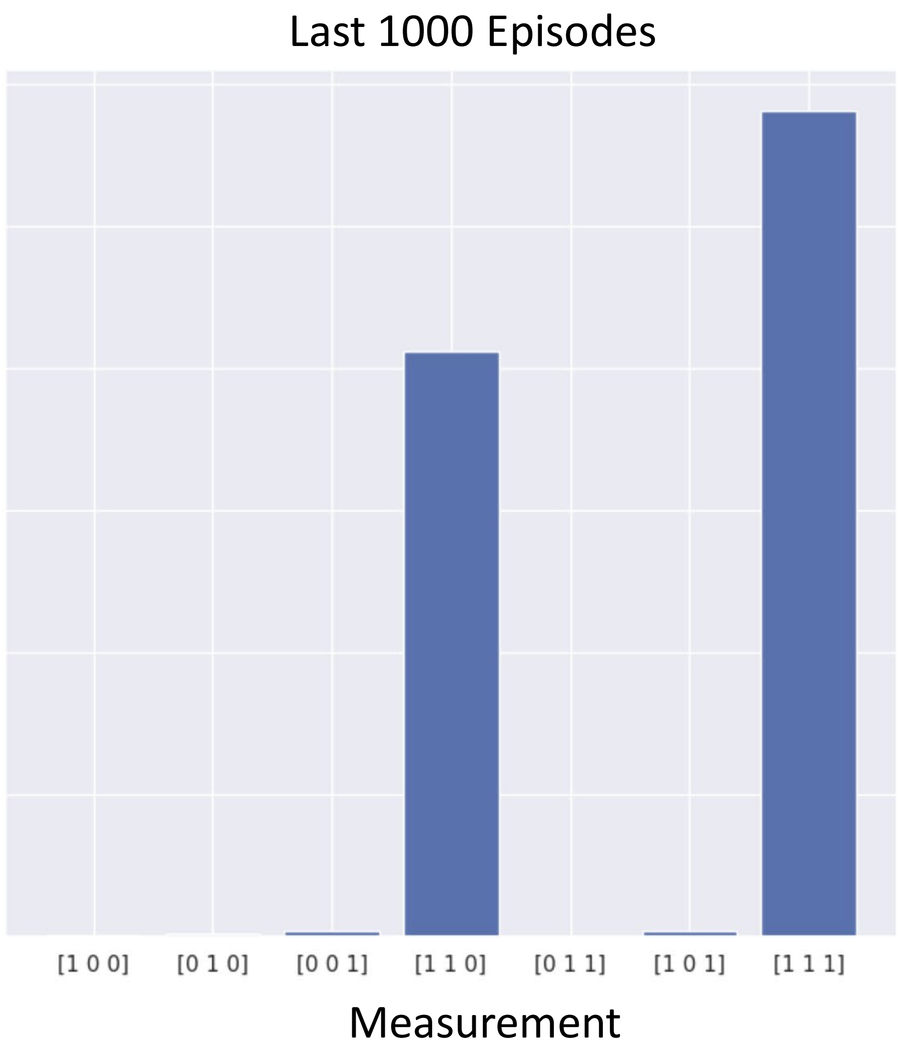


Model chosen at the end of an episode



Results – Measurement Distribution





Next Steps

1. Calculating neutron intensities rather than just the dispersion
   - current difficulty with python bindings to SpinW
2. Including the finite resolution of the instrument
3. Exploring the use of alternative methods
   - Gaussian Processes
4. Publishing

Acknowledgements

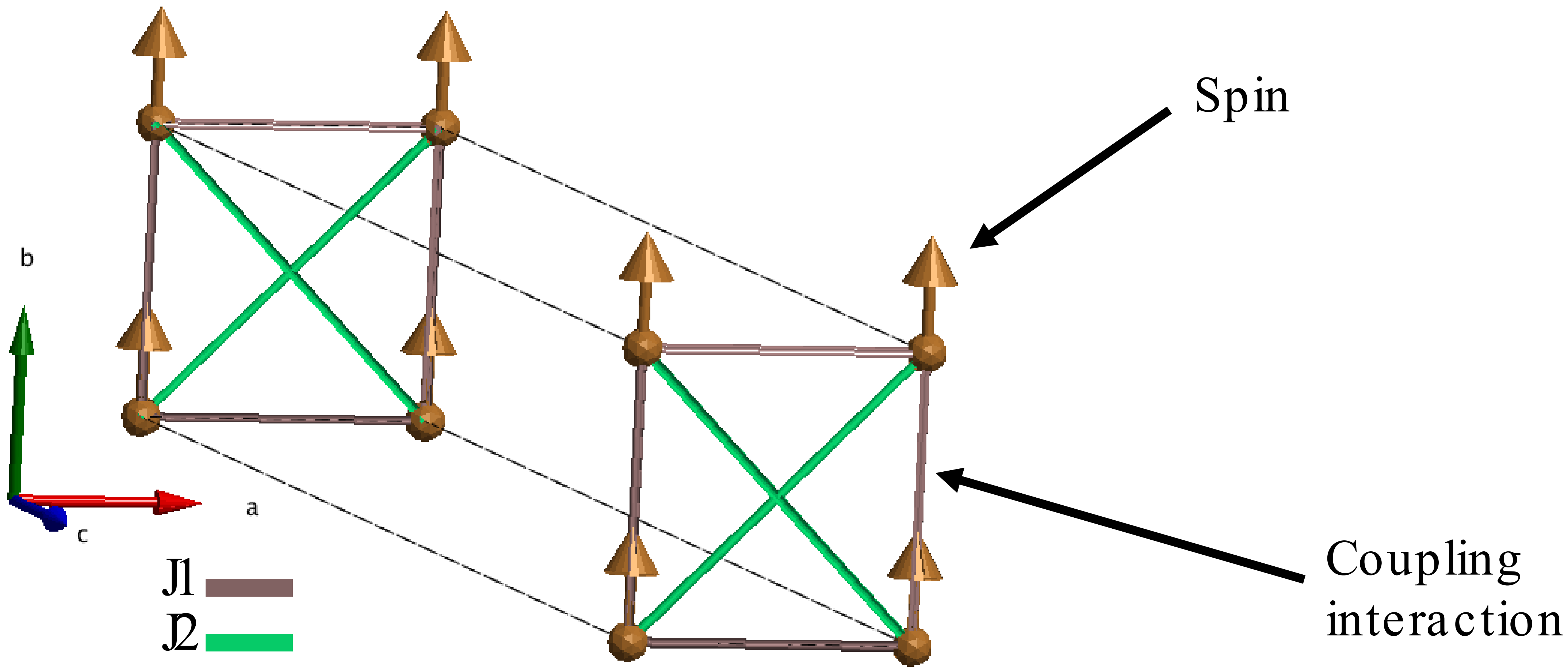
Thanks to the SURF program directors, interns and mentors.

Special thanks to William Ratcliff.

We are grateful for funding from the Center for High Resolution Neutron Scattering.

Any Questions?

