# Equilibrium and dynamical computation schemes for vibronic models of nonadiabatic systems

by

Neil George Raymond

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Chemistry

Waterloo, Ontario, Canada, 2022

## Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

| | |
|---|---|
| External Examiner: | Michael Schuurman<br>Adjunct Professor, Dept. of Chemistry & Biomolecular Sciences, University of Ottawa |
| Supervisor(s): | Pierre-Nicholas Roy<br>Professor, Dept. of Chemistry, University of Waterloo |
| | Marcel Nooijen<br>Professor, Dept. of Chemistry, University of Waterloo |
| Internal Member(s): | Scott Hopkins<br>Associate Professor, Dept. of Chemistry, University of Waterloo |
| | Germán Sciaini<br>Associate Professor, Dept. of Chemistry, University of Waterloo |
| Internal-External Member: | Roger Melko<br>Professor, Dept. of Physics, University of Waterloo |

## Author's Declaration

## Abstract

Two computational schemes were investigated for studying vibronic models of nonadiabatic systems: Path Integral Monte Carlo (PIMC) and vibrational electronic coupled cluster (VECC).

A PIMC method was used to investigate the distributions of normal modes of nonadiabatic systems. The key element to this approach is the use of gaussian mixture distributions (GMD), and the evaluation of the path integral through matrix products instead of individual matrix elements. By using PIMC with a GMD-reduced scheme, we showed that it is possible to circumvent a sign-problem presented by the non-stoquastic nature of the vibronic Hamiltonian. An alternative GMD scheme was shown to be effective for systems with weaker coupling when the vibronic Hamiltonian is not non-stoquastic.

In recent years a new approach, VECC, was developed, in the Nooijen group by the graduate student Songhao Bao, to describe electronic absorption spectra. The problem of determining detailed equations of motion (EOM) used in VECC was solved and implemented in a software package named `termfactory`. We explored a variety of ways to represent the constraints of the equations of motion for further generalization of `termfactory`. A particular realization of these EOM were developed and applied to produce vibrationally-resolved electronic spectra from auto-correlation functions (ACFs) by way of real-time propagation of a coupled cluster (CC) wave function whose operators are expressed in the second quantized formulation.

Our implementation, called `t-amplitudes`, showed excellent agreement with Multi-configuration time-dependent Hartree (MCTDH) for small molecules, but with much improved computational runtimes. For a larger system, hexahelicene, we were able to replicate the overall shape of the spectra as presented in the literature.

Code for both `termfactory` and `t-amplitudes` is available on GitHub.

## Acknowledgements

This thesis would not be possible if not for the support of various people both within and without academia. My sister who is amazing and puts her all into everything that she does. My parents for their love and endless support.

I am grateful to my supervisors Professors Pierre-Nicholas Roy and Marcel Nooijen for taking a chance on me almost a decade ago. Marcel, who spent countless hours teaching me and reviewing my documents in detail. P.-N. for his continuing moral support, advice and general enthusiasm for life.

I would not have continued on with my studies if not for the community that was fostered in the Theoretical Chemistry research group. I need to especially thank Dmitri Iouchtchenko for his patience and support. I am grateful to have had the opportunity to collaborate with Songhao Bao. Of course, Spencer, Matt, Kevin, Lindsay, and many more contributed to the supportive and social environment.

In addition, I am thankful for the various group members who were readily available to play squash and provide me with a necessary reprieve.

Finally, to all the other people in my life supporting me, thank you!

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

**ACF** auto-correlation function. iv, xi, 4, 18–20, 117, 118, 120, 126–129, 152

**BO** Born-Oppenheimer. vii, 1, 2, 23–28

**BOA** Born–Oppenheimer approximation. 2

**CC** coupled cluster. iv, 4, 14–17, 23, 128

**CCSD** coupled cluster single-double. 129

**CFG** context-free grammar. 185

**DFT** Density Functional Theory. 2, 32, 146

**DoF** degrees of freedom. 2, 4, 6, 8, 9, 23, 24, 44, 48, 83, 91, 119, 122–125, 148

**EOM** equations of motion. iv, 4, 6, 14, 16, 17, 83, 98, 117, 118, 125, 128, 148, 149

**FC** Frank–Condon. 23, 31, 33

**GMD** gaussian mixture distribution. 40, 42, 54, 81, 164, 165

**HF** Hartree-Fock. 15, 32

**HO** harmonic oscillator. 23

**ILP** Integer linear programming. 103, 104, 112, 115, 116, 149

**LVM** linear vibronic model. 129, 146

**MC** Monte Carlo. 9, 10, 39, 164

**MCMC** Markov chain Monte Carlo. 4, 9, 11, 12

**MCMH** Monte Carlo Metropolis-Hastings. 13, 164

**MCTDH** Multi-configuration time-dependent Hartree. xvii, 127–129, 133, 136, 139, 143, 146, 149, 203

**MH** Metropolis-Hastings. xi, 4, 9, 12, 13, 39, 40, 42, 48, 81, 150, 190

**ML-MCTDH** Multi-layer Multi-configuration time-dependent Hartree. 146

**MP** Møller–Plesset perturbation theory. 23, 32

**NACT** nonadiabatic coupling term. 25, 27, 28

**NDPDA** non-deterministic pushdown automaton. xii, 185–187

**NOE** Normal Ordered Exponential. 16, 17

**PDF** probability density function. 10, 165

**PES** potential energy surface. 2, 31, 32

**PI** Path Integral. 6–9, 13, 40, 42, 150, 152, 164, 166

**PIGS** Path Integral Ground State. 150

**PIMC** Path Integral Monte Carlo. iv, 3, 4, 21, 38, 39, 48, 49, 55–58, 60–62, 64–66, 68–80, 148–150, 164, 168

**QM** quantum mechanical. 1, 4, 6, 14, 117, 118

**RR** rigid rotor. 23

**SE** Schrödinger equation. 1, 15, 16, 23, 25–28

**SOS** sum-over-states. 23, 31, 32, 48, 49, 52, 53, 55, 59, 63, 67, 79, 81, 148, 150

**TD-DFT** time-dependent Density Functional Theory. 33, 35

**TDSE** time-dependent Schrödinger equation. 116, 118, 119

**VECC** vibrational electronic coupled cluster. iv, 3–5, 14, 83, 117, 122, 123, 126, 129, 133, 136, 139, 143, 146–149

# List of Listings

# Chapter 1

# Introduction

In computational chemistry, our objective is to compute and predict properties of quantum mechanical (QM) systems via software based on of mathematical models and employing sophisticated algorithms and approximations to achieve maximum efficiency and accuracy. Consistent, efficient, and accurate calculations are needed across a wide spectrum of technologies and fields of study, and computational chemistry can provide this data at a significantly reduced cost when compared to in-lab experiments. To take just two examples, combinatorial computational chemistry has been used to study options for battery cathodes in lithium batteries that are used in EVs [1], while the investigation of geometric, electronic, and optical properties of electronic acceptor materials allows one to design better organic solar cells? [2].

In Quantum mechanics, one of the paramount equations is the Schrödinger equation (SE). Heisenberg showed us that this differential equation could be cast as an eigenvalue problem in which eigenvalues correspond to the energies of the states, and the eigenvectors can be used to construct wavefunctions. These two ingredients can be used to construct a series of physical observables and properties. The approach one takes in addressing the SE depends on systems and properties of interest. In Quantum mechanics, the total energy of a system is described by the Hamiltonian operator $\hat{H}$. Usually the Hamiltonian is composed of kinetic and potential energy terms

$$\hat{H} = \hat{K} + \hat{V}. \tag{1.1}$$

We can express the time-independent form of the SE

$$\hat{H}\left|n\right\rangle = E_n\left|n\right\rangle \tag{1.2}$$

where $E_n$ are known as the energy eigenvalues and $|n\rangle$ are the eigenstates. Note that we use the Dirac notation here and will use it throughout this thesis. To solve the SE one can use a basis $|i\rangle$ to represent the Hamiltonian operator $\hat{H}$ as a matrix $\boldsymbol{H}$ with matrix elements $H_{ii'} = \langle i|\hat{H}|i'\rangle$.

Routine applications of electronic structure methods in quantum chemistry use the Born-Oppenheimer (BO) approximation in which the focus is on a single electronic-state at a time, and the effects of nuclear motion can be included using a (model) potential energy surface.

Take a common computational program, such as `Gaussian`[3]. We can obtain very accurate properties for common systems, such as the entropy of dinitrogen ($N_2$). Calculating the entropy using `Gaussian` takes on the order of tens of seconds and produces a result that is within $6 \times 10^{-2} \mathrm{cal\,mol^{-1}\,K^{-1}}$ of experimental results, as seen in Table 1.1 [1]

|  | S ($\mathrm{cal\,mol^{-1}\,K^{-1}}$) |
|---|---|
| Gaussian | $4.5735 \times 10^{1}$ |
| NIST[4] | $4.5796 \times 10^{1}$ |

Table 1.1: Entropy of Dinitrogen ($N_2$)

The BO approximation may break down when electronic surfaces are close together in important regions of nuclear configuration space. Such situations require a description of coupled motion of electrons and nuclei, involving multiple electronic states. A convenient approach is the use of a vibronic model Hamiltonian: vibronic meaning that it considers only vibrational and electronic contributions; not rotational, magnetic, translation, etc.

In the vibronic frame of reference, our basis is comprised of two degrees of freedom (DoF): $A$ electronic states $a_i \in a_1, a_2, \ldots, a_A$, and $N$ vibrational normal modes $q_i \in q_1, q_2, \ldots, q_N$. For the nuclear co-ordinates the easiest thing to do is to have a product basis.

$$|i\rangle = \bigotimes_{k=1}^{N} |i_k\rangle \tag{1.3}$$

Typically the index $i_k$ runs from 1 to $d_k$, the size of the Hilbert space for DoF $k$. We set the number of basis functions to be BF $= d_k$ per DoF, where we assume that, for the purpose of this illustration, $d_k$ is the same for all $k$ [5].

While in a standard electronic structure BO approach a single surface $a_1$ would be considered and $\hat{V}$ would describe a single potential energy surface (PES), vibronic models consider multiple surfaces $a_1, a_2, \ldots, a_A$ and $\hat{V}$ is instead a potential energy matrix. The vibronic Hamiltonian will be expressed in the diabatic basis[2] (aka a non-adiabatic basis) as opposed to an adiabatic basis[3]. Adiabatic in this case means the potential $\hat{V}$ contribution is diagonal with respect to the basis, whereas in the diabatic basis it is instead the kinetic $\hat{K}$ contribution that is diagonal.

One simple reason for the use of approximations, like the Born–Oppenheimer approximation (BOA) is the data size of the mathematical terms, such as the Hamiltonian. As seen in Figure 1.1 the amount of space necessary to store the Hamiltonian matrix on a computer roughly follows:

$$\text{GB of storage} \propto \left(A * \mathrm{BF}^{N}\right)^{2}. \tag{1.4}$$

[1]Calculated in the ground state using Density Functional Theory (DFT), `B3LYP cc-pVTZ` basis.

[2]Diabatic states are further described in Section 3.3.

[3]This notion of adiabatic is **not** the "thermodynamic" notion of adiabatic. We mean adiabatic in the sense that wavefunctions are eigenfunctions of the Hamiltonian.

The blue line represents the storage required for the matrix representation of a Hamiltonian defined using BF $= 15$ basis functions[4] and $A = 4$ electronic states. The amount of storage required to add 1 normal mode $N = 4 \rightarrow N = 5$ increases by two orders of magnitude: from 640GB (0.64TB), which is feasible to store on RAM for large compute servers, to 100TB, and therefore requires storage on disk. Even a relatively small system of $N = 10$ modes and $A = 4$ electronic states requires $10^{16}$ GB for an insufficient 15 basis functions. Furthermore, this is only the size required to store the Hamiltonian, we haven't even performed any computations! This storage cost is an example of what is referred to as the curse of dimensionality.

