[redacted]

MOHA PROPOSAL FOR GSOC

OVERVIEW

This proposal suggests a project for the Google Summer of Code program, focusing on the development of a software package for model Hamiltonians. The proposal aims to implement various model Hamiltonians, including the XXZ Heisenberg, Ising, t-J-U-V model, and t-J model, and incorporate methods for manipulating Hamiltonian parameters such as Rauk's dictionary and Parr-Pariser dictionary. The project also includes testing of the software package using benchmark systems, optimization for performance and scalability, and writing documentation for users.

Objectives

- **Develop a software package for generating model Hamiltonians for various physical systems that can be used for research at molecular dynamics, condensed matter physics, and quantum mechanics.**
- **Incorporate methods for manipulating the Hamiltonian parameters by specifying energies associated with resonance/hopping/bond term between sites of the lattice**
- **Test the software package using benchmark systems**

ABOUT ME

Basic information

- Name: [redacted]
- Email: [redacted]
- GitHub: [redacted]
- Time zone: GMT-4 (Eastern Daylight Saving Time)

Background and programming experience

I'm PhD student at [redacted], Canada, pursuing a degree in the field of theoretical and computational chemistry. Throughout my undergrad and master's degree I developed strong soft and hard skills that are crucial for successful implementing the Model Hamiltonian project. Specifically, I have experience in working at team for software package for selecting diverse subsets of molecules/materials from larger sets. In this project, I have taken on the majority of the software engineering work and contributed to clustering-based methods while also being willing to perform tedious tasks as needed. Effective collaboration, besides deep understanding of algorithms, was a key component of the project and I received the best feedback by the project leader – Dr. Fanwang Meng (currently a postdoc at MIT). Another project I'm currently working on is about exploring the structure of the electronic wavefunction using tools from mathematical statistics and machine learning. This project requires writing fast and efficient code and reading recent mathematics literature. Also, this project enhances my knowledge in advanced mathematics such as abstract algebra and linear algebra. On top of it, in this project I'm using various strongly correlated systems, including ones described by Hubbard and Huckel models, which gives me a user prospective on the best software for building Model Hamiltonians that I can incorporate in this project.

PROJECT DESCRIPTION AND TIMELINE

Phase 0: before the start of the project

- Conduct literature review of model Hamiltonian theory and get familiar with model Hamiltonians by studying the books:
    - "A Method for Studying Model Hamiltonians" by N. N. Bogolyubov
    - “Group theory in Quantum Mechanics" by Volker Heine.
    - “Modern Quantum Chemistry” by A. Szabo and Neil S. Ostlund
- Get familiar with the existing project architecture, documentation, and testing style that have been already implemented:
    - Read and understand the project's abstract classes to get a high-level understanding of the structure and relationships between classes of the codebase.
    - Review the project's testing suite to understand the types of tests that have been implemented, such as unit tests, integration tests, and acceptance tests.
    - Derive mapping equations between the occupation based and spin based Hamiltonians that will simplify testing on the future steps
    - Practice writing new tests and adding them to the existing testing suite, following the established conventions.

Phase 1: Week 1-4

- Finish implementing the XXZ Heisenberg model Hamiltonian and its more specific variations: XXX Heisenberg, Ising and Richardson-Gaudin models.
    - XXZ Heisenberg model has the following form:

$$\widehat{H}_{XXZ} = \sum_{p} \left(\mu_p^z - J_{pp}^{\mathrm{eq}}\right) S_p^z + \sum_{pq} J_{pq}^{\mathrm{ax}} S_p^z S_q^z + \sum_{pq} J_{pq}^{\mathrm{eq}} S_p^+ S_q^-$$

        - If $J^{\mathrm{ax}} = J^{\mathrm{eq}} \Rightarrow$ XXX Heisenberg model.
        - $J^{\mathrm{eq}} = 0 \Rightarrow$ Ising model
        - $J^{ax} = 0 \Rightarrow$ Richardson-Gaudin Model that has a form:

$$\widehat{H}_{\text{picket fence}} = \sum_{p=0}^{N-1} \left(p - \frac{N-1}{2}\right) + J^{\mathrm{eq}} \sum_{pq} S_p^+ S_q^-$$

    Overall, it’s enough to implement only generalized XXZ Heisenberg model and then specify J^{ax} *and* J^{eq} to get 3 more types of Hamiltonians. Even though some part of code for XXZ Heisenberg model is already written, this part is going to take the most amount of time because it requires building a new class that has to be 1) integrated into the current code architecture and 2) be a parent class for the XXX Heisenberg, Ising and Richardson-Gaudin Hamiltonians.