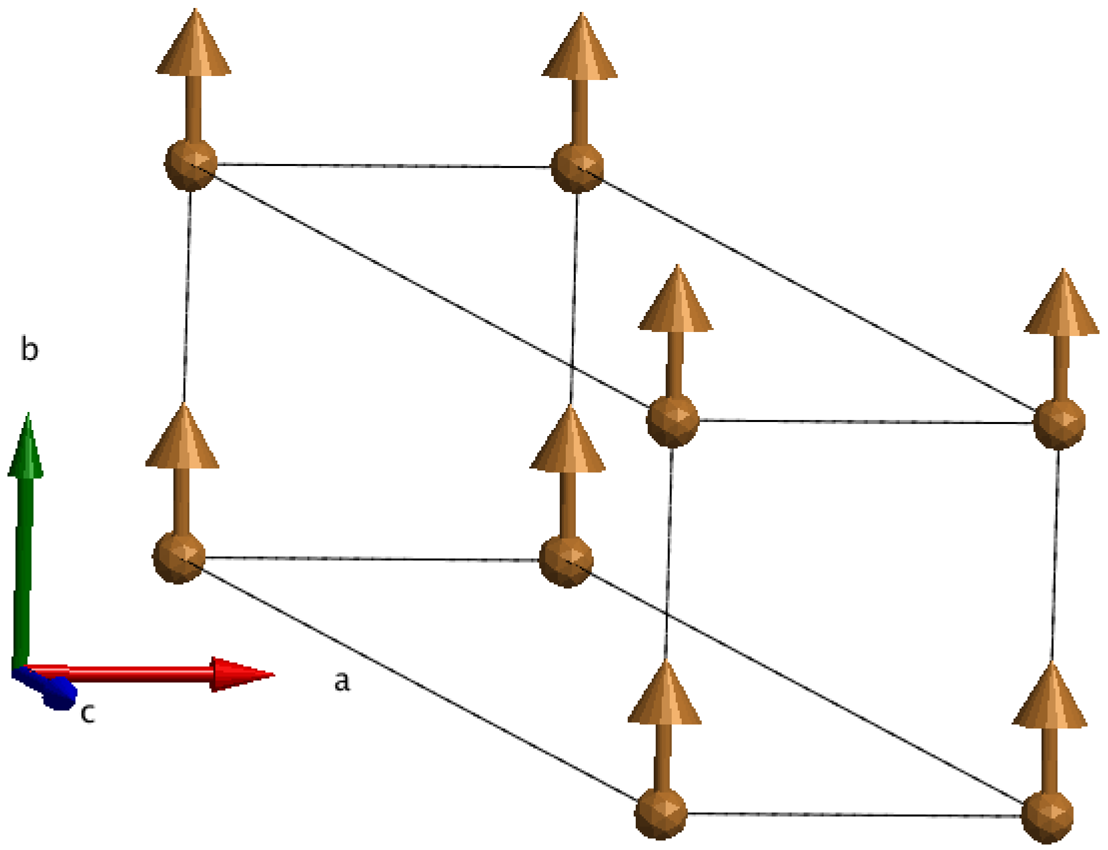
Magnet Lattice Example



Goals & Impact

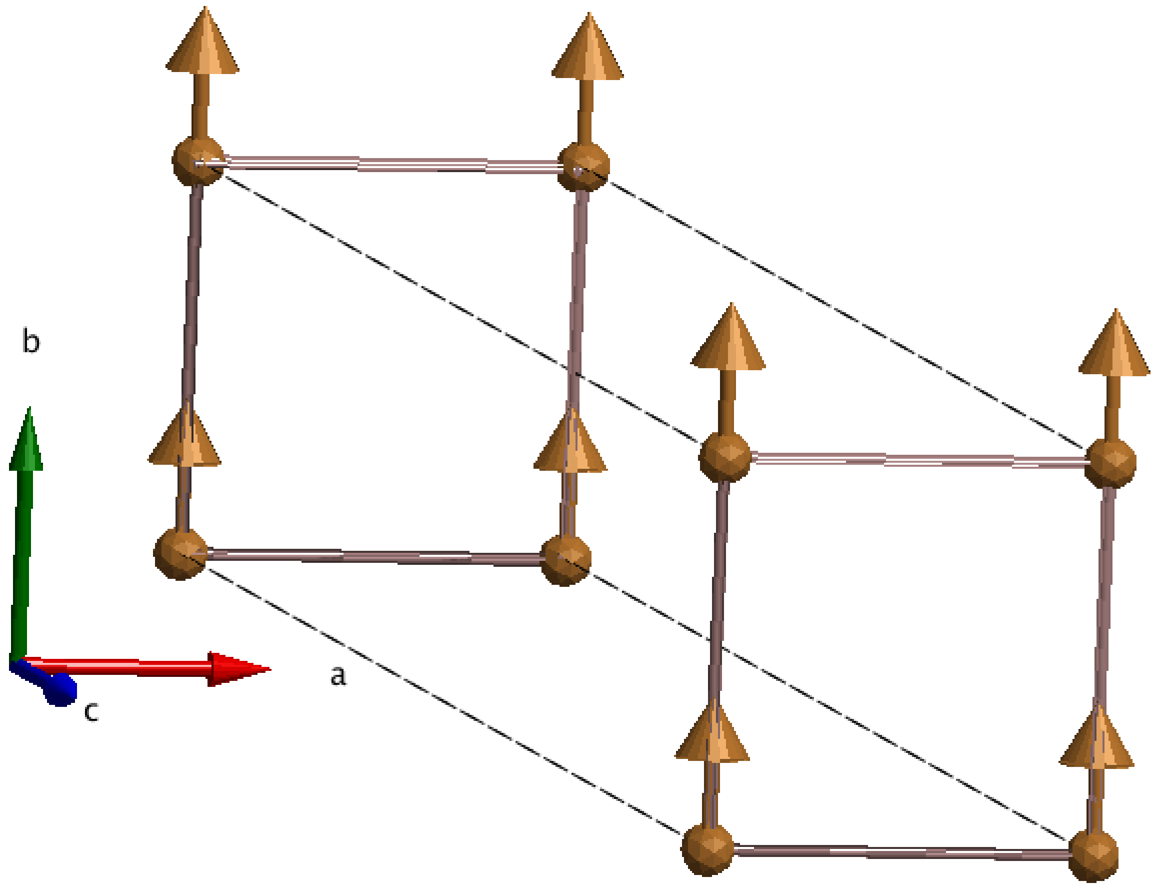
- Beam time is valuable – limited access
- Want more efficient measurements as not all are required
- Software to be implemented on instruments using NICE