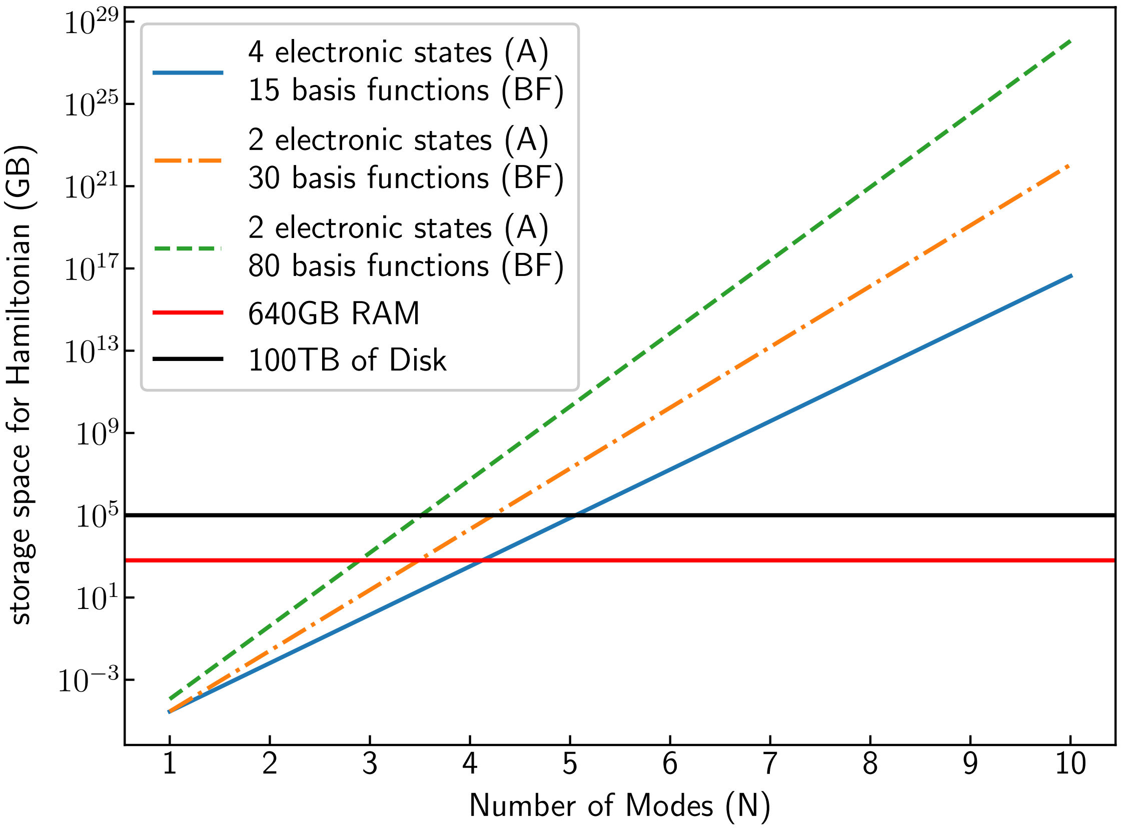

Figure 1.1: Memory (GB) needed to store Hamiltonian per number of basis functions.

The heart of the matter is finding a good balance between accuracy and computational performance by exploiting contextually appropriate approximations. The type of system, the form of the Hamiltonian, and the wavefunction, are some of the factors that affect which approximations are "good".

In this thesis the starting point will be a vibronic model and the goal is to develop suitable strategies to calculate thermal or spectral properties based on an underlying vibronic model. The two approaches Path Integral Monte Carlo (PIMC) and vibrational electronic coupled cluster (VECC) are quite different, and consequently they use different wavefunctions, different approximations, and different methods of evaluating properties of interest.

[4]Harmonic oscillator

## 1.1 Overview of the thesis

This thesis is a study of more efficient methods for a specific class of QM systems, namely vibronic models, of nonadiabatic systems. In Chapter 3, I describe the motivation for these models by showing where common approximations break down. I describe how the new diabatic representation can overcome numerical approximation issues in Section 3.3. An explicit diabatization scheme for deriving these vibronic models is presented in Section 3.3.2. The reader should be familiar with the mathematical definitions of the vibronic models in Section 3.4 as they are used extensively in the remainder of the thesis. I provide a concrete example of a diabatization procedure in Section 3.5.

We will look at two methods: PIMC and VECC.

The PIMC method is focused on systems that are in thermodynamic equilibrium. Through the use of Feynman path integrals, Monte Carlo Markov Chains, and various numerical approximations, we can compute statistical estimates of QM properties. I outline the basic PIMC method for approximating integrals in Section 2.1, the general idea behind Markov chain Monte Carlo (MCMC) in Section 2.2, and finish by presenting an illustrative example of the Metropolis-Hastings (MH) method. In this work I introduce the notion of tracing out electronic DoF exactly, instead of stochastically, which reduces the variance by avoiding a sign problem. The majority of the PIMC work is presented in Chapter 4, detailing the subtleties of the method, as well as discussing its results.

The VECC approach, is used to produce vibrationally-resolved electronic spectra from auto-correlation functions (ACFs) by way of real-time propagation of a coupled cluster (CC) wavefunction whose operators are expressed in the second quantization formalism[5]. The pathway for these complicated calculations is covered in Section 2.3. The extension of CC theory from standard single surface vibrational problems to multiple electronic states is discussed in Section 2.3.1. The exact derivation of this representation is not presented in this thesis as it was not a contribution of this author[6]. Section 2.3.2 lays out the details of how VECC theory can be applied to generate ACFs and subsequently vibrationally-resolved electronic spectra. The VECC approach in this thesis is limited to time-dependent problems[7].

By parameterizing a wave function, deterministic equations of motion (EOM) can be solved to propagate wavefunctions in real and imaginary time. There are many possible choices of parameterization, each with their own accuracy and computational cost. Small changes in the parameterization can, and do, lead to drastically different EOM. In order to facilitate exploration of different parameterizations, I developed a program to derive and implement EOM. This program eliminates the need to manually derive EOM, avoiding mistakes and wasted effort. The focus of Chapter 5 is on this program: the problem it is designed to solve, how it operates in practice, and explorations of more general approaches to handle future challenges.

[5]Sometimes also referred to as occupation number representation.

[6]If the reader is interested in reading more about the derivation the primary source is [6].

[7]In principle the VECC approach can be extended to thermal problems, which is discussed further in Section 2.3.

Finally, in Chapter 6, using a particular parameterization, we simulate vibronic spectra using VECC.

# Chapter 2

# Background Theory

Sections 2.1 and 2.2, cover the background information for Chapter 4. Section 2.3 is a primer for Chapters 5 and 6.

## 2.1 Path Integral Formulation of Statistical Mechanics

In exact diagonalization, one expands the wavefunction into a basis. The size of the basis grows exponentially with the number of DoF. For instance, if one takes $d$ basis functions per DoF, the total size of the basis for $N$ DoF is $d^N$. Since the cost of exact diagonalization scales as $(d^N)^3$, we therefore say that the cost of exact diagonalization scales exponentially with the number of DoFs.

In the Path Integral (PI) formulation of statistical mechanics, we replace the QM trace by a functional integral[7]. In practice, we partition this functional integral into $P$ discrete integrals. The advantage of this new representation is that classical EOM are easier to solve.

Here I briefly outline the core principle of the imaginary time PI formulation that will be employed in this thesis.

### 2.1.1 Basic Formulation

For a system described by a Hermitian Hamiltonian $\hat{H}$ and a basis of continuous DoF $x$

$$I = \int \mathrm{d}x \, |x\rangle \langle x| \,, \tag{2.1}$$

the canonical partition function is

$$Z = \int \mathrm{d}x \, \langle x| \, e^{-\beta \hat{H}} \, |x\rangle \,. \tag{2.2}$$

With a Hamiltonian $\hat{H} = \hat{K} + \hat{V}$ we commonly want to treat the kinetic and potential operators separately. The issue is that, in general, $e^{-\hat{K}}$ and $e^{-\hat{V}}$ do not commute

$$\left[e^{-\hat{K}}, e^{-\hat{V}}\right] \neq 0, \tag{2.3}$$

which means

$$e^{-\beta\hat{H}} = e^{-\beta(\hat{K}+\hat{V})} \neq e^{-\beta\hat{K}}e^{-\beta\hat{V}}, \tag{2.4}$$

and so we employ the Trotter-Suzuki factorization[1][8]

$$e^{A+B} = \lim_{N\to\infty}(e^{A/N}e^{B/N})^N, \tag{2.5}$$

which is exact in the limit, allowing us to treat $\hat{K}$ and $\hat{V}$ separately

$$e^{-\beta\hat{H}} = e^{-\beta(\hat{K}+\hat{V})} = \lim_{P\to\infty}(e^{-\frac{\beta}{P}\hat{K}}e^{-\frac{\beta}{P}\hat{V}})^P = \lim_{P\to\infty}(e^{-\tau\hat{K}}e^{-\tau\hat{V}})^P, \tag{2.6}$$

which we apply to Equation (2.2)

$$Z = \lim_{P\to\infty}\int \mathrm{d}x \, \langle x| \prod_{i=1}^{P} e^{-\tau\hat{K}}e^{-\tau\hat{V}} |x\rangle \,. \tag{2.7}$$

Inserting resolutions of the identity produces a PI discretization of Z:[2]

$$Z = \lim_{P\to\infty}\int \mathrm{d}x_1\,, \mathrm{d}x_2\,, \ldots, \mathrm{d}x_P \prod_{i=1}^{P} \langle x_i| \, e^{-\tau\hat{K}}e^{-\tau\hat{V}} \, |x_{i+1}\rangle \,, \tag{2.8}$$

where $\tau = \frac{\beta}{P}$ for $P$ imaginary-time slices (also known as "beads").

## Applications beyond the partition function

If other properties of interest are desired, it is straightforward to insert an 'action' operator $\hat{A}$ inside to obtain some property $A$:

$$\langle A\rangle = \frac{1}{Z}\,\mathrm{Tr}\left[e^{-\beta\hat{H}}\hat{A}\right] \tag{2.9}$$

$$= \frac{1}{Z}\lim_{P\to\infty}\int \mathrm{d}x_1\,, \mathrm{d}x_2\,, \ldots, \mathrm{d}x_P \left(\prod_{i=1}^{P} \langle x_i| \, e^{-\tau\hat{K}}e^{-\tau\hat{V}} \, |x_{i+1}\rangle\right) A_{\mathrm{estim}}(x_1, x_2, \ldots, x_P). \tag{2.10}$$

[1]Also known as the Lie product formula.

[2]A step-by-step derivation is shown in supplementary material.

For example, suppose we are interested in the distribution of a particular co-ordinate $x$:

$$\rho(x) = \frac{1}{Z} \operatorname{Tr}\Big[e^{-\beta \hat{H}} \delta(\hat{x} - x)\Big]. \tag{2.11}$$

This is known as the diagonal of the reduced density matrix for co-ordinate $x$. In path integral form it is expressed as:

$$\rho(x) = \frac{1}{Z} \lim_{P\to\infty} \int \mathrm{d}x_1 \,, \mathrm{d}x_2 \,, \ldots, \mathrm{d}x_P \left( \prod_{i=1}^{P} \langle x_i | \, e^{-\tau \hat{K}} e^{-\tau \hat{V}} \, | x_{i+1} \rangle \right) \frac{1}{P} \sum_{i=1}^{P} \delta(x_i - x). \tag{2.12}$$

The average over $\delta(x_i - x)$ is a reflection of the equivalency of all the beads ($P$): the Feynman path is cyclic in imaginary time. Some other properties of interest include: the distribution of individual modes $\boldsymbol{q}_i$, the true distribution, the partition function, the expectation value of the energy, and the Helmholtz energy.

In this way, different properties can be expressed in a PI formulation. Similar formulations can be derived for different choice of basis, partitioning of the Hamiltonian, and order of Suzuki-Trotter factorization. Each will exhibit different trade-offs between accuracy and computational efficiency.

In the next section I will outline the specific formulation that I chose to use.

### 2.1.2 Representations specific to this work

The specific path integral form used in Chapter 4 is different to the general form listed above in Equation (2.8). It will be expressed in terms of both continuous vibrational DoFs and discrete electronic DoFs. In the same fashion as before, the continuous co-ordinates $q$ will be treated through integration and the discrete co-ordinates $A$ will be treated through summation.

The systems explored with PIs will be expressed in terms of $N$ continuous normal mode co-ordinates

$$\boldsymbol{q} = q_1, q_2, \ldots, q_N \quad | \quad q_i \in \mathcal{R}, \tag{2.13}$$

and $A$ discrete electronic states

$$\boldsymbol{a} = a_1, a_2, \ldots, a_A \quad | \quad a_i \in \{1, 2, \ldots, A\}\,. \tag{2.14}$$

I make use of a compact notation to represent $P$ integrals over $\boldsymbol{q}_i$ [3]

$$\int \mathrm{d}\boldsymbol{Q} = \int \mathrm{d}\boldsymbol{q}^P = \int \mathrm{d}\boldsymbol{q}_1 \,, \mathrm{d}\boldsymbol{q}_2 \,, \ldots, \mathrm{d}\boldsymbol{q}_P \,, \tag{2.15}$$

[3]Where $\boldsymbol{Q}$ and $\boldsymbol{q}^P$ can be used interchangeably depending on whichever is most helpful in aiding the reader.

and

$$\sum_{\boldsymbol{a}}^{A} = \sum_{a_1=1}^{A} \sum_{a_2=1}^{A} \cdots \sum_{a_P=1}^{A} . \tag{2.16}$$

For these DoFs, the PI discretization of Z can be expressed:

$$Z = \lim_{P\to\infty} \int \mathrm{d}\boldsymbol{q}^P \sum_{\boldsymbol{a}}^{A} \langle \boldsymbol{q}_i, a_i | \, e^{-\tau\hat{K}} e^{-\tau\hat{V}} \left| \boldsymbol{q}_{i+1}, a_{i+1} \right\rangle . \tag{2.17}$$

In this representation, I have a mixture of path integrals and path sums due to the mixture of continuous and discrete DoF. In Chapter 4, the Hamiltonian is partitioned in a specific way $\hat{H} = \hat{h} + \hat{V}$ and has two important properties: $\hat{h}$ is diagonal in the discrete electronic DoFs, and $\hat{V}$ is diagonal in the continuous DoFs. This means that

$$Z = \lim_{P\to\infty} \int \mathrm{d}\boldsymbol{q}^P \sum_{\boldsymbol{a}}^{A} \langle \boldsymbol{q}_i, a_i | \, e^{-\tau\hat{h}} \left| \boldsymbol{q}_{i+1}, a_i \right\rangle \left\langle \boldsymbol{q}_{i+1}, a_i \right| e^{-\tau\hat{V}} \left| \boldsymbol{q}_{i+1}, a_{i+1} \right\rangle . \tag{2.18}$$

The exact definition of this partitioning and the reason for these properties are presented in Section 3.4.1.

