My Dissertation Title

HONORS THESIS

Presented in Partial Fulfillment of the Requirements for Graduate with Research Distinction in the Honors College of Arts and Sciences of The Ohio State University

By

Matthias Heinz

Undergraduate Program in Physics

The Ohio State University

2018

Thesis Committee:

Professor Richard Furnstahl, Advisor

Some Joe #1

Some Jane #2

Some other Person #3

© Copyright by

Matthias Heinz

2018

DRAFT FEBRUARY 19, 2018

Abstract

An abstract goes here. It should be less than **500 words**.

Table of Contents

List of Figures

Figure **Page**

List of Tables

Table **Page**

Chapter 1
Introduction

STILL WIP

- How general to start?
- How can I communicate the general problems of nuclear theory to a layperson?
- Introduction to Hamiltonian formalism?
- Will emphasize computational complexity due to basis size

1.1 Similarity Renormalization Group

The similarity renormalization group (SRG), whose use in nuclear physics was initially explored at OSU, is one method of reducing the computational complexity of low-energy nuclear calculations. The idea behind it is to continuously unitarily transform the operator of interest (for example, the Hamiltonian) into a simpler form. This simpler form is chosen to allow the large basis to be truncated without affecting the operator eigenvalues, which are essential to the truncated operator's utility in later calculations.

1.1.1 Jacobi Coordinates

1.1.2 Harmonic Oscillator States with Proper Symmetry

Chapter 2
Python Library

A large component of my research has been the development of a 1-D SRG Python library. The goal of this library is to offer a logical abstraction to the details of the SRG method to allow others to test features of SRG in one dimension. It aims to be as permissible as possible, keeping in line with the Python duck typing philosophy, while offering useful feedback for unsupported or obviously incorrect usage. A comprehensive set of unit tests for all parts of the library attempts to instill confidence that the library achieves this goal.

2.1 Design

The class-based abstractions offered by the library offer consistent representations of different logical classes of objects present in any SRG run. The goal is to make code for SRG runs clean and clear to any reader, and to encapsulate certain consistency conditions to make it difficult for the programmer to violate them and write obviously incorrect code. The SRG Solver class is logically different from the other classes in that is not a representation of some data, but rather the abstraction of a state machine of sorts, similar to the typically implementation of numericall differential equation solvers. Instances of the other classes represent concrete objects that have unambiguous logical representations.

IDEA: Tree graph explaining the dependencies between classes: ie, basis consists of an ordered set of states, operator couples a matrix to a specific basis, srg evolves an operator, etc.

NOTES:

- Need to work on OOP terms and decide on what specific parts are
- Is the term d̈ependenciesf̈or classes correct?

2.1.1 State

The state class is one of two classes without dependencies. It offers an abstraction for an n-body harmonic oscillator wavefunction. The basic interface consists of a constructor to create an n-body state, and a `val` function to compute the value of the wavefunction at an $n-1$-tuple of momenta. This abstraction is necessary for the transformation from a momentum basis to a harmonic oscillator basis, as the transformation involves the evaluation of the wavefunction at every discretized momentum in the momentum basis.

There are two variants of the state class. The first is the basic 2-body harmonic oscillator state, which abstracts the basic 2-body harmonic oscillator wavefunction. The 2-body harmonic oscillator state is uniquely defined by its harmonic oscillator number. The `val` method returns the value of this single wavefunction, which involves the evaluation of the n-th Hermite polynomial, with n being the harmonic oscillator number.

The second variant is the general n-body (a-body?) state, which abstracts a normalized linear combination of $n-1$-body-2-body states. Every state has an additional condition for consistency: every product state in the linear combination must have the same total harmonic oscillator number. This prevents, among other things, a linear combination of 2-body states from being a valid state, as unique states have

different total harmonic oscillator number. The `val` method returns the linear combination of the `val` method results for the 2-body and $n-1$-body states that make up the state. The recursive definition of the n-body state ensures that any n-body state created will be made up of states that are guaranteed to fulfill the consistency conditions.

2.1.2 Basis

The basis class offers an immutable, iterable, indexable, ordered collection of state objects. The basic interface implemented by all subclasses consists of an n-body property and a type property, to allow for the user to understand the basic qualities of the basis. The basis object is intended to be used to generate transformations and to be coupled with data matrices to give meaning to the data.

The first subclass of the basis class is the p-basis, a quick representation of a momentum basis. The data representation for an n-body p-basis is a list of n-tuples, with the first entry in the tuple being the weight of that node and the remaining $n-1$ entries being the Jacobi momenta for that node. The construction of the momentum basis supports the generation of an even or weighted grid (Gaussian quadrature) in single momentum space, then constructing the n-body space by generating all possible tuples of $n-1$ momenta from the single momentum space. Currently, this has only been used for 2-body momentum bases, as the potential used for my research is a 2-body potential.

The second subclass of the basis class is the HO-basis, a list of state objects. The construction of the n-body HO-basis generates a complete list of appropriatedly symmetrized states up to some user-defined maximum total harmonic oscillator number. As explained in the introduction (where?), the HO-basis offers a more general representation of the data that allows SRG to be applied easily to larger systems, unlike

the p-basis. Thus, the generation of correctly symmetrized general n-body HO-bases is essential to testing SRG for larger systems.

2.1.3 Operator

2.1.4 SRG Solver

2.2 Implementation

2.2.1 Libraries Used

2.2.2 Handling Undefined Behavior

2.3 Testing

2.3.1 Framework

2.3.2 Code Coverage

Chapter 3
Exploring SRG in 1-D

3.1 Objectives

3.1.1 Verify Many-body Force Induction

3.1.2 Test Alternative Generators

3.2 Methods

3.2.1 2-Body Tests

3.2.2 3-Body Tests

Chapter 4
Results

4.1 Many-Body Forces Induced

4.2 Alternative Generators