Magnetic Crystal Structures



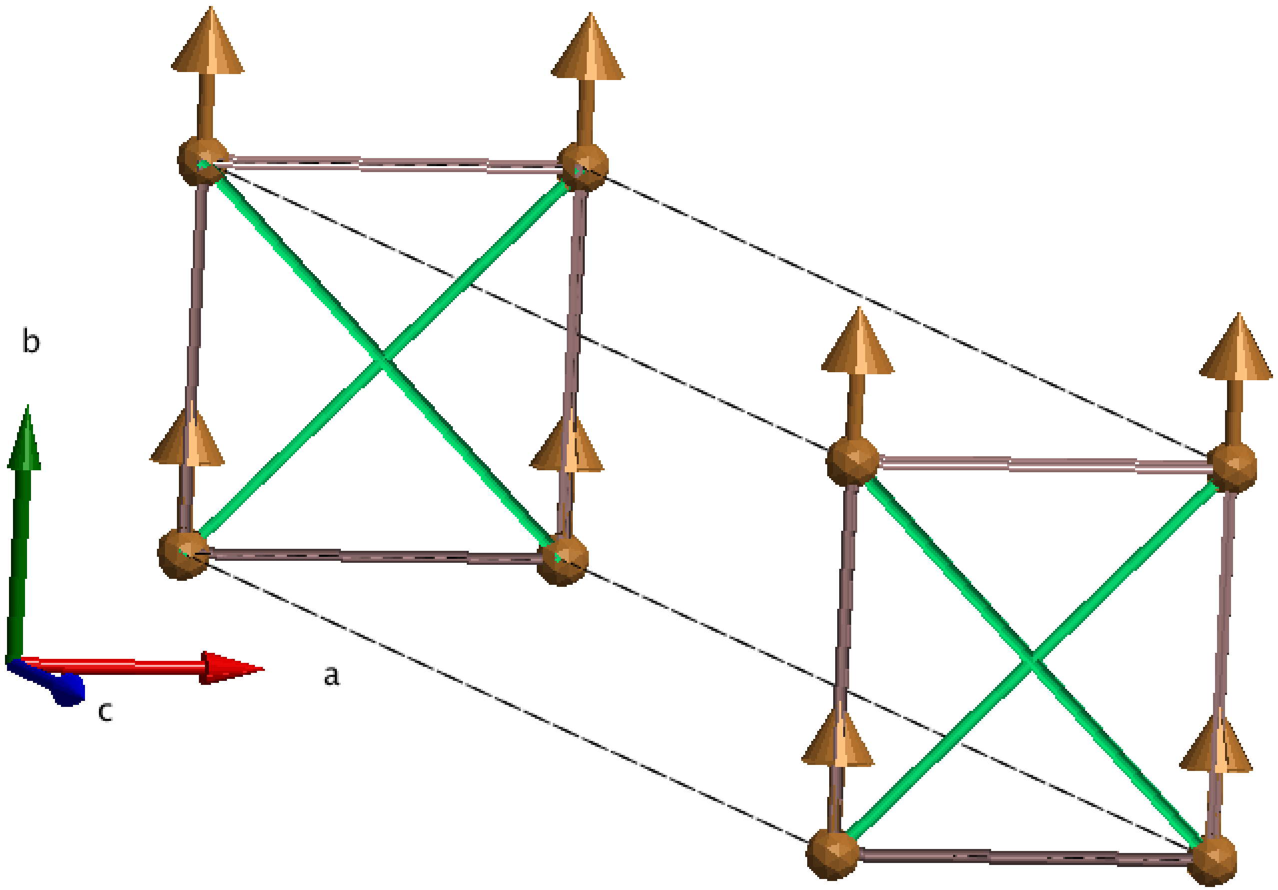
Square lattice structure

Magnetic Crystal Structures



Nearest neighbor model

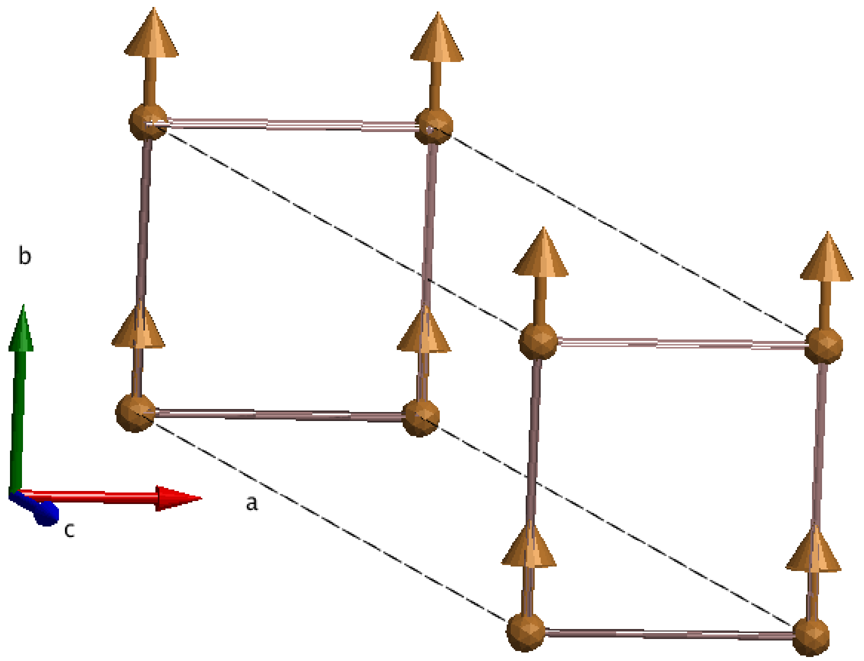
Magnetic Crystal Structures



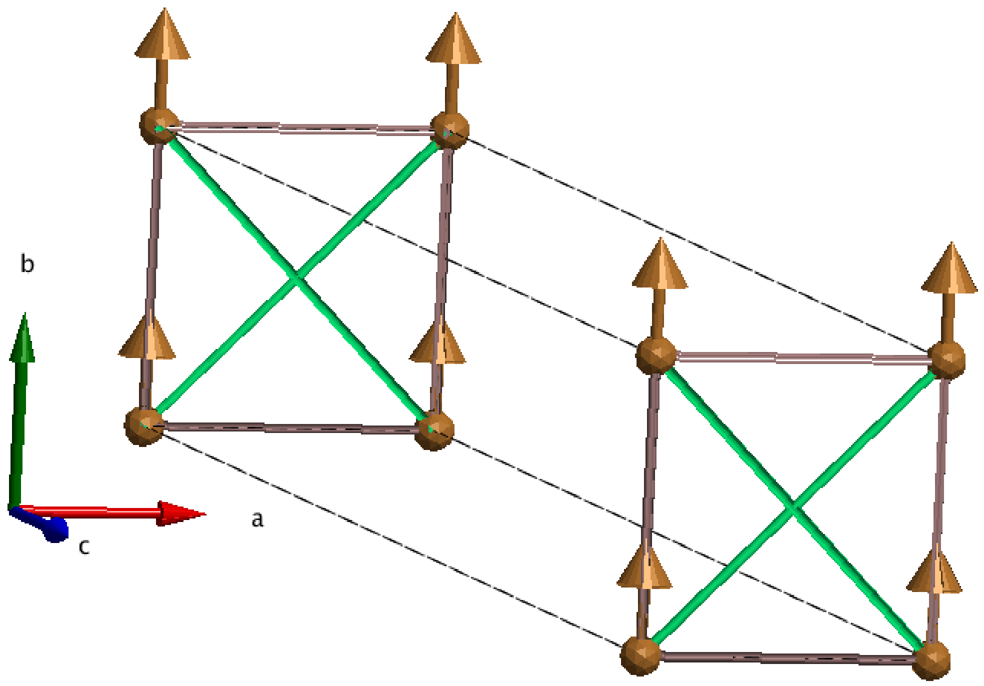
Next-nearest neighbor model

Key Parameters

J1 – coupling constant for nearest neighbor interactions