For the purpose of this thesis, it is important to highlight that the error in the Trotter-Suzuki expansion is inversely proportional to $P$, and that as $P \to \infty$ the expansion is exact. Therefore when applying this method to systems, one commonly shows that they are "converged" in the imaginary-time dimension; or said another way, that they have a sufficiently large number of beads.

Now that I have derived PI expressions for properties of interest in general, I will explain how to evaluate these integrals with a Monte Carlo (MC) scheme.

## 2.2 Monte Carlo and Metropolis Hastings

Section 2.1 outlines how to express properties of interest by evaluating integrals over matrix elements of $e^{-\beta\hat{H}}$. In practice, these integrals are very expensive to calculate and it is common to employ various approximate methods to avoid direct calculation of the integrals. In this thesis, I employ the use of MCMC integration, specifically a MH scheme, to approximate these integrals.

### 2.2.1 Monte Carlo Integration

MC at its core is a method for approximating an integral. The method is exact in the limit of infinite samples, but in real-world situations, has an error proportional to $\frac{1}{\sqrt{N}}$ for $N$ samples. In general, "Monte Carlo" methods refers to a broad class of computational/mathematical algorithms. For the purpose of this thesis, whenever I discuss Monte Carlo I am referring specifically to Monte Carlo integration, which is a numerical technique for approximating an integral.

For a continuous random variable $X$ having a probability density function (PDF) $p(x)$, the expected value of a function $f(x)$ is:

$$\langle f(x)\rangle_p = \frac{\int \mathrm{d}x\, p(x) f(x)}{\int \mathrm{d}x\, p(x)} \tag{2.19}$$

We define a Monte Carlo estimator as the mean of $f(x)$ over $N$ samples from $p(x)$, $(x_1, \ldots, x_N)$.

$$\tilde{f}(x) = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{2.20}$$

This ratio of integrals can be approximated using the Central Limit Theorem[9, pp. 278-281]

$$\frac{\int \mathrm{d}x\, p(x) f(x)}{\int \mathrm{d}x\, p(x)} = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \pm \left(\frac{\sigma^2}{N}\right)^{\frac{1}{2}} \tag{2.21}$$

$$\langle f(x)\rangle_p = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{2.22}$$

Note that the second term in Equation (2.21) is inversely proportional to $\sqrt{N}$. Therefore to obtain an effective estimate, a small variance $\sigma^2$ or a large sample size is required. A reduction in the variance of a sample distribution is computationally advantageous as smaller sample sizes result in shorter run times. In an ideal world, sampling from $p(x)$ would be easy and computational efficient, but in practice many PDFs of interest have numerical complications. Various methods exist to tackle this problem, two of which I present below.

### 2.2.2 Importance Sampling

Importance sampling is a statistical method that can reduce the variance $\sigma^2$ of sampling. Certain values of the input random variables in a simulation have more impact on the parameters being estimated than others. If these important contributions are emphasized by sampling more frequently, then the estimator variance is reduced. The principle of importance sampling is that sampling from a new distribution $d^*$ is equivalent to sampling with weight $\frac{d^*}{d}$ from our original distribution $d$, biasing the sample obtained towards $d$ [9, pp. 284-286]. The reduction in variance leads to more efficient calculation of parameters of interest, since fewer samples are needed for convergence. In addition, we can choose the new distribution $d^*$ so that it is more efficient to sample than $d$. Here is a simple example of how importance sampling can be used to express the partition function:

$$Z = \int \mathrm{d}\boldsymbol{q}\, d(\boldsymbol{q}) \tag{2.23}$$

$$Z = \int \mathrm{d}\boldsymbol{q}\, d^*(\boldsymbol{q}) \frac{d(\boldsymbol{q})}{d^*(\boldsymbol{q})} \tag{2.24}$$

$$\frac{Z}{\int \mathrm{d}\boldsymbol{q}\, d^*(\boldsymbol{q})} = \frac{\int \mathrm{d}\boldsymbol{q}\, d^*(\boldsymbol{q}) \frac{d(\boldsymbol{q})}{d^*(\boldsymbol{q})}}{\int \mathrm{d}\boldsymbol{q}\, d^*(\boldsymbol{q})} \tag{2.25}$$

$$\frac{Z}{\int \mathrm{d}\boldsymbol{q}\, d^*(\boldsymbol{q})} = \left\langle \frac{d(\boldsymbol{q})}{d^*(\boldsymbol{q})} \right\rangle_{d^*} \tag{2.26}$$

$$Z = \left\langle \frac{d(\boldsymbol{q})}{d^*(\boldsymbol{q})} \right\rangle_{d^*} \left( \int \mathrm{d}\boldsymbol{q}\, d^*(\boldsymbol{q}) \right). \tag{2.27}$$

Other quantities of interest can also be expressed in terms of $d$ and $d^*$, by deriving new estimators. Of note is that for certain distributions $d^*(\boldsymbol{q})$, the term $\int \mathrm{d}\boldsymbol{q}\, d^*(\boldsymbol{q})$ can be analytically evaluated.

Importance sampling provides one approach to addressing the difficulty of producing samples $x_i \in X$ by lowering the variance. Next we will look at another approach: MCMC.

### 2.2.3 Markov Chain Monte Carlo methods

Markov chain Monte Carlo are a group of methods for sampling from a probability distribution. Whereas in importance sampling each sample was drawn independently, in MCMC samples are now autocorrelated, which allows MCMC algorithms to narrow in on the quantity that is being approximated, even with a large number of random variables. Instead of having to design an optimal distribution $d^*$ from which to sample, MCMC methods construct a Markov Chain by starting in an initial state $X_0$ and transitioning to new states $X_1, X_2, \cdots$ [10, pp. 270]. Assuming the Markov Chain is ergodic, then the probability of drawing a sample $x_i$ will have probability density proportional to the target distribution $P(x)$.

An integral can then be evaluated as the expectation value over $X$[4],

$$\langle f(x) \rangle_X \rightarrow_{\mathrm{MCMC}} \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(X_i). \tag{2.28}$$

One can calculate properties of the form $\langle A \rangle$ from the trace

$$\langle A \rangle = \frac{1}{Z} \mathrm{Tr}\left[ e^{-\beta \hat{H}} \hat{A} \right], \tag{2.29}$$

[4]Employing the Markov chain central limit theorem in a somewhat similar manner as the central limit theorem in Equation (2.21).

and also obtain histograms

$$h(x) = \frac{1}{Z} \operatorname{Tr}\Big[e^{-\beta\hat{H}} \delta(\hat{x} - x)\Big]. \tag{2.30}$$

For example, if we want to compute the distribution of the first normal mode $q_1$

$$h(q_1) = \frac{1}{\operatorname{Tr}_R[\hat{\varrho}_R(\beta)]} \operatorname{Tr}_R[\hat{\varrho}_R(\beta)\delta(\hat{q}_1 - q_1)] = \langle \delta(\hat{q}_1 - q_1) \rangle \,. \tag{2.31}$$

There are multiple approaches for constructing Markov Chains. For the purposes of this thesis, I focus solely on the Metropolis-Hastings MCMC approach.

### 2.2.4 Metropolis-Hastings Algorithm

Metropolis-Hastings (MH) method gives a prescription for generating a Markov Chain.

A MH algorithm for a target distribution $P(x)$ and a sampling distribution $d(x)$ with $T$ steps:

**Algorithm 1:** Generic Metropolis-Hastings algorithm

```
1  $x_0 \leftarrow A$
2  $t \leftarrow 0$
3  while $t \leq T$ do
4      $x' \leftarrow d(x'|x_t)$
5      $A(x', x_t) = \min(1, \frac{P(x')d(x_t|x')}{P(x_t)d(x'|x_t)})$
6      $u \leftarrow \mathcal{U}\{0, 1\}$
7      if $u \leq A(x', x_t)$ then
8          $x_{t+1} = x'$
9      else
10         $x_{t+1} = x_t$
11     $t \leftarrow t + 1$
```

Applying this algorithm generates a series of states $x_0, x_1, x_2, \ldots, x_T$ which form a Markov Chain $X$ that can be used as shown in Equation (2.28). After the "burn-in" / pre-equilibration period, samples can be treated as coming from the target distribution $P(x)$.

**Example** As an example I will show how MH can be used to approximate a modified continuous normal distribution

$$\pi(x) = \sin^2(x) * \sin^2(2x) * \mathcal{N}(\mu = 0, \sigma^2 = 1) \tag{2.32}$$

using the continuous uniform distribution as the proposal distribution

$$\mathcal{U}_{[x-\alpha, x+\alpha]}, \tag{2.33}$$

whose width is controlled by the parameter $\alpha$. The basic MH algorithm is shown in Algorithm 1. In Figure 2.1 I have plotted the histogram of results generated using the algorithm with parameters $N = 5e4$ and $\alpha = 2$. The exact code used to generate this histogram is available in supplementary material Listing D.1.

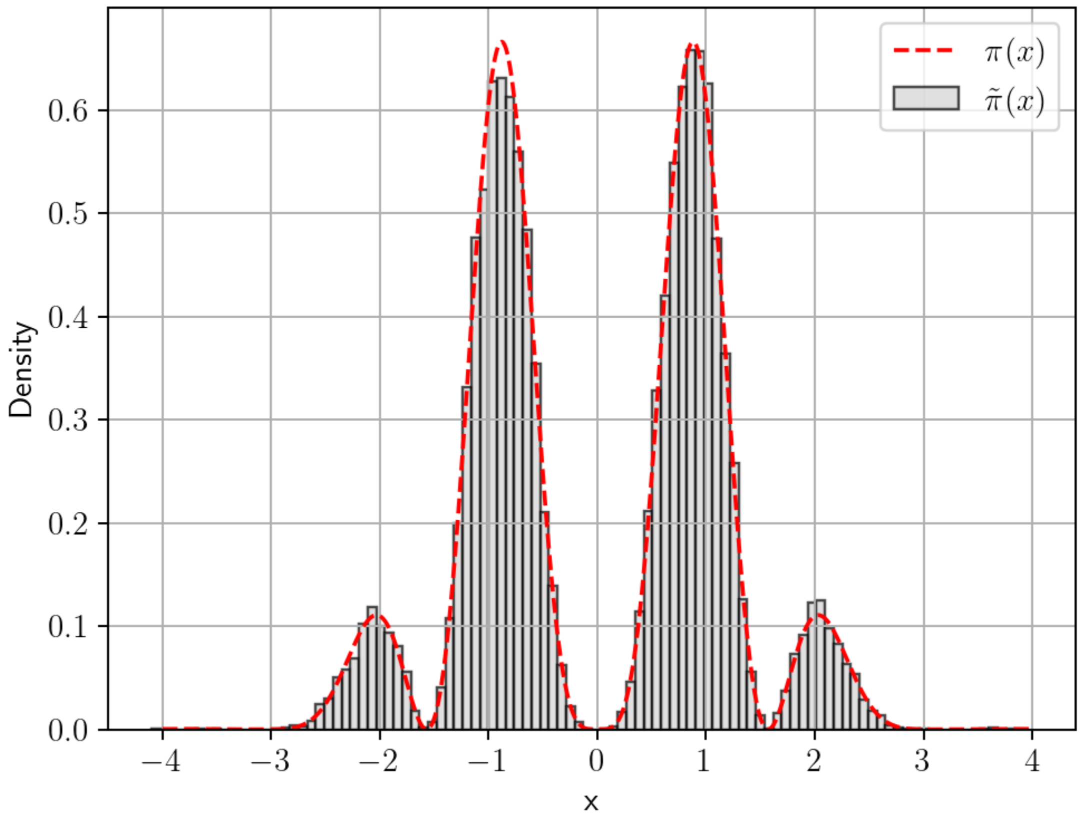

Figure 2.1: Estimation of the distribution $\pi(x)$ in Equation (2.32) using a basic MH algorithm.

**Considerations** Obtaining independent samples from the Markov Chain requires only taking every $N$th sample, where $N$ is sufficiently large that $x_i$ and $x_{i+N}$ are not correlated. To reach an ergodic state requires pre-equilibration and/or "burn-in". This process involves throwing away $M$ initial samples, such that $M$ is large enough that the $x_{N+1}$ sample is independent of the initial state. The choice of proposal distribution also has an effect on the performance of the algorithm, but is not as dominant as in importance sampling.

If the reader still has questions about Monte Carlo Metropolis-Hastings (MCMH) I would recommend consulting[11].

In this section, I have covered how a PI formulation evaluated using a MCMH approach can be used to obtain properties of interest from a QM system. In the next section I will cover the background for the second main approach for calculating properties of interest.

## 2.3 Vibrational Electronic Coupled Cluster

The vibrational electronic coupled cluster (VECC) method is quite different from what we've seen so far in Section 2.1. Rather than using a stochastic approach to obtain thermal equilibrium properties, in VECC one parameterizes a wave function (or reduced density matrices in thermal approaches) and solves for deterministic equations of motion (EOM). The starting point for both problems is the same and is based on a vibronic Hamiltonian.

Coupled Cluster methods are widely used in electronic structure theory, although there are also some results for single surface vibrational problems. However, the approach has not been used before for vibronic Hamiltonians involving multiple electronic states[5]. The detailed so-called Normal Ordered Exponential (NOE) Theory that forms the basis for this work is described elsewhere, in a paper that discusses both real and imaginary time propagation schemes [6]. The extension to vibronic theory is an ongoing problem. B. Songhao, M. Nooijen, and I expect to submit a paper, detailing this extension, for publication in 2023. It is important to note that the majority of the theoretical work has been done by B. Songhao and M. Nooijen.