- Add submodules for testing the implemented models, including sources such as the infinite limit solution using the Bethe ansatz of the antiferromagnetic chain, as described in this reference .

Phase 2: Week 5-8

- Implement the Model Hamiltonian based on both spins and occupation numbers by stacking together already existed model PPP+ Hamiltonian and XXZ Heisenberg model.

    - For modeling superconductors, it is sometimes useful to embellish every site with both occupation numbers and spins. So, in the most general case the New Hamiltonian includes all he terms from the most general occupation-number Hamiltonian and all the terms from the most general spin-Hamiltonian + coupling term:

$$\hat{H}_{occ-spin} = \sum_{p} \varrho_p \left(\hat{n}_{p_\alpha} + \hat{n}_{p_\beta}\right) \hat{S}_p^Z + \sum_{p} \varsigma_p \left(\hat{n}_{p_\alpha} - \hat{n}_{p_\beta}\right) \hat{S}_p^Z + \sum_{p \neq q} \varrho_{pq} \left(\hat{n}_{p_\alpha} + \hat{n}_{p_\beta}\right) \hat{S}_q^Z + \sum_{p \neq q} \varsigma_{pq} \left(\hat{n}_{p_\alpha} - \hat{n}_{p_\beta}\right) \hat{S}_q^Z$$

      The above can be implemented by basically summing already implemented PPP+ Hamiltonian and XXZ Heisenberg model:

$$\hat{H}_{\mathbf{occ-spin}} = \sum_{pq} h_{pq} a_p^\dagger a_q + \sum_{p} U_p \hat{n}_{p\alpha} \hat{n}_{p\beta} + \frac{1}{2} \sum_{p \neq q} \gamma_{pq} (\hat{n}_{p\alpha} + \hat{n}_{p\beta} - Q_p)(\hat{n}_{q\alpha} + \hat{n}_{q\beta} - Q_q) + \sum_{p \neq q} g_{pq}^{pair} a_{p\alpha}^\dagger a_{p\beta}^\dagger a_{q\beta} a_{q\alpha} + \sum_{p} (\mu_p^Z - J_{pp}^{\mathbf{eq}}) S_p^Z + \sum_{pq} J_{pq}^{\mathbf{ax}} S_p^Z S_q^Z + \sum_{pq} J_{pq}^{\mathbf{eq}} S_p^+ S_q^-$$

    - Charge = 0, $g_{pq}^{pair} = \mathbf{0}$ ⇒ t-J-U-V:

$$\hat{H}_{\mathbf{tJUV}} = \sum_{pq} h_{pq} a_p^\dagger a_q + \sum_{p} U_p \hat{n}_{p\alpha} \hat{n}_{p\beta} + \frac{1}{2} \sum_{p \neq q} \gamma_{pq} (\hat{n}_{p\alpha} + \hat{n}_{p\beta})(\hat{n}_{q\alpha} + \hat{n}_{q\beta}) + \sum_{p} (\mu_p^Z - J_{pp}^{\mathbf{eq}}) S_p^Z + \sum_{pq} J_{pq}^{\mathbf{ax}} S_p^Z S_q^Z + \sum_{pq} J_{pq}^{\mathbf{eq}} S_p^+ S_q^-$$

    - U = $\boldsymbol{\gamma}$ = 0 ⇒ t-J model

Implementing this section doesn't require a lot of time and efforts considering that previous steps were coded correctly. Again, by creating the parent class that can be a sum of model Hamiltonians from the previous steps and introducing minor specification of parameters in child classes, 3 model Hamiltonians can be implemented relatively easy.

- Add submodules for testing the created models.

Testing part, however, will require much more time compared to the amount of time spent on testing during the phase 1. That is because there are not so many sources available for reference data. Potentially, reference systems can be generated using existing open source software.

Phase 3: Week 9-12

- Incorporate methods for manipulating the Hamiltonian parameters by Rauk's and Parr-Pariser dictionary for specifying energies associated with resonance/hopping/bond term between sites as described at Hückel molecular theory

- Test the software package using benchmark systems.
- Optimize the software package for performance and scalability.
- Document the software package and provide example codes and tutorials for users.

This phase is focused on polishing and documenting implemented models. The most time-consuming part is implementing Rauk's and Parr-Pariser dictionary that will take from 10 to 14 days, because it requires changing the architecture of the code and finding new reference systems. Documentation and optimization parts shouldn't be hard considering following good practises of writing code on the previous stages and will require minor polishing documentation that will have been already written.

Code architecture