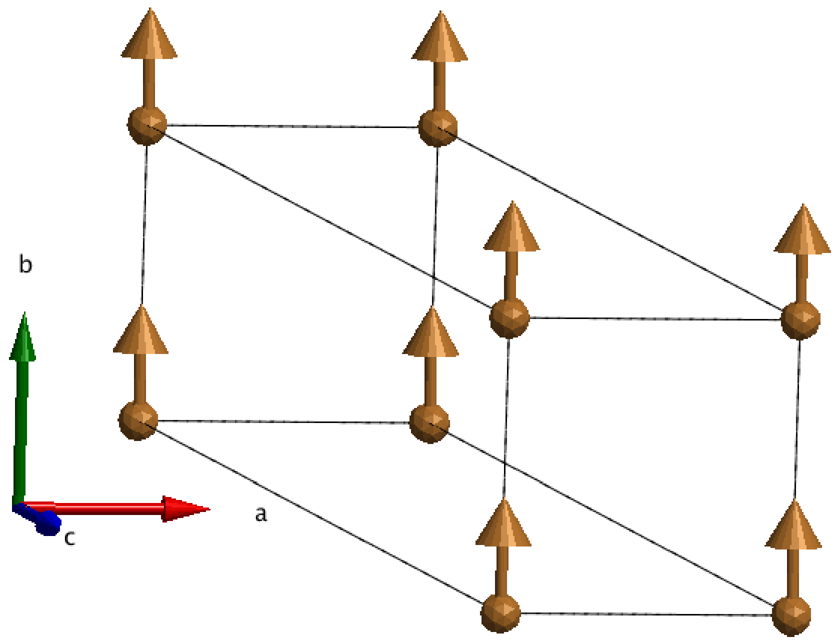


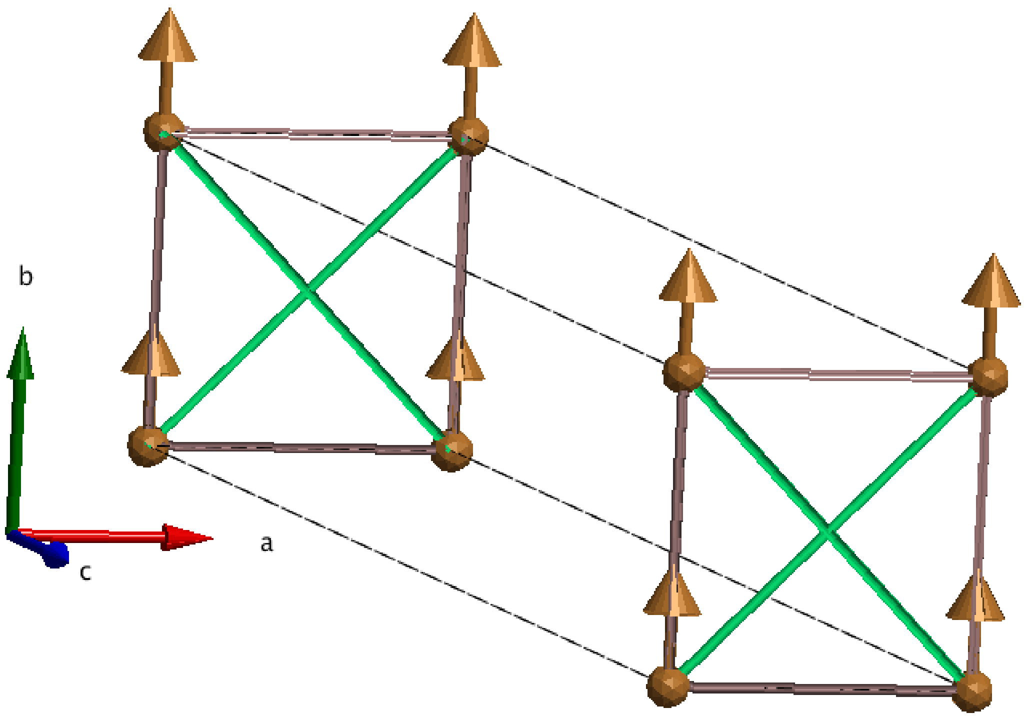
J2 – coupling constant for next-nearest neighbor interactions



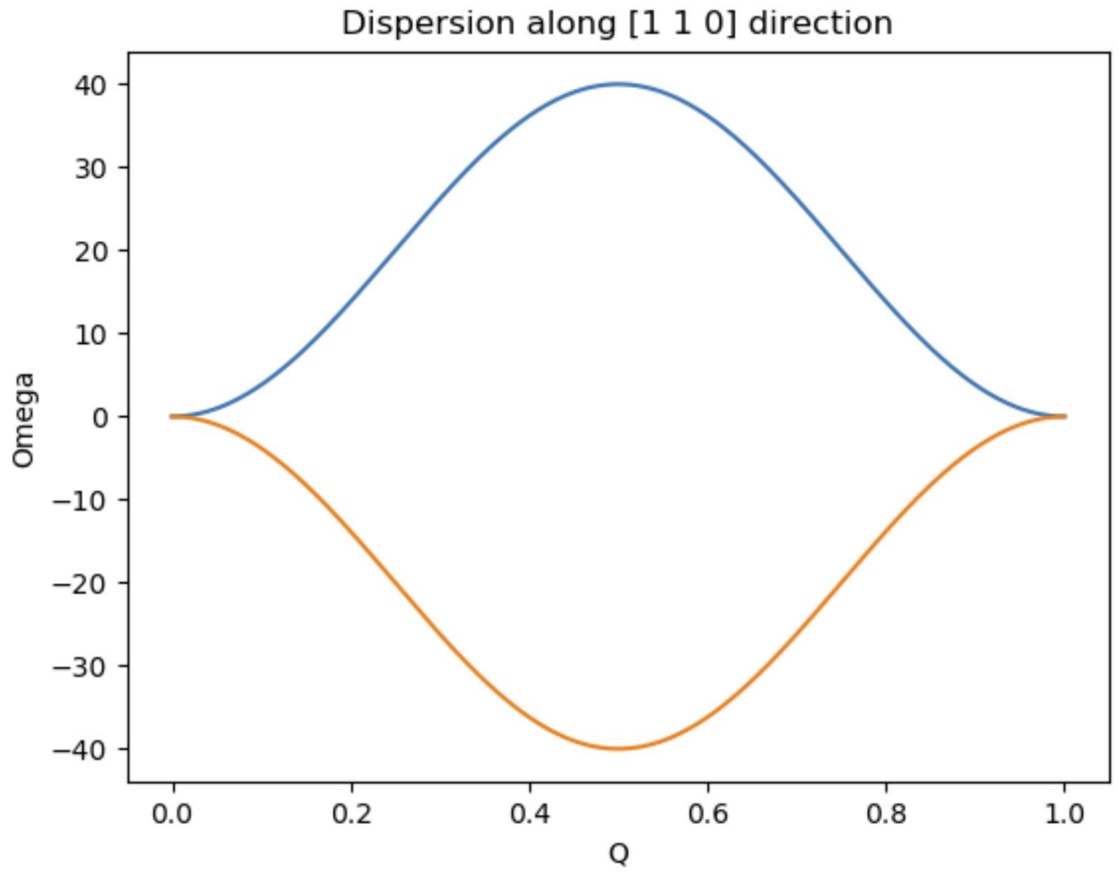
Defining the Problem

You know the basic structure → how do you know which interactions there are & their coupling constants?