Here we will introduce the primary ideas. Coupled Cluster theory in the context of electronic structure theory will be reviewed briefly, and then we will set the stage for applications to vibrational and vibronic problems. There are a number of possible variations to VECC and the details concerning each one of these approaches would be cumbersome to derive and implement. For this reason we developed a computer-aided approach to derive and implement equations, further described in Chapter 5. In Chapter 6 we apply a particular variation of VECC to calculate time-correlation functions, which can be applied to simulate vibronic spectra. These spectra, simulated using VECC, are compared to experimental spectra, and benchmarked against other theoretical chemistry software packages.

### 2.3.1 Coupled Cluster theory in electronic structure theory and generalizations to vibrational problems

CC theory is most widely applied in the field of electronic structure theory. It was first formulated in the mid 1960's [12, 13] and has seen many developments since, see e.g. [14, 15, 16]. It is currently implemented in most advanced electronic structure packages, and, perusing a number of highly technical developments, it can be applied to large systems contain hundreds of atoms [17, 18, 19, 20, 21, 22] . In this synopsis we will limit ourselves to the essentials while preparing the ground for applications to a new field of study: Vibronic problems.

[5]Except for unpublished (undergraduate) work in the Nooijen group.

**Second Quantization** Electronic CC theory is cast in the language of second quantization using Fermionic annihilation and creation operators that obey anticommutation relations, e.g.

$$\left\{\hat{p}^\dagger, \hat{q}^\dagger\right\} = 0; \quad \left\{\hat{p}^\dagger, \hat{q}\right\} = \delta_{pq} \tag{2.34}$$

A creation operator can be characterized by its action on a Slater determinant, e.g. [15]

$$\hat{p}^\dagger \left|\phi_q \cdots \phi_s\right\rangle = \left|\phi_p \phi_q \cdots \phi_s\right\rangle, \tag{2.35}$$

while conversely an annihilation operator removes an orbital from a determinant (or yields zero), e.g.

$$\hat{p} \left|\phi_p \phi_q \cdots \phi_s\right\rangle = \left|\phi_q \cdots \phi_s\right\rangle. \tag{2.36}$$

A Slater determinant may then be written as a series of creation operators acting on a vacuum state

$$\hat{a}_p^\dagger \hat{a}_q^\dagger \cdots \hat{a}_s^\dagger \left|\right\rangle = \left|\phi_p \phi_q \cdots \phi_s\right\rangle. \tag{2.37}$$

Single reference CC theory is built on a reference determinant $|\Phi_0\rangle$, usually obtained from a preceding mean-field Hartree-Fock (HF) calculation, that provides a partitioning into occupied orbitals, labelled $i, j, \cdots$ and virtual orbitals $a, b, \cdots$. The operators $\hat{a}^\dagger, \hat{i}$ create excitations out of the reference determinant, and are referred to as (quasiparticle) q-creation operators. Conversely q-annihilation operators $\hat{i}^\dagger, \hat{a}$ annihilate the reference state

$$\hat{i}^\dagger \left|\Phi_0\right\rangle = \hat{a} \left|\Phi_0\right\rangle = 0. \tag{2.38}$$

**Electonic Coupled Cluster** Starting with the SE

$$\hat{H} \left|\Psi\right\rangle = E \left|\Psi\right\rangle, \tag{2.39}$$

in CC theory one expresses the wavefunction using an exponential operator acting on a single determinant reference wave function $|\Phi_0\rangle$

$$\left|\Psi\right\rangle = e^{\hat{T}} \left|\Phi_0\right\rangle \tag{2.40}$$

The cluster operator

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \hat{T}_3 + \cdots, \tag{2.41}$$

consists of so-called singles (S), doubles (D) and higher-order excitation operators, where

$$\hat{T}_1 = \sum_{a,i} t_i^a \hat{a}^\dagger \hat{i}; \qquad \hat{T}_2 = \frac{1}{4} \sum_{a,b,ij} t_{ij}^{ab} \hat{a}^\dagger \hat{i} \hat{b}^\dagger \hat{j}; \quad \cdots; \tag{2.42}$$

This yields the parameterized SE which is to be solved for the amplitudes $t_i^a, t_{ij}^{ab}, \cdots$

$$\hat{H} e^{\hat{T}} \left|\Phi_0\right\rangle = e^{\hat{T}} \left|\Phi_0\right\rangle E \tag{2.43}$$

The CC amplitude equations are derived by multiplying the parameterized SE by $e^{-\hat{T}}$ and projecting against excited bra states $\langle\Phi_\lambda| = \langle\Phi_0|\,\hat{\Omega}_\lambda$,

$$\langle\Phi_0|\,\hat{\Omega}_\lambda e^{-\hat{T}}\hat{H}e^{\hat{T}}\,|\Phi_0\rangle = 0 \tag{2.44}$$

The energy can be obtained as $E = \langle\Phi_0|\,\hat{H}e^{\hat{T}}\,|\Phi_0\rangle$. The key idea underlying CC is the exponential parameterization, which yields size-consistent approximations [23, 24]. This parameterization is primarily responsible for the high accuracy of the approach, in particular the CCSD(T) approach in which a perturbative approximation is made for the numerous triple excitations. Important for our purposes, the evaluation of the amplitude equations proceeds through an application of Wick's theorem, which implies that products of operators are written in normal ordering with respect to the reference state. In a normal ordered product the q-annihilation operators are moved to the right of q-creation operators, using essentially the anticommutation relations, and as a result only fully contracted, or *fully paired* terms survive in the amplitude equations[15]. This is the essence of the approach that will be transplanted to the vibrational problem.

Traditionally, CC theory has been used for electronic ground states that are fairly well characterized by a single determinant. Further extensions include so-called EOM variations which extend the approach to excited states, while lots of work has been done on further generalizations to so-called multireference situations, where the single determinant reference state is not very suitable (for an overview see [25]). In the context of this thesis, another important advance is the use of exponential parameterizations to obtain thermal properties and partition functions using a direct calculation, and not using a sum-over-states (SOS) formulation. The recent work by S. Bao and M. Nooijen [6] is particularly relevant in this context. In their thermal approaches one generalizes the notion of normal ordering and defines contractions in terms of the thermal populations of a simpler one-body hamiltonian. The key features of the fully contracted working equations of CC theory remain valid, but in addition one integrates amplitudes over real or imaginary time. An important complication in the theory, is that there is no partitioning into occupied and virtual orbitals, and the various substitution operators (rather than excitation operators) $\hat{p}^\dagger, \hat{q}$ no longer commute. As a result, one introduces Normal Ordered Exponential (NOE) operators [26], and the amplitude equations are more complicated to derive.

**Vibrational Coupled Cluster** Coupled Cluster theory can be generalized to vibrational problems. In recent work by Hirata and Feauchaux [27, 28] both ground and excited state calculations have been explored in a single surface Born-Oppenheimer context. Earlier work by Prasad et al. is also of interest [29]. The theory is closely analogous then to electronic structure theory, with a few important changes. The annihilation and creation operators refer to the ladder operators for a harmonic approximation in a dimensionless parameterization, and satisfy commutation rather than anticommutation relations. This is an important simplification compared to electronic structure theory as it means that operators can be permuted (especially in a normal ordered product) without a sign change. For the most straightforward applications, the reference state is taken to be the ground state of a harmonic oscillator, and annihilation operators annihilate the reference state. In this thesis we will refer to the bosonic operators corresponding to normal modes simply as operators $\hat{i}, \hat{j}^\dagger$, satisfying

$$\left[\hat{i}^\dagger, \hat{j}^\dagger\right] = 0; \quad \left[\hat{i}, \hat{j}^\dagger\right] = \delta_{ij}. \tag{2.45}$$

The vibrational CC Ground state wave function can be parameterized in terms of operators that contain $1, 2$, or in general $N$ creation operators

$$\hat{T}^n = \frac{1}{n!}\sum t^{i_1, i_2, \ldots, i_n} \hat{i}_1^\dagger \hat{i}_2^\dagger \cdots \hat{i}_n^\dagger.$$

Interestingly, in the theory, one does not introduce a basis set. All states are parameterized by second quantized operators. This feature is responsible for the potentially high efficiency of the approach.

For single surface Hamiltonians the basic theory is straightforward. For spectroscopy purposes it is most convenient to solve the time-dependent Schrödinger equation (TDSE)

$$i\frac{\mathrm{d}\left|\Psi\right\rangle}{\mathrm{d}\tau} = \hat{H}\left|\Psi(\tau)\right\rangle, \tag{2.46}$$

for a wavefunction expressed using the vibrational cluster operator ($\hat{T} = \hat{T}^0 + \hat{T}^1 + \hat{T}^2 + \cdots$)

$$\left|\Psi(\tau)\right\rangle = e^{\hat{T}(\tau)}\left|\phi(\tau = 0)\right\rangle, \tag{2.47}$$

and finally calculate a time correlation function $C(\tau)$:

$$C(\tau) = \langle\phi(\tau = 0)|e^{\hat{T}(\tau)}|\phi(\tau = 0)\rangle = e^{t^0}. \tag{2.48}$$

The EOM for the t-amplitudes can be formulated, after multiplication by $e^{-\hat{T}}$ and projection, as

$$\frac{\mathrm{d}t_\lambda}{\mathrm{d}\tau} = \langle\phi_0|\hat{\Omega}_\lambda\frac{\mathrm{d}\hat{T}}{\mathrm{d}\tau}|\phi_0\rangle = \langle\phi_0|\hat{\Omega}_\lambda e^{-\hat{T}}\hat{H}e^{\hat{T}}|\phi_0\rangle \tag{2.49}$$

These equations can be propagated from suitable initial conditions, and most often one would use $|\phi(\tau = 0)\rangle = |\phi_0\rangle$.

**Thermal Properties** The theory is substantially more complicated if one aims to calculate thermal properties directly, and we refer elsewhere for a discussion. Similar to the electronic structure case, to parameterize the thermal density matrix one has to introduce operators that have both annihilation and creation operators, for example

$$\hat{T}_1^1 = \sum_{i,j} t_j^i \hat{i}^\dagger \hat{j}, \quad \hat{T}_1^2 = \sum_{i,j,k} t_k^{ij} \hat{i}^\dagger \hat{j}^\dagger \hat{k} \tag{2.50}$$

Moreover, the contractions that enter thermal Wick's theorem can have non-integer values and in the NOE approach by M. Nooijen and S. Bao[6] a normal ordered exponential operator is used to parameterize the normal ordered thermal many-body density. It will take us too far afield to discuss this approach in detail, but the essential take away, in the context of this thesis, is that the equations for the thermal theory, in particular, are hard to derive and implement by hand. For this reason it is attractive to use computer software to derive the equations. This computer-aided approach has a long history in both chemistry and physics, see [30] as an early example, and [31] as a further example. Likewise, to explore the methods considered in this thesis such a tool is very valuable, and this is discussed in Chapter 5 of the thesis. The code generator discussed in Chapter 5 can handle both "thermal and spectroscopic" cases by setting a few options in the program. The more general thermal procedure is a major complication, however, in the development of the code-generator, and will be discussed at length in Chapter 5.

### 2.3.2 Vibrational Electronic Coupled Cluster Theory

Absorption, emission, and fluorescence spectra are commonplace in many chemistry research fields. They can be used to identify various properties, such as: characterization of chemical compounds, the presence of free electrons, what functional groups are present, double bonds, and the geometric structure of compounds. Spectra can be generated through a variety of methods, both physically and theoretically. For vibronic spectroscopy, the calculation of spectra is most conveniently done in a time dependent fashion. Given a wave function at some initial time, e.g. $|\Psi_0\rangle$ one propagates this wave function in time, formally,

$$|\Psi(\tau)\rangle = e^{-i\hat{H}\tau}\,|\Psi_0\rangle$$

and calculates the time-autocorrelation function auto-correlation function (ACF).

$$C(\tau) = \langle\Psi_0|e^{-i\hat{H}\tau}|\Psi_0\rangle \tag{2.51}$$

Taking a Fourier transform of the autocorrelation function generates the spectrum of interest.

An example auto-correlation function (ACF) is plotted in Figure 2.2[6]. Applying the Fourier transform[7] to the $x, y$ points of the ACF produces a new set of $x, y$ points. These can then be plotted, as in Figure 2.3, and interpreted as a vibrationally-resolved electronic spectra.