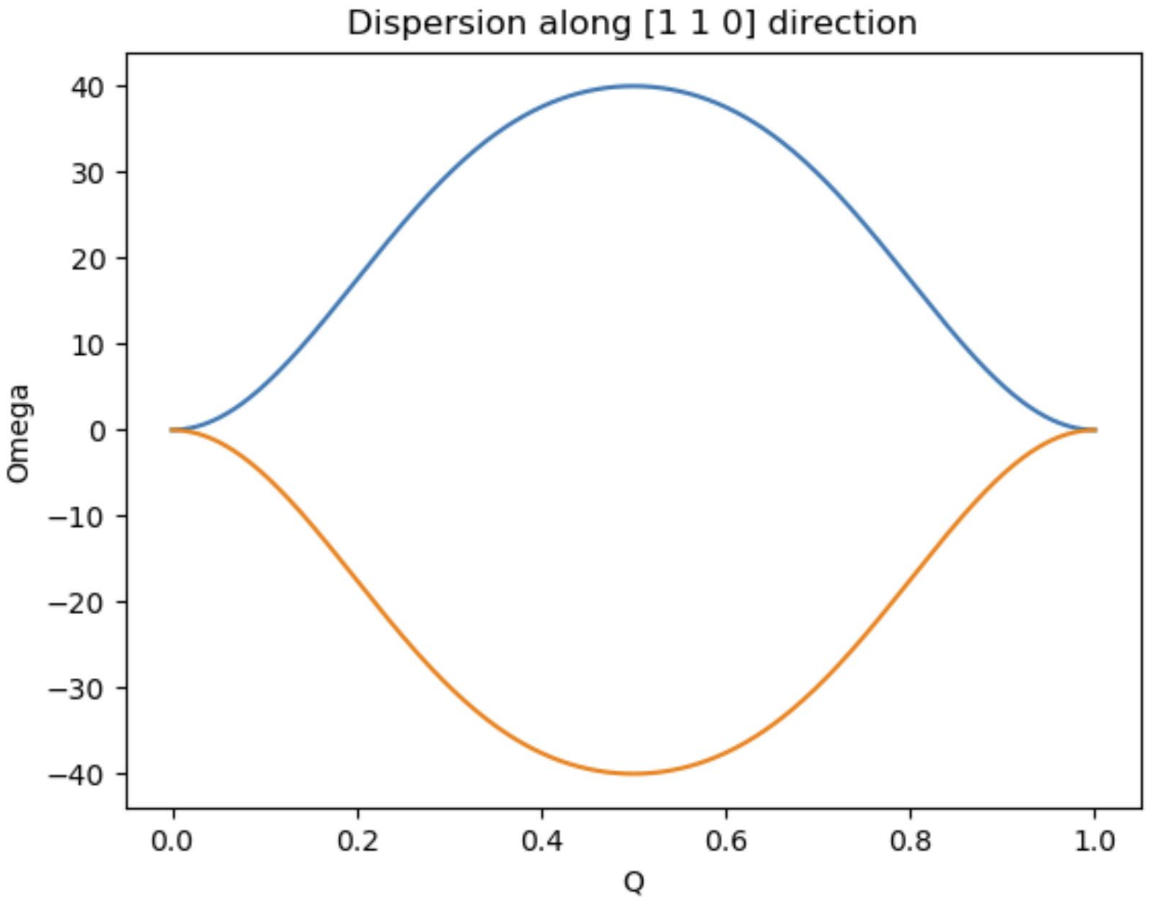




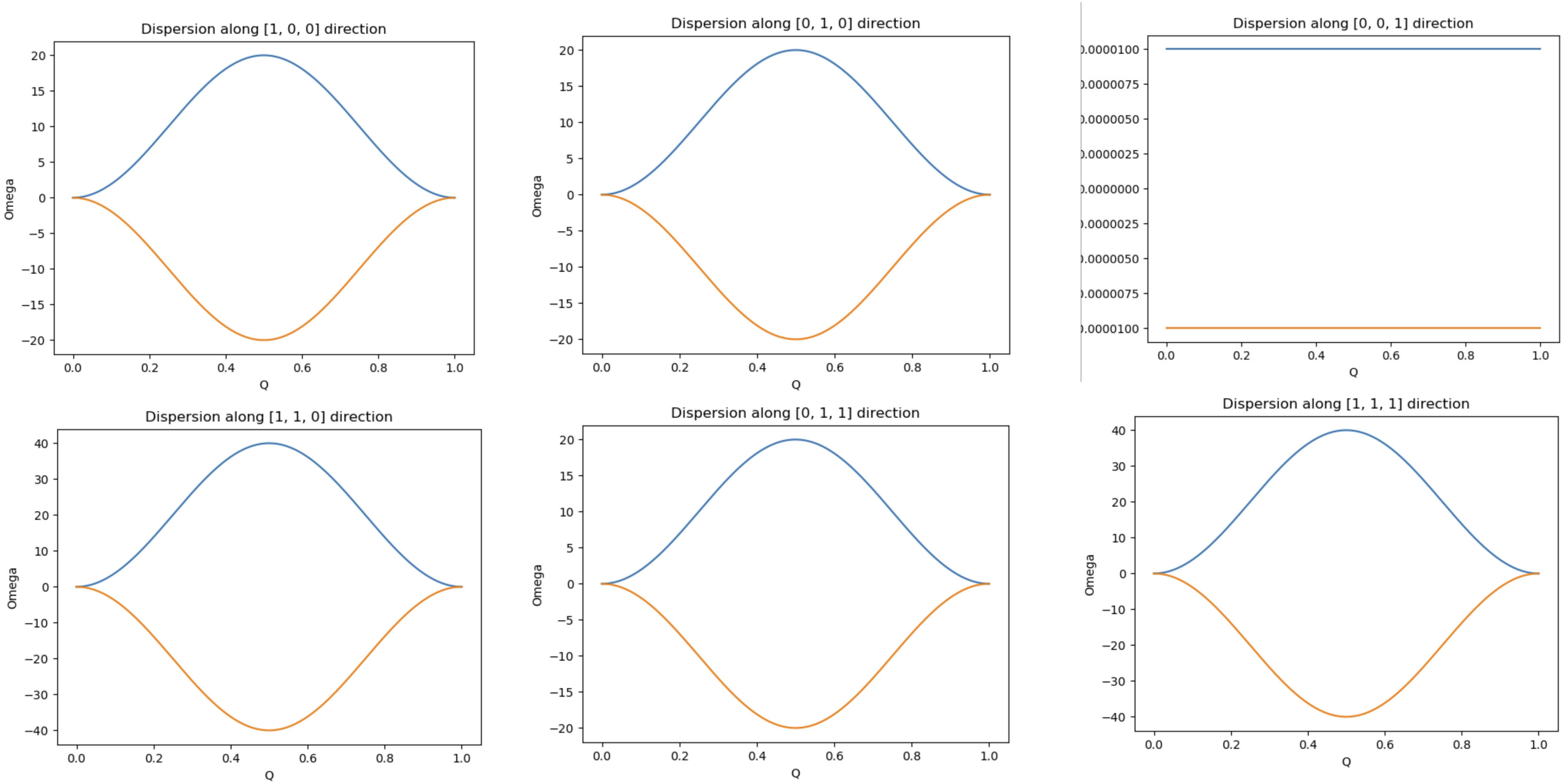
Spinwaves

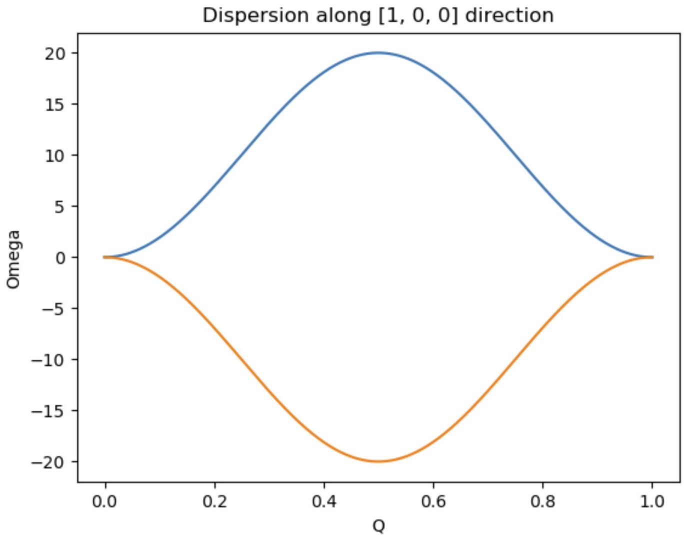


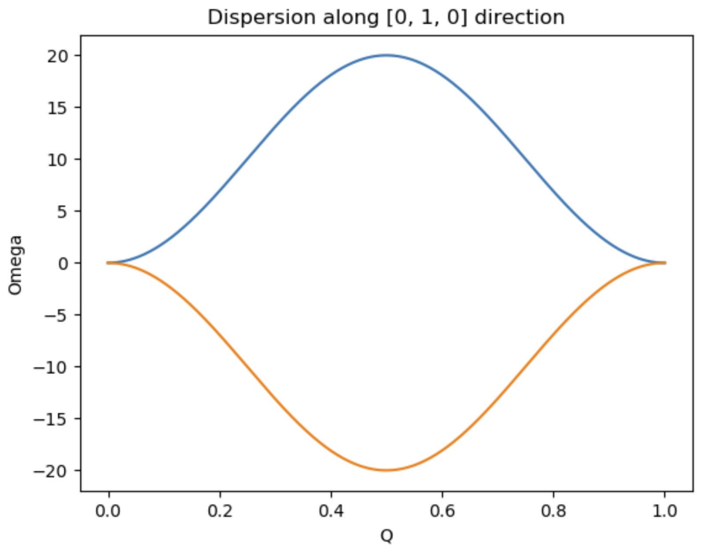
Nearest neighbor model

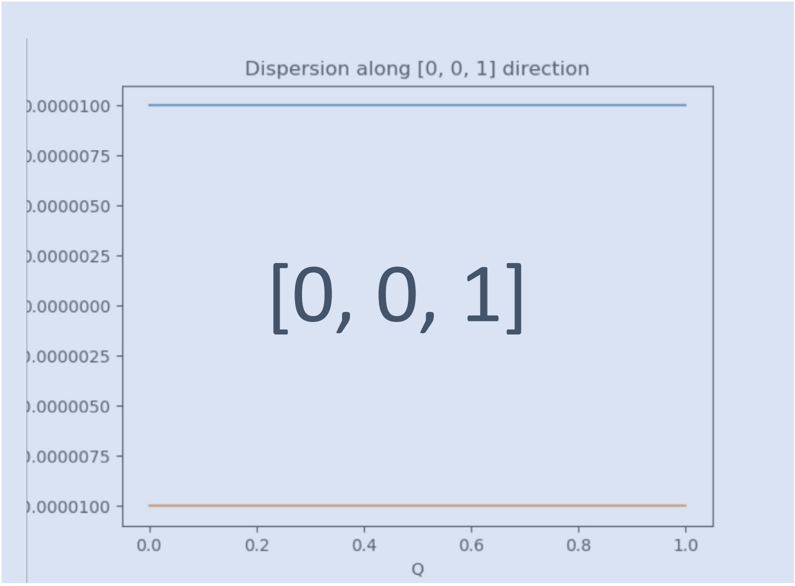


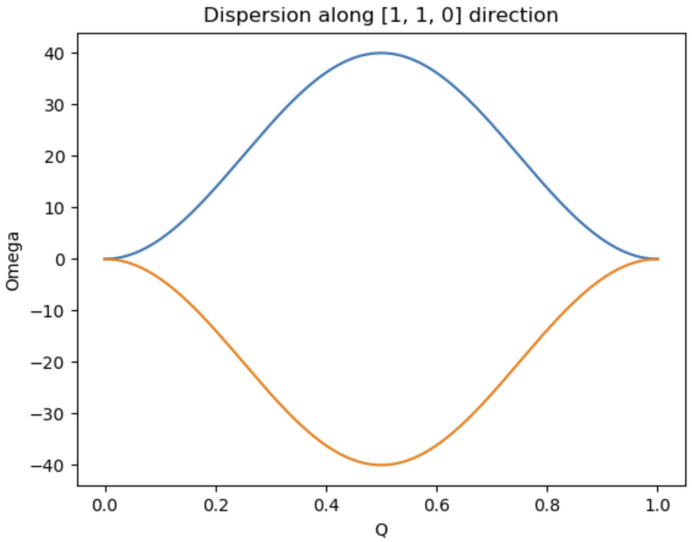
Next-nearest neighbor model

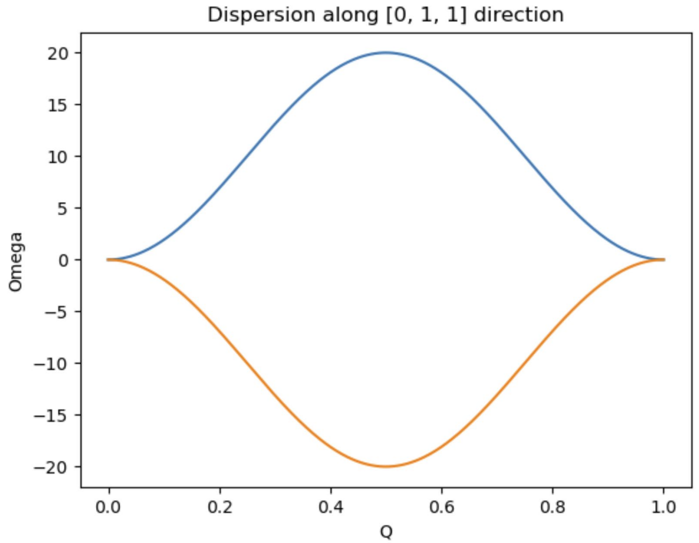

Dispersion along [1, 0, 0] direction
Dispersion along [0, 1, 0] direction
Dispersion along [0, 0, 1] direction
Dispersion along [1, 1, 0] direction
Dispersion along [0, 1, 1] direction
Dispersion along [1, 1, 1] direction
Omega
Q

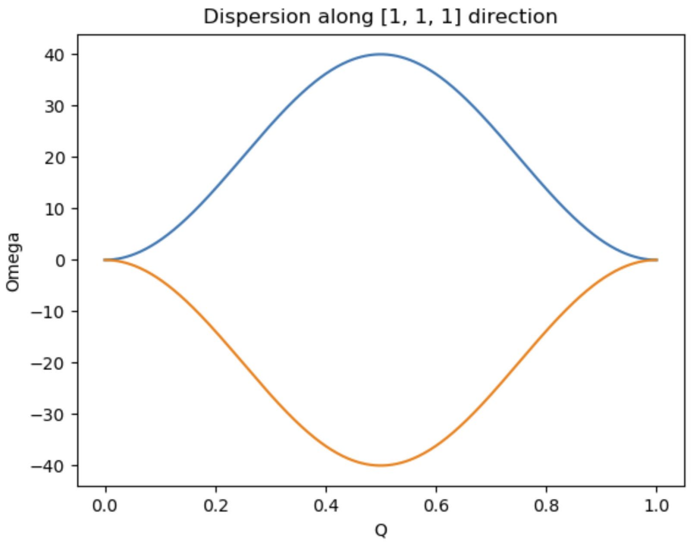

Dispersion along [1, 0, 0] direction
20
15
10
5
0
−5
−10
−15
−20
Omega
0.0
0.2
0.4
0.6
0.8
1.0
Q