[6]A vibronic model of $H_2O$ was used to generate this figure. More details can be found in Section 6.3.4.
[7]Technical details of how the Fourier transform was performed are covered in Section 6.3.2

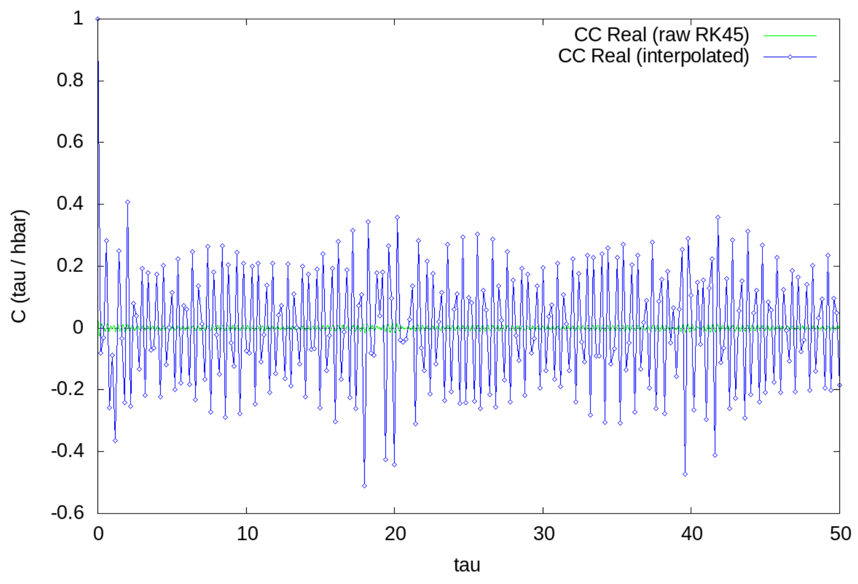

Figure 2.2: Example auto-correlation function (ACF).

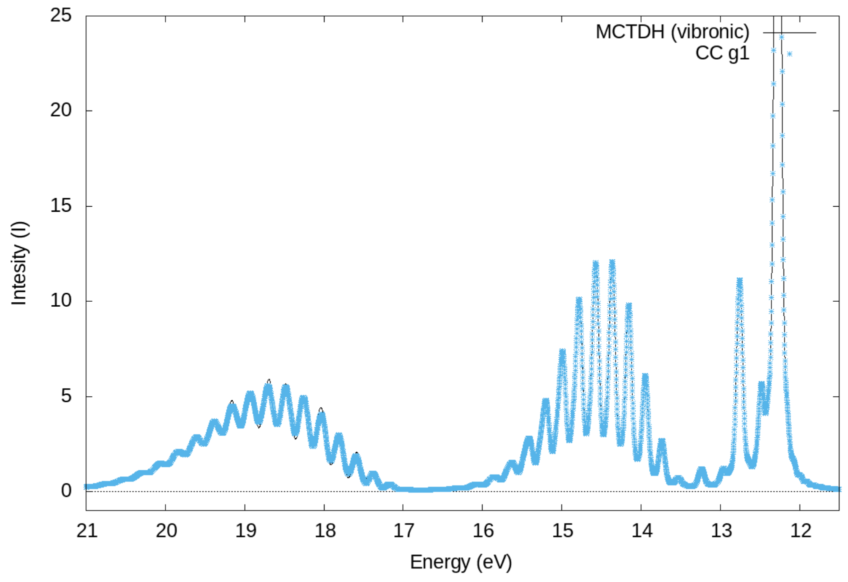

Figure 2.3: Example electronic spectra.

For vibronic problems of interest the initial wave function is expressed as

$$|\Phi_0\rangle = \sum_b |0, b\rangle \, X_b \tag{2.52}$$

in which the label $b$ indexes the diabatic electronic states[8] , and 0 indicates the ground vibrational level in the electronic ground state, as described by a multimodal harmonic oscillator.

[8]Diabatic states are covered in Sections 3.2 and 3.3.

In the current work we focus on absorption spectroscopy from the ground vibrational state, and use the diabatic Condon approximation, in which the electronic transition dipoles $X_b$ are treated as constants calculated at the ground state equilibrium geometry. In the context of VECC it is most flexible to propagate each electronic component individually

$$|\Psi_b(\tau)\rangle = e^{-i\hat{\mathbf{H}}\tau}\,|0,b\rangle \tag{2.53}$$

which leads to population of the other electronic states over time. Then cross-correlation functions can be calculated

$$\langle 0,a|\Psi_b(\tau)\rangle = \langle 0,a|e^{-i\hat{\mathbf{H}}\tau}|0,b\rangle = U_{ab}(\tau). \tag{2.54}$$

One can then calculate the overall ACF

$$ACF(\tau) = \sum_{a,b} X_a \mathcal{U}_{ab}(\tau) X_b, \tag{2.55}$$

by numerical integration over the parameter $\tau$, that can be chosen to be time $t$. A similar construction can also be employed to calculate thermal properties for vibronic Hamiltonian and one integrates over inverse temperature $\beta$.

In what follows, we focus on the problem of obtaining operator-based parameterizations for a general wave function $\Psi(\tau)$, starting from some initial state $\Psi_0$. The wave function for multiple electronic surfaces can easily be parameterized in a linear fashion

$$|\Psi(\tau)\rangle = \sum_b \hat{Z}_b(\tau)\,|0,b\rangle \tag{2.56}$$

Here $a, b$, and $c$ will indicate the various electronic surfaces, as above. The equation of motion can then be obtained by substituting in the time-dependent Schrödinger equation and projecting on a suitable set of states, expressed as operators,

$$\sum_b \langle 0,a|\,\hat{\Omega}_\lambda \frac{d\hat{Z}_b}{d\tau}\,|0,b\rangle = \sum_b \langle 0,a|\,\hat{\Omega}_\lambda \hat{\mathbf{H}} \hat{Z}_b(\tau)\,|0,b\rangle \tag{2.57}$$

Here and in what follows the surface-specific part is represented by boldface $\hat{\mathbf{H}}$, our convention to represent the matrix aspect in the diabatic electronic basis. The evaluation of such terms is relatively straightforward, but the accuracy of results is not very promising. We need to incorporate an exponential parameterization into the definition of the wave function. However, it is not obvious how to do this. Below we will list a number of possibilities. Each of these approaches constitutes a different set of equations, that would have to be implemented in a computer code that can propagate the wave function parameters and then obtain the cross-correlation functions. The purpose of the research presented in the thesis is to facilitate the derivation of lengthy equations and produce computational procedures that can be organized in a final computer code. Let us just provide a small sample of possibilities that have been explored over time.

$$|\Psi(\tau)\rangle = e^{\hat{\mathbf{T}}} |\Psi_0\rangle \tag{2.58}$$

In the first expression $\hat{\mathbf{T}}$ has a similar "electronic matrix" form as the vibronic Hamiltonian. This formulation has formal appeal, but it leads to complicated equations that are numerically not very well behaved. Another possibility is to use

$$|\Psi(\tau)\rangle = e^{\hat{T}(\tau)}\hat{\mathbf{Z}}(\tau) |\Psi_0\rangle \tag{2.59}$$

In this equation $\hat{T}$ represents a pure vibrational operator, that is the same for each electronic surface. In this approach one has to develop separate equations for a state-averaged operator $\hat{T}$, in the spirit of Ehrenfest dynamics, and then propagate also the equations for the linear parameters in the operator $\hat{\mathbf{Z}}$. A third possibility removes the state-averaged aspect and uses

$$|\Psi(\tau)\rangle = \sum_b e^{\hat{T}_b(\tau)}\hat{Z}_b(\tau) |0, b\rangle \tag{2.60}$$

Again one has to posit separate equations for the state-specific operators $\hat{T}_b(\tau)$ and $\hat{Z}_b(\tau)$. The reader may ask, why one would not use an ansatz like

$$|\Psi(\tau)\rangle = \sum_b e^{\hat{T}_b(\tau)} |0, b\rangle \tag{2.61}$$

This would be possible as a unitary formulation, but in our formulation the initial contribution from many states would be close to zero, at least initially, and this leads to numerical instabilities. The use of intrinsically Unitary formulations makes it hard to calculate the overlap with the initial states $\langle 0, a|$.

Clearly this state of uncertainty leaves many questions and many details unanswered. This is precisely the state of affairs we face and the path forward is to develop computer-aided derivations of detailed working equations, keeping in mind the flexibility such a tool would need to have, and including the possibility to generalize methodology to thermal problems. In Chapter 4 we discuss such a tool, keeping the formal theory to a minimum. In chapter 5 a specific version of the theory is considered and put to the test.

## 2.4 Realistic Computations

It is important to touch upon decisions made during the development of the theory and software in this thesis. Of note is the computational cost of the algorithms within. Here I highlight the major bottlenecks as well as the steps taken to tackle them and any bottlenecks still remaining.

In terms of the PIMC method in Chapter 4, the dominant cost is the evaluation of the matrices $\mathbb{M}$, $\mathbb{O}$, and $\tilde{\mathbb{O}}$. In my profiling of the Python code, roughly 75–85% of the runtime is matrix evaluation. The scaling of this also grows fastest with system size $A$ and $N$. Therefore the priority here is twofold, efficient matrix evaluation and diagonalization. Both of these tasks were accomplished with low-level software libraries: BLAS, LaPACK, etc. [32, 33]

For the VECC method in Chapter 5, there are two aspects to consider: the *generation* of the LaTeX documents and Python code, and the computational evaluation of the Python code. For all input parameter choices that I have used, the cost of *generating* the LaTeX documents and the Python code is negligible, on the order of a few minutes **at most**, so this is not a concern. The *evaluation* of the Python code dominates the runtime[9] and motivates much of the work in Chapter 5. We employ a variety of methods to tackle this problem: sparse symmetrization, collating equivalent permutable terms, and employing optimized `einsum()` function calls using a third-party library. While the evaluation of the residual terms could potentially benefit from being implemented in a compiled language[10] such as FORTRAN, C, or C++, this would incur the cost of dozens of weeks of development time by an experienced programmer. Additionally these performance benefits are also not guaranteed. An implementation in a new programming language risks introducing many bugs, which are hard to identify due to the approximate nature of the theory. To verify the new implementation, it would be necessary to identically compare with the prior code at a low level, which would be a slow arduous task.

As the saying goes "premature optimization is the root of all evil"[11]. With regards to optimizations of the computational evaluation of integrals present in this thesis my focus was towards the dominant contributing factors. Balancing development time with computing gains was a consideration. Standard matrix operations, like addition, multiplication and diagonalization represent the majority of the computational cost in methods used in this thesis. These operations can be computed using low level C libraries such as IntelMKL or OpenBLAS [34, 35, 36, 37, 38]. In Python this can be done by linking to these libraries through Numpy [39]. In this way I am able to get the best of both worlds, the development speed and ease of use of a scripting language and the computational efficiency of a compiled language.

[9]The runtime to propagate the wavefunction and hence produce a spectra.

[10]As opposed to a interpreted language like Python.

[11]Usually attributed to Donald Knuth, although possibly originated in a book from the 1960's.

# Chapter 3

# Vibronic Models

## 3.1 Quantum Mechanical Models

The (non-relativistic) Hamiltonian for molecular systems, in the absence of external fields, is easy to write down as the sum of kinetic energy plus potential energy, extending the sums over all particles in the system (electrons plus nuclei):

$$\hat{H} = \sum_a \frac{P_a^2}{2M_a} + \sum_{a<b} \frac{q_a q_b}{|r_a - r_b|} \tag{3.1}$$

The equations that determine phenomena in chemistry are, first, the time-dependent, and second, the time-independent Schrödinger equation (SE):

$$\hat{H}\Psi = E\Psi. \tag{3.2}$$

However, those equations involving states that depend generically on both electronic and nuclear DoF are exceedingly hard to solve.

In most of quantum chemistry, the BO approximation is invoked, and one uses electronic structure calculations to obtain ground and excited electronic states at relevant particular nuclear configurations. A wide variety of methods are available for this purpose, e.g. semi-empirical, Density Functional and ab initio approaches like CC, Møller–Plesset perturbation theory (MP), and variations thereof[40]. The BO approximation is often combined with additional harmonic and rigid rotor (RR) approximations, such that in most of traditional quantum chemistry, one optimizes geometries (i.e. finds stationary points on BO surfaces), and then includes: nuclear eigenstates using the (polyatomic) harmonic oscillator (HO) for vibrations, rotational motion using RR, and translational motion using the "particle in a box" model. In this fashion, one can estimate thermal free energies for molecular systems, approximating the more rigorous sum-over-states (SOS) approach from statistical mechanics.

Likewise, in electronic spectroscopies, one solves for minima on ground and electronic surfaces, and invokes the harmonic Frank–Condon (FC) approach to obtain the vibrational fine structure on electronic spectra. This Rigid-Rotor-Harmonic-Oscillator-Born-Oppenheimer

(RR-HO-BO) approach is widely used for rigid molecular systems and can be combined with most electronic structure methods of choice.

However, these methods break down when electronic surfaces are close together at relevant nuclear geometries. More accurate wavefunctions describing combined electronic and nuclear DoF require the inclusion of a small number of close-lying adiabatic states, and including the so-called non-adiabatic coupling between such states. Such phenomena are well known in areas of electronic spectroscopy and in discussions of photochemical reactions, in which transitions between electronic states are of key interest[41].

As part of ongoing investigations in the Nooijen research group, such breakdown of single surface BO approximation phenomena are also relevant when discussing the thermochemistry of a manifold of low-lying states[6]. The most straightforward generalization of the RR-HO-BO approach is the construction of vibronic models in a small set of diabatic states that describe the electronic structure in a limited region of nuclear configuration space. The key aspect is that the Hamiltonian that describes coupled nuclear-electronic states is described by a potential-energy matrix, rather than a scalar function. We will limit ourselves in this work to simple quadratic models, in close analogy to harmonic oscillators. In addition, rotational effects are not considered.

In this chapter, we will outline the construction of vibronic models and their mathematical structure using the ladder operators, familiar for the harmonic oscillator. These models are the starting point for novel operator approaches to discuss both nuclear dynamics and spectroscopy, and statistical mechanics. These new methods are not the focus of the thesis, however. We will discuss the mathematical structure of equations and develop an equation and code generator to derive relevant equations in a quite abstract but generic fashion. Some example applications will be presented.

In Section 3.2 we begin by working through the standard (BO) approach to the electronic Hamiltonian $\hat{H}^{\text{el}}$. Numerical challenges related to computing specific factors motivate the use of a different representation, the diabatic representation, as described in Section 3.3. The specific mathematical definition of a vibronic model, in the diabatic representation, (used throughout the thesis) is presented in Section 3.4. Finally, a step-by-step example of a diabatization process is presented in Section 3.5. This is presented for those unfamiliar with vibronic models so as to ground the discussion presented in Sections 3.3 and 3.4.

To understand further chapters the reader need only familiarize themselves with the mathematical definitions in Section 3.4.

## 3.2 Adiabatic states

To discuss the complications associated with coupled nuclear-electronic states we first need to define the *adiabatic* states. We will then look at the evaluation of the full molecular SE, in terms of those adiabatic states, and see (in Equations (3.16a) to (3.16d)) how problematic nonadiabatic coupling term (NACT) directly arise from the definition of these adiabatic states.

The molecular Hamiltonian $\hat{H}$ is comprised of electronic and nuclear contributions to the kinetic and potential energies, with corresponding electronic ($\boldsymbol{r}$), and nuclear ($\boldsymbol{R}$) coordinates. The full molecular SE is as follows:

$$\hat{H}(\boldsymbol{r}, \boldsymbol{R})\Psi_n(\boldsymbol{r}, \boldsymbol{R}) = E_n\Psi_n(\boldsymbol{r}, \boldsymbol{R}). \tag{3.3}$$

### 3.2.1 Clamped Nuclei

The clamped nuclei approximation assumes that due to the difference in the time scale of electronic and nuclear motion, nuclei being much heavier than electrons, these two degrees of freedom can be separated and solved for separately. The nuclei are therefore fixed relative to the electrons. The molecular Hamiltonian can be written as the sum of two terms

$$\hat{H} = \hat{T}^N + \hat{H}^{el}, \tag{3.4}$$

where the first term is the nuclear kinetic energy and the second term refers to the electronic Hamiltonian:

$$\hat{H}^{el} = \hat{T}^{el} + \hat{V}^{N,N} + \hat{V}^{N,e} + \hat{V}^{e,e}, \tag{3.5}$$

which includes all contributions, except $\hat{T}^N$. Starting with fixed nuclei, electronic structure theory states that

$$\hat{H}^{\text{el}}(\boldsymbol{r}; \boldsymbol{R})\psi_\lambda(\boldsymbol{r}; \boldsymbol{R}) = E_\lambda(\boldsymbol{R})\psi_\lambda(\boldsymbol{r}; \boldsymbol{R}). \tag{3.6}$$

For emphasis in Equation (3.6) I have clearly indicated the electronic $\boldsymbol{r}$ and nuclear coordinates $\boldsymbol{R}$, on $\hat{H}^{\text{el}}$. The parametric dependence on the fixed nuclear coordinates $\boldsymbol{R}$ is indicated by the use of a semicolon: $\psi_\lambda(\boldsymbol{r}; \boldsymbol{R})$. The electronic wavefunctions $\psi_\lambda(\boldsymbol{r}; \boldsymbol{R})$ are approximate eigenfunctions of $\hat{H}^{\text{el}}$ and are referred to as the *adiabatic* states [42].

### 3.2.2 Born-Oppenheimer and Born-Huang approximations

A general parametrization of the complete set of eigenstates (including nuclear motion) is given by

$$\Psi(\boldsymbol{r}, \boldsymbol{R}) = \sum_\lambda \psi_\lambda(\boldsymbol{r}; \boldsymbol{R})\chi_\lambda(\boldsymbol{R}). \tag{3.7}$$

In practice a finite number of states $K$ is chosen, and thus $\Psi(\boldsymbol{r}, \boldsymbol{R})$ is expressed in terms of a truncated basis of $K$ eigenstates. If one were to choose to represent the wavefunction using only a **single** term ($\lambda \in K$):

$$\Psi(\boldsymbol{r}, \boldsymbol{R}) \approx \psi(\boldsymbol{r}; \boldsymbol{R})\chi(\boldsymbol{R}) \tag{3.8}$$

this would be the BO approximation. Using the full expansion with all $K$ states is the *Born-Huang* expansion.

## Electronic eigenstates

We will tackle the full molecular SE for a single eigenstate of $\hat{H}$

$$\hat{H}\Psi(\boldsymbol{r}, \boldsymbol{R}) = E\Psi(\boldsymbol{r}, \boldsymbol{R}), \tag{3.9}$$

where $E$ is a constant and

$$\hat{H} = \hat{T}^{\mathrm{N}} + \hat{H}^{\mathrm{el}}. \tag{3.10}$$

We start by directly evaluating the SE using Equation (3.8)

$$(\hat{T}^{\mathrm{N}} + \hat{H}^{\mathrm{el}}) \sum_{\lambda} \psi_{\lambda}(\boldsymbol{r}; \boldsymbol{R})\chi_{\lambda}(\boldsymbol{R}) = E \sum_{\lambda} \psi_{\lambda}(\boldsymbol{r}; \boldsymbol{R})\chi_{\lambda}(\boldsymbol{R}). \tag{3.11}$$

Expanding the left side

$$= (\frac{1}{2M_a}\boldsymbol{\nabla}_a \cdot \boldsymbol{\nabla}_a + \hat{H}^{\mathrm{el}}) \sum_{\lambda} \psi_{\lambda}(\boldsymbol{r}; \boldsymbol{R})\chi_{\lambda}(\boldsymbol{R}) \tag{3.12}$$

$$= \sum_{\lambda} \left(\tilde{A}_{\lambda} + \tilde{B}_{\lambda} + \tilde{C}_{\lambda} + \tilde{D}_{\lambda}\right), \tag{3.13}$$

with the four terms

$$\tilde{A}_{\lambda} = \sum_{a} \frac{1}{2M_a} \chi_{\lambda}(\boldsymbol{R}) \nabla_a^2 \psi_{\lambda}(\boldsymbol{r}; \boldsymbol{R}), \tag{3.14a}$$

$$\tilde{B}_{\lambda} = \sum_{a} \frac{1}{2M_a} \psi_{\lambda}(\boldsymbol{r}; \boldsymbol{R}) \nabla_a^2 \chi_{\lambda}(\boldsymbol{R}), \tag{3.14b}$$

$$\tilde{C}_{\lambda} = \sum_{a} \frac{1}{2M_a} \boldsymbol{\nabla}_a \psi_{\lambda}(\boldsymbol{r}; \boldsymbol{R}) \boldsymbol{\nabla}_a \chi_{\lambda}(\boldsymbol{R}), \tag{3.14c}$$

$$\tilde{D}_{\lambda} = E_{\lambda}(\boldsymbol{R}) \psi_{\lambda}(\boldsymbol{r}; \boldsymbol{R}) \chi_{\lambda}(\boldsymbol{R}), \tag{3.14d}$$

where $A$ labels the nuclei and $\lambda$ labels the electronic states. Multiplying Equation (3.13) by $\psi_{\mu}(\boldsymbol{r}; \boldsymbol{R})$ and integrating out the electronic co-ordinates

$$= \sum_{\lambda} \left(A_{\mu\lambda} + B_{\mu\lambda} + C_{\mu\lambda} + D_{\mu\lambda}\right), \tag{3.15}$$

where

$$A_{\mu\lambda} = \int \left( \psi_\mu(\boldsymbol{r};\boldsymbol{R}) \sum_a \frac{1}{M_a} \nabla_a^2 \, \psi_\lambda(\boldsymbol{r};\boldsymbol{R}) \, \mathrm{d}\boldsymbol{r} \right) \chi_\lambda(\boldsymbol{R}), \tag{3.16a}$$

$$B_{\mu\lambda} = \delta_{\mu\lambda} \sum_a \frac{1}{2M_a} \nabla_a^2 \, \chi_\lambda(\boldsymbol{R}), \tag{3.16b}$$

$$C_{\mu\lambda} = \sum_a \left( \int \psi_\mu(\boldsymbol{r};\boldsymbol{R}) \boldsymbol{\nabla}_a \psi_\lambda(\boldsymbol{r};\boldsymbol{R}) \, \mathrm{d}\boldsymbol{r} \right) \frac{-1}{M_a} \boldsymbol{\nabla}_a \chi_\lambda(\boldsymbol{R}), \tag{3.16c}$$

$$D_{\mu\lambda} = \delta_{\mu\lambda} E_\lambda(\boldsymbol{R}) \chi_\lambda(\boldsymbol{R}). \tag{3.16d}$$

There are two major challenges with these equations. First and foremost, are the NACTs: $C_{\mu\lambda}$ in Equation (3.16c) [43]. In the full Born-Huang expansion, the NACTs are difficult to calculate, and incorporate. The second issue, is that the energies $E_{\mu\lambda}(\boldsymbol{R})$ are hard to obtain in compact form. They are poorly defined around conical intersections, and cannot be easily represented by a low-order Taylor series expansion.

As previously stated, in the BO approximation the full wavefunction is approximated by a single $\lambda$ state. In the single state case $\mu = \lambda$, resulting in the NACTs vanishing due to the electronic states being a normalized orthonormal basis set. In most calculations $A$, commonly referred to as the diagonal BO correction, is sufficiently small that it is neglected.

Under the BO approximation, only the $b$ and $D$ remain, resulting in the familiar BO SE

$$\left( \hat{T}^{\mathrm{N}} + E(\boldsymbol{R}) \right) \psi(\boldsymbol{r};\boldsymbol{R}) \chi(\boldsymbol{R}) = E \psi(\boldsymbol{r};\boldsymbol{R}) \chi(\boldsymbol{R}), \tag{3.17}$$

which gives us the energy $E$ of a single eigenstate $\Psi(\boldsymbol{r};\boldsymbol{R})$ of $\hat{H}$. In a similar fashion, this derivation can be extended to multiple eigenstates $\Psi_n(\boldsymbol{r};\boldsymbol{R})$ and corresponding energies $E_n$.

## 3.3 Diabatic States

If one's goal is to include the effects of the NACTs, another approach is necessary. Using a new basis the calculation of these terms becomes a feasible task. These *diabatic* states are derived from a linear combination of specific adiabatic states $\psi_\lambda(\boldsymbol{r};\boldsymbol{R})$:

$$\phi_a(\boldsymbol{r};\boldsymbol{R}) = \sum_\lambda \psi_\lambda(\boldsymbol{r};\boldsymbol{R}) U_{\lambda a}(\boldsymbol{R}), \tag{3.18}$$

where $U_{\lambda a}(\boldsymbol{R})$ is the rotation matrix which depends on $\boldsymbol{R}$. At a reference geometry $\boldsymbol{R}_0$ these states are equivalent

$$\phi_a(\boldsymbol{r};\boldsymbol{R}_0) = \psi_\lambda(\boldsymbol{r};\boldsymbol{R}_0), \tag{3.19}$$

as a consequence of our choice of $U_{\lambda a}(\boldsymbol{R})$ the overlap between diabatic states is minimized

$$\int \mathrm{d}\boldsymbol{r} \, \phi_a(\boldsymbol{r};\boldsymbol{R}_0) \boldsymbol{\nabla} \phi_b(\boldsymbol{r};\boldsymbol{R}) \approx 0 \quad | \quad a \neq b \tag{3.20}$$

In the diabatic basis we also choose to neglect the diagonal BO correction, $A_\lambda$ in Equation (3.15), from our calculation of the electronic Hamiltonian. In this manner, we have addressed the two troublesome terms from Equations (3.16a) to (3.16d)

### 3.3.1 Coupling coefficients

In the new diabatic basis the electronic energies $E_\lambda(\boldsymbol{R})$ become a potential energy matrix

$$\begin{aligned} V_{ab}(\boldsymbol{R}) &= \sum_\lambda U_{\lambda a}(\boldsymbol{R}) E_\lambda(\boldsymbol{R}) U_{\lambda b}(\boldsymbol{R}) \\ &= \underline{\boldsymbol{U}}^\dagger \underline{\boldsymbol{E}} \underline{\boldsymbol{U}}. \end{aligned} \tag{3.21}$$

Now we can express the SE, for nuclear motion, in the diabatic basis

$$\hat{T}^{\mathrm{N}} \chi_a(\boldsymbol{R}) + \sum_b V_{ab}(\boldsymbol{R}) \chi_b(\boldsymbol{R}) = E\, \chi_a(\boldsymbol{R}). \tag{3.22}$$

How does the diabatic basis compare to the previous adiabatic basis? In the diabatic basis we have no cumbersome NACTs. Also our potential energy matrix $V_{ab}(\boldsymbol{R})$ has smooth curves for each matrix element, and as a consequence low-order Taylor series suffice to provide adequate model hamiltonians. However, the disadvantage is that we require extensive calculations to perform the diabatization at many displaced geometries, as seen in Equations (3.25) and (3.26), to obtain the Taylor series expansion.