Dispersion along [0, 1, 0] direction
20
15
10
5
0
−5
−10
−15
−20
Omega
0.0
0.2
0.4
0.6
0.8
1.0
Q


Dispersion along [0, 0, 1] direction
[0, 0, 1]
0.0
0.2
0.4
0.6
0.8
1.0
Q


Dispersion along [1, 1, 0] direction
40
30
20
10
0
−10
−20
−30
−40
Omega
0.0
0.2
0.4
0.6
0.8
1.0
Q


Dispersion along [0, 1, 1] direction
20
15
10
5
0
−5
−10
−15
−20
Omega
0.0
0.2
0.4
0.6
0.8
1.0
Q

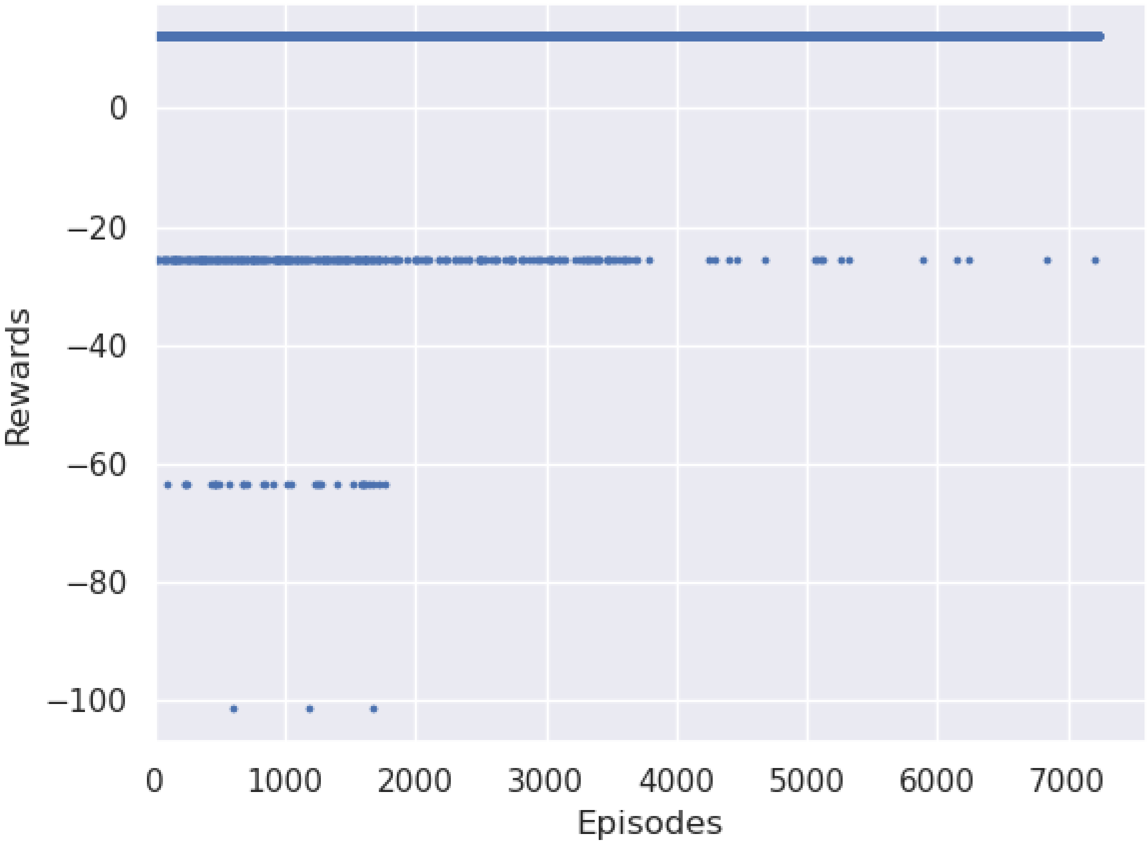

Dispersion along [1, 1, 1] direction
40
30
20
10
0
−10
−20
−30
−40
Omega
0.0
0.2
0.4
0.6
0.8
1.0
Q

GOAL

Minimize the number of measurements necessary by implementing a way to determine the most useful measurements.

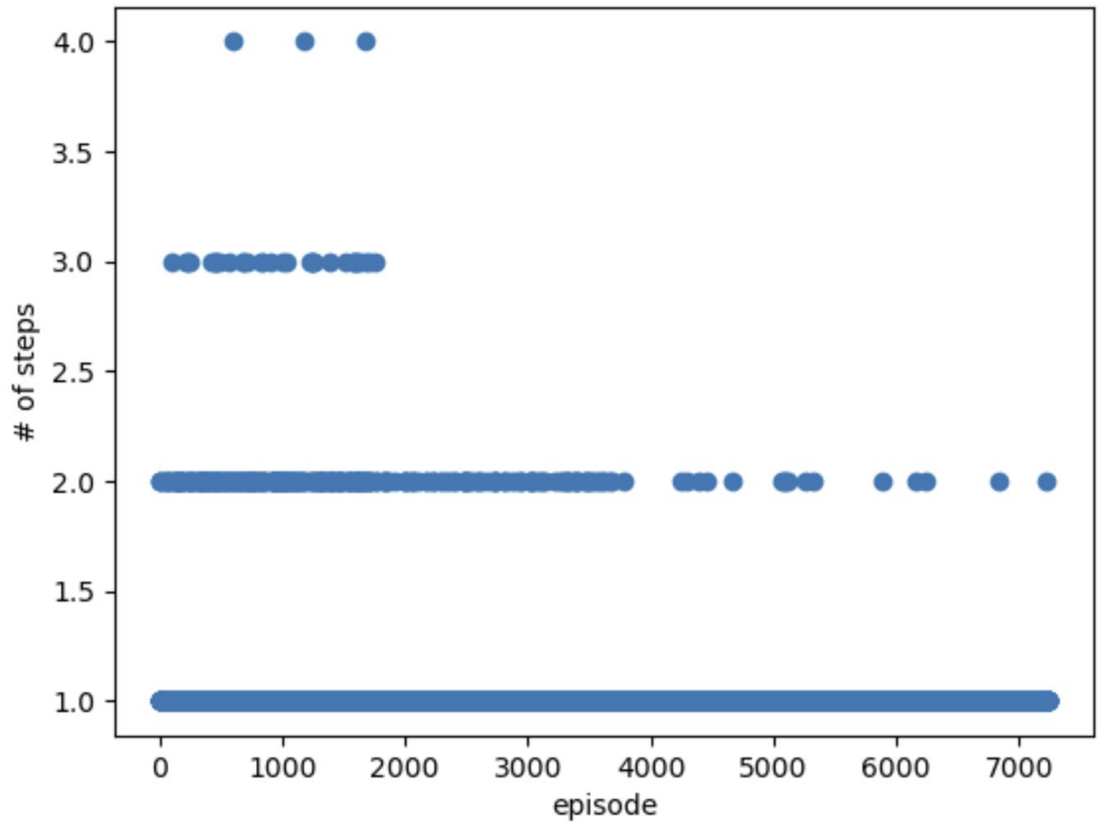
-- useful: distinguish between different models & find correct values