### 3.3.2 Diabatization scheme

Let us now discuss diabatization, the process by which we compute diabatic states. Recall, that the diabatic states $\phi_a(\boldsymbol{r};\boldsymbol{R})$ are defined in terms of the rotation matrix $U_{\lambda a}(\boldsymbol{R})$. This matrix can be formally defined as

$$\delta_{\mu a} = \sum_\lambda S_{\lambda\mu} U_{\lambda a}(\boldsymbol{R}), \tag{3.23}$$

where

$$S_{\lambda\mu}(\boldsymbol{R}_0 + \Delta\boldsymbol{q}) = \int \mathrm{d}\boldsymbol{r}\, \psi_\lambda(\boldsymbol{r};\boldsymbol{R}_0)\psi_\mu(\boldsymbol{r};\boldsymbol{R}_0 + \Delta\boldsymbol{q}) \tag{3.24}$$

**Overall scheme** I present an example a general approach for calculating a single diabatic state $\phi_a(\boldsymbol{r};\boldsymbol{R})$:

1. Select a set of adiabatic states[1] $\{\psi_1, \psi_2, \ldots, \psi_j, \ldots, \psi_N\}$

[1]It can be the case that only certain excitations are of interest and so then the adiabatic states are chosen appropriately.

2. Calculate the overlap $S_{\lambda\mu}$ between adiabatic states at the ground state geometry $\boldsymbol{R}_0$ and at a displaced geometry along a normal mode $\boldsymbol{R}_0 \pm \Delta\boldsymbol{q}_i$ using Equation (3.24)
3. Calculate $\underline{\boldsymbol{U}}$ which minimizes the difference between $S_{aj}$ and $\delta_{aj}$
4. transform $V_{ab}(\Delta\boldsymbol{q})$
5. Obtain coupling coefficients $E_a^{b,i}$ through numerical differentiation as shown in Equation (3.26)

**Constructing a vibronic model** Recall from Equation (3.21), that the derivation using these new diabatic states results in a diabatic potential energy matrix $V_{ab}(\mathbf{R})$ whose matrix elements were well defined. To produce a vibronic model, vibronic coupling terms are obtained as Taylor series expansions of $V_{ab}$, up to some order, commonly quadratic.

$$V_{ab}(\boldsymbol{q}) = V_{ab}\delta_{ab} + \sum_i g_i^{ab} q_i + \frac{1}{2}\sum_{ij} g_{ij}^{ab} q_i q_j + \cdots . \tag{3.25}$$

The coefficients of the coupling terms are obtained through numerical differentiation, like so:

$$g_i^{ab} = \frac{V_{ab}(\boldsymbol{R}_0 + \Delta\boldsymbol{q}_i) - V_{ab}(\boldsymbol{R}_0 - \Delta\boldsymbol{q}_i)}{2\Delta q_i}. \tag{3.26}$$

The quadratic and higher terms can be derived in a similar fashion. With the understanding of adiabatic states, diabatic states, and the diabatization process, I can now define the Hamiltonian of the vibronic models used in this thesis.

## 3.4 Vibronic Models

When a vibronic model is being discussed in this thesis I am referring to a Hamiltonian of the general form in Equation (3.27), whose numerical values have been obtained through a diabatization procedure as outlined in Section 3.3. This Hamiltonian is expressed in discrete electronic co-ordinates $A$, and continuous vibrational co-ordinates $\hat{q}_j$. Note that $\hat{q}_j$ are dimensionless normal modes[2]. All methods are outlined for models truncated at second order. Although, results are presented only for models truncated at first order terms.

### 3.4.1 Notation

I define the general form of a Hamiltonian as obtained from a diabatization procedure:

$$\hat{H}^{aa'} = E^{aa'} + \left(\frac{1}{2}\sum_j^N \omega_j(\hat{p}_j^2 + \hat{q}_j^2)\right)\delta_{aa'} + \sum_j^N g_j^{aa'}\hat{q}_j + \sum_{jj'}^N g_{jj'}^{aa'}\hat{q}_j\hat{q}_{j'}. \tag{3.27}$$

[2]They have been nondimensionalized in the standard approach. A straightforward example can be found here.

**Chapter 3 Form** In chapter 4 the vibronic model is split in a specific fashion:

$$\hat{H} = \hat{h} + \hat{V}, \tag{3.28}$$

where $\hat{h}$ contains all terms that can be described by harmonic oscillators

$$\hat{h}^{a} = E^{aa} + \frac{1}{2}\sum_{j}^{N}\omega_j(\hat{p}_j^2 + \hat{q}_j^2) + \sum_{j}^{N} g_j^{aa}\hat{q}_j, \tag{3.29}$$

and $\hat{V}$ contains the remaining terms

$$\hat{V}^{aa'} = E^{aa'}(\delta_{aa'} - 1) + \sum_{j}^{N} g_j^{aa'}\hat{q}_j(\delta_{aa'} - 1) + \sum_{jj'}^{N} g_{jj'}^{aa'}\hat{q}_j\hat{q}_{j'}. \tag{3.30}$$

**Chapter 4 Form** In Chapter 5 the vibronic model will be expressed in terms of second quantized operators. By defining the creation and annihilation operators in terms of position $\hat{q}$ and momentum $\hat{p}$ operators [3]:

$$\hat{a}_j^\dagger = \frac{1}{\sqrt{2}}\left(\hat{q}_j + i\hat{p}_j\right) \qquad \hat{a}_j = \frac{1}{\sqrt{2}}\left(\hat{q}_j - i\hat{p}_j\right), \tag{3.31}$$

such that

$$\hat{q}_j = \frac{1}{\sqrt{2}}(\hat{a}_j + \hat{a}_j^\dagger) \qquad \hat{p}_j = \frac{1}{\sqrt{2}i}(\hat{a}_j - \hat{a}_j^\dagger), \tag{3.32}$$

we can express the Hamiltonian in second quantized operators:

$$\begin{aligned}\hat{H}^{aa'} = E^{aa'} &+ \left(\frac{1}{2}\sum_{j}^{N}\omega_j(1 + \hat{a}_j^\dagger\hat{a}_j)\right)\delta_{aa'} \\ &+ \frac{1}{\sqrt{2}}\sum_{j}^{N} g_j^{aa'}(\hat{a}_j + \hat{a}_j^\dagger) + \frac{1}{2}\sum_{jj'}^{N} g_{jj'}^{aa'}(\hat{a}_j + \hat{a}_j^\dagger)(\hat{a}_{j'} + \hat{a}_{j'}^\dagger).\end{aligned} \tag{3.33}$$

The quadratic contribution can be re-written by expanding

$$(\hat{a}_j + \hat{a}_j^\dagger)(\hat{a}_{j'} + \hat{a}_{j'}^\dagger) = \hat{a}_j\hat{a}_{j'} + \hat{a}_j\hat{a}_{j'}^\dagger + \hat{a}_j^\dagger\hat{a}_{j'} + \hat{a}_j^\dagger\hat{a}_{j'}^\dagger, \tag{3.34}$$

and using the commutation relation $[\hat{a}_j, \hat{a}_{j'}^\dagger] = \delta_{jj'}$ to replace $\hat{a}_j\hat{a}_{j'}^\dagger$

$$\hat{a}_j\hat{a}_{j'} + \hat{a}_j\hat{a}_{j'}^\dagger + \hat{a}_j^\dagger\hat{a}_{j'} + \hat{a}_j^\dagger\hat{a}_{j'}^\dagger = \hat{a}_j\hat{a}_{j'} + (\delta_{jj'} + \hat{a}_{j'}^\dagger\hat{a}_j) + \hat{a}_j^\dagger\hat{a}_{j'} + \hat{a}_j^\dagger\hat{a}_{j'}^\dagger, \tag{3.35}$$

[3]Specifically the nondimensionalized $\hat{q}$ and $\hat{p}$.

such that

$$\begin{aligned}&\frac{1}{2}\sum_{jj'}^{N} g_{jj'}^{aa'}(\hat{a}_j + \hat{a}_j^\dagger)(\hat{a}_{j'} + \hat{a}_{j'}^\dagger) \\ &= \frac{1}{2}\sum_{jj'}^{N} g_{jj'}^{aa'}\delta_{jj'} + \frac{1}{2}\sum_{jj'}^{N} g_{jj'}^{aa'}(\hat{a}_j\hat{a}_{j'} + \hat{a}_{j'}^\dagger\hat{a}_j + \hat{a}_j^\dagger\hat{a}_{j'} + \hat{a}_j^\dagger\hat{a}_{j'}^\dagger).\end{aligned} \tag{3.36}$$

Applying this to Equation (3.33) gives a modified Hamiltonian

$$\begin{aligned}\hat{H}^{aa'} &= E^{aa'} + \frac{1}{2}\sum_{jj'}^{N} g_{jj'}^{aa'}\delta_{jj'} \\ &+ \left(\frac{1}{2}\sum_{j}^{N}\omega_j(1 + \hat{a}_j^\dagger\hat{a}_j)\right)\delta_{aa'} \\ &+ \frac{1}{\sqrt{2}}\sum_{j}^{N} g_j^{aa'}(\hat{a}_j + \hat{a}_j^\dagger) + \frac{1}{2}\sum_{jj'}^{N} g_{jj'}^{aa'}(\hat{a}_j\hat{a}_{j'} + \hat{a}_{j'}^\dagger\hat{a}_j + \hat{a}_j^\dagger\hat{a}_{j'} + \hat{a}_j^\dagger\hat{a}_{j'}^\dagger).\end{aligned} \tag{3.37}$$

This is the processing of writing the Hamiltonian in normal order.

### 3.4.2 Displaced Model

Here I present the displaced model from my previous work [44]. This model is a linear vibronic model with two electronic surfaces $A$, and two normal modes $q$. The off-diagonal coupling terms $\hat{q}_2$ can be tuned by changing the $\gamma$ coefficient. This allows me to explore the system over a range of values: from having no coupling $\gamma = 0$ and being a FC system where the $\hat{q}_1$ mode dominates, to small to medium coupling $0.01 \leq \gamma \leq 0.1$ where neither $\hat{q}_1$ or $\hat{q}_2$ dominates, to very strong coupling $0.1 \leq \gamma \leq 0.2$ where the $\hat{q}_2$ dominates. This system is small enough that SOS can be performed to check the results.

This system is described by the Hamiltonian

$$\begin{aligned}\hat{H} &= \hat{h} + \hat{V} \\ &= \begin{bmatrix} E^a + \hat{h}_o + \lambda\hat{q}_1 & 0 \\ 0 & E^b + \hat{h}_o - \lambda\hat{q}_1\end{bmatrix} + \gamma_i\begin{bmatrix} 0 & \hat{q}_2 \\ \hat{q}_2 & 0\end{bmatrix}.\end{aligned} \tag{3.38}$$

You can see the JSON file format in Listing E.3 and the `.op` format in Listing E.4.

### 3.4.3 Jahn-Teller Model

Here I present a Jahn-Teller model also from my previous work[44]. This model is a linear vibronic model with two electronic surfaces $A$, and two normal modes $q$. It has the important property that the PES is symmetric in both modes. The steepness of the curvature of the well in the PES can be tuned by the $\gamma$ coefficient. Again we can explore the system over a range of values: from having no coupling $\gamma = 0$ and being a simple harmonic oscillator, to small to medium coupling $0.01 \leq \gamma \leq 0.1$, to very strong coupling $0.1 \leq \gamma \leq 0.2$. This system is small enough that SOS can be performed to check the results.

This system is described by the Hamiltonian

$$\hat{H} = \hat{h} + \hat{V} \tag{3.39}$$

$$= \begin{bmatrix} E_i + \hat{h}_{\text{o}} + \lambda_i \hat{q}_1 & 0 \\ 0 & E_i + \hat{h}_{\text{o}} - \lambda_i \hat{q}_1 \end{bmatrix} + \lambda_i \begin{bmatrix} 0 & \hat{q}_2 \\ \hat{q}_2 & 0 \end{bmatrix}. \tag{3.40}$$

You can see the JSON file format in Listing E.5 and the `.op` format in Listing E.6.

## 3.5 Real World Diabatization Example

An explanation of the automated process of creating a vibronic model will be presented using water ($H_2O$) as an example. The process is a slightly modified version of the approach used in Aranda and Santoro's work[45]. Code to automate the task of preforming this diabatization was developed in collaboration with Benny Chen.

Work is ongoing to make this code publicly available on GitHub[4]. An example of the final output of the diabatization process is provided in Listing E.2.

The process has, broadly, five phases:

1. Section 3.5.1 Geometry optimization
2. Section 3.5.2 Calculating geometry displacements
3. Section 3.5.3 Calculating adiabatic states
4. Section 3.5.4 Evaluating the electronic integral
5. Section 3.5.5 Collating numerical results

### 3.5.1 Step 1: Geometry Optimization & Frequency Calculation

The process begins by choosing a specific molecule, in this case: water ($H_2O$). A representation is chosen for the molecule, specifically a Z-matrix is constructed in Gaussian[5].

An `opt+freq` calculation is performed in Gaussian, using the input file Listing 3.1. This optimizes the three-dimensional geometry of the molecule, as well as calculating the vibrational frequencies through normal mode analysis. Multiple computational methods can be used: DFT, HF, MP, and various basis sets can also be selected: `B3LYP`, `cc-pVTZ`, etc.

This step produces a `.log` file which contains four pieces of necessary information:

1. Listing 3.2 The charge and multiplicity

[4]Hopefully by September.