Results

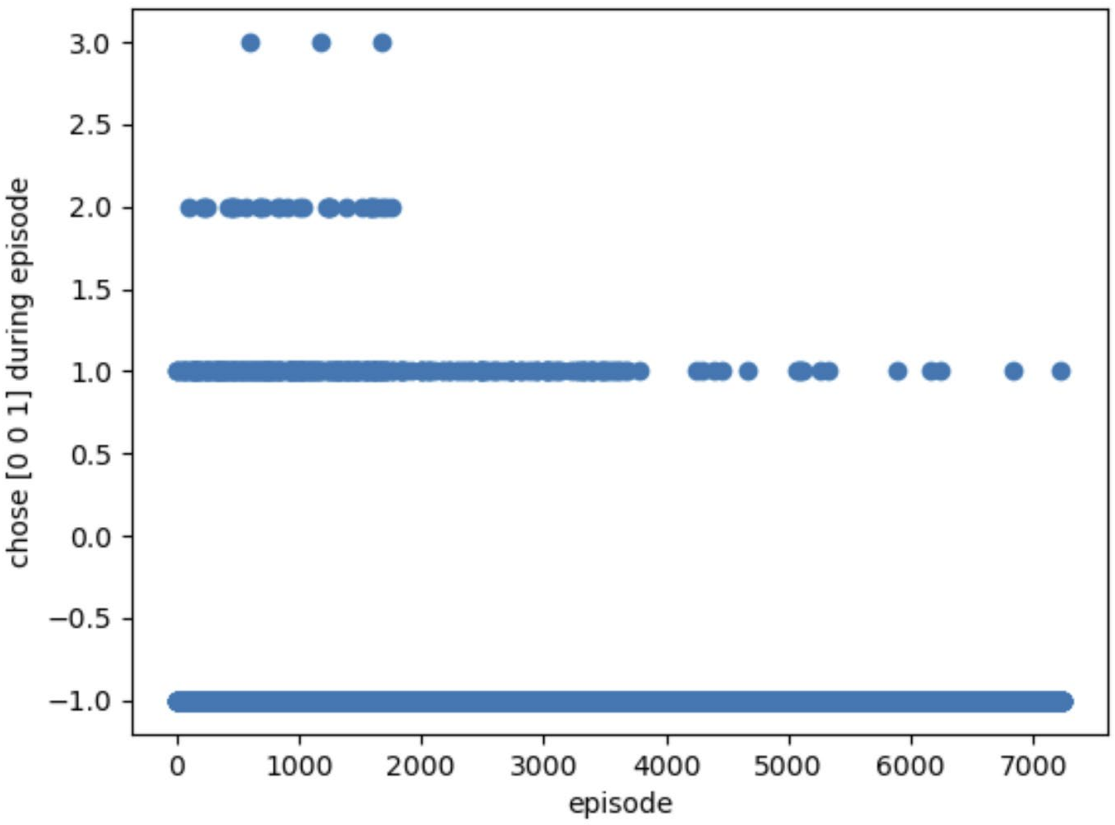


0
−20
−40
−60
−80
−100
Rewards
0
1000
2000
3000
4000
5000
6000
7000
Episodes

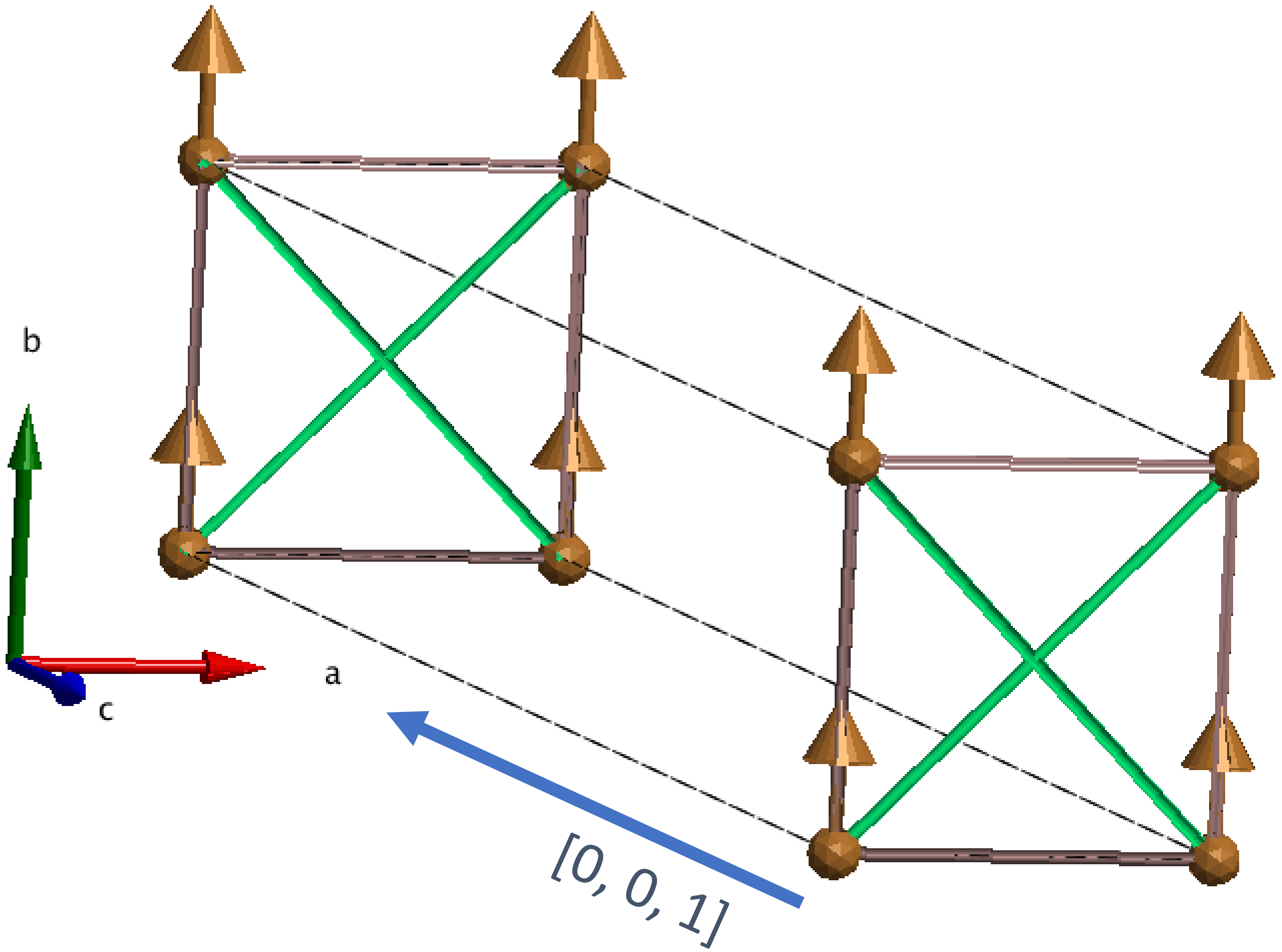
Results



4.0
3.5
3.0
2.5
2.0
1.5
1.0
# of steps
0
1000
2000
3000
4000
5000
6000
7000
episode



3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
chose [0 0 1] during episode
0
1000
2000
3000
4000
5000
6000
7000
episode


b
a
c
[0, 0, 1]

Conclusion