[5]Other computational chemistry software packages could be used.

```
%nprocshared=8
%Mem=30GB
#p opt freq=hpmodes b3lyp/6-311++g(d,p) geom=connectivity

Water opt+freq 6-311++g(d,p)

0 1
 O                    0.00000000    1.44654086    0.00000000
 H                    0.96000000    1.44654086    0.00000000
 H                   -0.32045459    2.35147669    0.00000000

 1 2 1.0 3 1.0
 2
 3
```

Listing 3.1: Example Gaussian input file `water_optfreq.com` for $H_2O$

```
 Symbolic Z-matrix:
 Charge =  0 Multiplicity = 1
```

Listing 3.2: Charge and multiplicity in `water_optfreq.log`

2. Listing 3.3 The optimized geometry of the molecules in cartesian co-ordinates
3. Listing 3.4 The vibrational frequencies of the 3N-6 modes
4. Listing 3.5 The atomic masses of each atom (in amu)

### Excitation calculation

Using the new optimized geometry (Listing 3.3) a time-dependent Density Functional Theory (TD-DFT) calculation is executed. The user chooses a number of electronic states to calculate: in this case three states. The input file is presented in Listing 3.6.

This calculation produces a `water_excited.log` output file. The FC vertical electronic excitations are taken from there and are used in the final step: Section 3.5.5

```
                         Input orientation:
 ---------------------------------------------------------------------
 Center     Atomic      Atomic             Coordinates (Angstroms)
 Number     Number       Type             X           Y           Z
 ---------------------------------------------------------------------
      1          8           0       -0.011972    1.429601    0.000000
      2          1           0        0.949258    1.466833   -0.000000
      3          1           0       -0.297741    2.348125    0.000000
 ---------------------------------------------------------------------
```

Listing 3.3: Optimized geometry section in `water_optfreq.log`

```
                           1         2         3
                          A1        A1        B2
      Frequencies ---  1602.4365 3818.8124 3923.9167
   Reduced masses ---     1.0829    1.0449    1.0824
  Force constants ---     1.6384    8.9782    9.8194
   IR Intensities ---    66.7263    9.2211   56.9048
```

Listing 3.4: Harmonic frequencies ($cm^{-1}$), Reduced Masses (AMU), Force constants (mDyne/A), and IR intensities (KM/Mole) in `water_optfreq.log`

```
  15.9949146
   1.0078250
   1.0078250
```

Listing 3.5: Atomic masses of each atom in water molecule inside the `masses` file

```
%chk=water_excited.chk
%nprocshared=10
%Mem=35GB
#p td=(nstates=3) b3lyp/6-311++g(d,p) scrf=check

Molecule 3 excited states 6-311++g(d,p)

0 1
8            0        -0.011972     1.429601     0.000000
1            0         0.949258     1.466833    -0.000000
1            0        -0.297741     2.348125     0.000000
```

Listing 3.6: Example Gaussian input file `water_excited.com` for $H_2O$

### 3.5.2 Step 2: Calculate geometry displacements

Recall from Equation (3.24) that for each normal mode $i$, the adiabatic states are evaluated at both the ground state geometry $q_0$, and a displaced geometry $\Delta q_i$. These displacements are calculated using Gaussian, by performing TD-DFT calculations at `cam-b3lyp/6-31G(d)` with the following options: `nosymm iop(3/33=4) td(nstates=8,conver=6) iop(9/40=5)`

The reference geometry is given in Table 3.1 and the displacements $\pm\delta q_i$ are given in Tables 3.2 to 3.4

Table 3.1: Reference geometry $q_0$ (dimensionless)

| Atom | X | Y | Z |
|---|---|---|---|
| O | 0.000 000 0 | 1.446 541 0 | 0.000 000 0 |
| H | 0.960 000 0 | 1.446 541 0 | 0.000 000 0 |
| H | −0.320 455 0 | 2.351 477 0 | 0.000 000 0 |

Table 3.2: Displaced geometry $q_0 \pm \delta q_1$ (dimensionless)

| Atom | $+\delta q_1$ | | | $-\delta q_1$ | | |
|---|---|---|---|---|---|---|
| | X | Y | Z | X | Y | Z |
| O | 0.0000000 | 1.4465410 | 0.0009871 | 0.0000000 | 1.4465410 | -0.0009871 |
| H | 0.9600000 | 1.4405950 | -0.0078328 | 0.9600000 | 1.4524870 | 0.0078328 |
| H | -0.3204550 | 2.3574230 | -0.0078328 | -0.3204550 | 2.3455310 | 0.0078328 |

Table 3.3: Displaced geometry $q_0 \pm \delta q_2$ (dimensionless)

| Atom | $+\delta q_2$ | | | $-\delta q_2$ | | |
|---|---|---|---|---|---|---|
| | X | Y | Z | X | Y | Z |
| O | 0.0000000 | 1.4465410 | -0.0004574 | 0.0000000 | 1.4465410 | 0.0004574 |
| H | 0.9600000 | 1.4411569 | 0.0036298 | 0.9600000 | 1.4519251 | -0.0036298 |
| H | -0.3204550 | 2.3568611 | 0.0036298 | -0.3204550 | 2.3460929 | -0.0036298 |

Table 3.4: Displaced geometry $q_0 \pm \delta q_3$ (dimensionless)

| Atom | $+\delta q_3$ | | | $-\delta q_3$ | | |
|---|---|---|---|---|---|---|
| | X | Y | Z | X | Y | Z |
| O | 0.0000000 | 1.4471697 | 0.0000000 | 0.0000000 | 1.4459123 | 0.0000000 |
| H | 0.9600000 | 1.4415518 | 0.0038239 | 0.9600000 | 1.4515302 | -0.0038239 |
| H | -0.3204550 | 2.3464878 | -0.0038239 | -0.3204550 | 2.3564662 | 0.0038239 |

### 3.5.3 Step 3: Evaluate electronic integral

Finally, Equation (3.24) can be obtained. In this step, the electronic integral $\int d\mathbf{r}$ is computed to obtain the overlap matrix $S_{\lambda\mu}(\boldsymbol{R}_0 \pm \Delta\boldsymbol{q}_i)$.

As we have six geometric displacements $\pm\Delta\boldsymbol{q}_i$, there are six corresponding blocks of output in `water_overdia.out`; starting at lines: `1956`, `2277`, `2677`, `2998`, `3398`, and `3719`. Each block of code has a header of the form in Listing 3.7, and contains the results of evaluating the adiabatic states $\psi_\lambda(\boldsymbol{r};\boldsymbol{R}_0)$, $\psi_\mu(\boldsymbol{r};\boldsymbol{R}_0+\Delta\boldsymbol{q})$

```
2122  *GEOMETRIC PERTURBATION No.           1
```

Listing 3.7: Example header, in `water_overdia.out`

For the first geometric displacement output block, we see the calculation of the overlap matrix element $S_{11}(+\Delta q_1)$ in Listing 3.8. Within lines `2122-2175`, the calculation is repeated once for each of the nine matrix element $S_{jj'}(+\Delta q_1)$. There are similar blocks, computing the nine $S_{jj'}(\pm\Delta q_i)$ matrix elements, for each of the six geometric displacements

```
2122  ista1 ista2           1           1
2123  aaa11    0.50012842920702916
2124  aaa12    5.2244234178751675E-008
2125  aaa21    1.1920706250027622E-007
2126  aaa22    2.2594236048360214E-004
2127  overlap=   0.49990241988371720
```

Listing 3.8: Overlap matrix element $S_{11}(\Delta q_i)$ in `water_overdia.out`

### 3.5.4 Step 4: Calculate transformation matrix

After finding the overlap matrix $S_{\lambda\mu}(\boldsymbol{R}_0 \pm \Delta\boldsymbol{q}_i)$, the transformation matrix $\underline{\boldsymbol{U}}$, which minimizes the difference between $S_{\mu a}$ and $\delta_{\mu a}$, is calculated. In this particular diabatization procedure, this is done using symmetric orthogonalization[6]. For the first geometric displacement, this process occurs on lines `2185-2261`; the $\underline{\boldsymbol{U}}$ and $S_{aj}$ are presented below in Listings 3.9 and 3.10. We can see that $S_{aj}$ is indeed equal to $\delta_{aj}$ up to $1 \times 10^{-5}$.

```
2237      1      0.99988      0.00572      0.01410
2238      2     -0.00572      0.99998     -0.00006
2239      3     -0.01410     -0.00002      0.99990
```

Listing 3.9: The transformation matrix $\underline{\boldsymbol{U}}$ for the first mode's positive displacement, in `water_overdia.out`

```
2248      1      1.00000     -0.00000      0.00000
2249      2     -0.00000      1.00000      0.00000
2250      3      0.00000      0.00000      1.00000
```

Listing 3.10: $S_{\mu a}$ matrix elements for the first mode's positive displacement, in `water_overdia.out`

Having obtained our $\underline{\boldsymbol{U}}$ we calculate our new diabatic states using Equation (3.18)

[6]Also know as (Löwdin) orthogonalization.

```
DIABATIZATION FROM MAX. OVERLAP
ad energy for positive displ   6.8996639976393750
ad energy for negative displ   6.8694104053466551
ad energy for positive displ   8.4022209856185430
ad energy for negative displ   8.3733959898313213
ad energy for positive displ   8.8246610541498871
ad energy for negative displ   8.7776126092186768
```

Listing 3.11: Adiabatic energies $E^a(\boldsymbol{R}_0 \pm \Delta\boldsymbol{q}_1)$ (eV) , in `water_overdia.out`

```
derivative of adiabatic energies
           1   0.15126796146359922
           2   0.14412497893610876
           3   0.23524222465605149
```

Listing 3.12: Symmetric derivative of adiabatic energies (eV) for the first state, in `water_overdia.out`

### 3.5.5 Step 5: Collate numerical results

Step 5 evaluates the diabatic coefficients from Equations (3.21) and (3.25), using all the parameters calculated in the prior steps, and produces one final file, which fully prescribes the vibronic model. An example of such a file is available in Listing E.2.

As previously described, see Equation (3.26), the coefficients of the coupling terms are obtained through numerical derivatives, once for each state we are calculating, at lines 2586, 3307, and 4028 in `water_overdia.out`. Looking at the first state, we can see the adiabatic energies $E^a(\boldsymbol{R}_0 \pm \Delta\boldsymbol{q}_1)$ in Listing 3.11, and the resulting symmetric derivatives in Listing 3.12. These derivatives are used to obtain the diabatic coefficients, which are presented at the end of the file `water_overdia.out`.

In Listing 3.13 we see the electronically diagonal linear coefficients $g_j^{11}$ (defined in eqs. (3.25) and (3.26)) for each mode $j$. In Listing 3.14 we see the off-diagonal linear coefficients $g_j^{12}$ for each mode $j$:

## 3.6 Wrapping up

The specific example here, is oriented towards producing vibronic models to be investigated through spectroscopy. At the moment, another method is also being explored, specifically

```
  COUPLING STATE <   1|H|   1>
 MODE     COUPLING
    1           0.15126954
    2           0.13700266
    3          -0.16662290
```

Listing 3.13: linear coefficients $g_j^{11}$ for each mode $j$, in `water_overdia.out`

```
  COUPLING STATE <  1|H|  2>
 MODE    COUPLING
    1          0.08494233
    2          0.07689810
    3         -0.07576917
```

Listing 3.14: linear coefficients $g_j^{12}$ for each mode $j$, in `water_overdia.out`

one including spin effects, to generate additional vibronic models. New models of the form defined in Section 3.4, can be used in a plug-and-play fashion with all the methods presented in this thesis.

Now that we have seen how to compute a vibronic model in the diabatic basis, I can start to get into the meat of computing properties of interest. In Chapter 4 I will explain the PIMC method, and show results calculated using the Hamiltonians defined in Sections 3.4.2 and 3.4.3. Chapter 5 contains a more general prescription for computing expectation values, which is then applied in Chapter 6, where I show resulting spectra for Hamiltonians defined in Equation (3.37).

# Chapter 4

# Path Integral Monte Carlo

The purpose of this chapter is to present a new approach for the calculation of the thermodynamic properties of vibronic models introduced in Chapter 2. The new approach is based on the Feynman path integral representation of quantum statistical mechanics. The method presented here is in principle exact, as properties will be obtained as statistical estimates over the *exact* thermal distribution using quantum Monte Carlo (MC) with Metropolis-Hastings (MH) sampling. The sources of error include statistical uncertainties, and systematic Trotter factorization error. We will show that statistical errors can be readily minimized as the standard error of estimate scales as $O(\frac{1}{\sqrt{M}})$ where $M$ is the number of independent samples. The Trotter error can be reduced by increasing the imaginary time discretization.

It will be shown that an important feature of vibronic models is that they lead to so-called *non-stoquastic*[1] Hamiltonians [46, pg. 2]. This means that a *sign-problem* will be expected in a quantum Monte Carlo application [47]. This chapter includes an approach to circumvent the sign-problem, based on an partial trace of the discrete electronic state degrees of freedom. One is then left with a reduced density matrix path integral approach for the continuous mode degrees of freedom. To my knowledge, this is the first formal identification of vibronic models as non-stoquastic Hamiltonians.

It is prohibitively expensive[2] to directly calculate many of the integrals present in this work. One solution to this problem is evaluating integrals discretely using methods such as MC and MH. In this work, my use of MH has formally exact sampling. In previous work, I have done approximate sampling and attempted to re-weight the sampling [44]. We will refer to the current method as PIMC and whereas the previous method is a re-weighted version of PIMC.

As the math can be quite dense in this chapter, I will begin by defining some of the most common mathematical objects in Section 4.1. Using this foundation, I present the detailed

[1]This is not a typo; stoquastic is a term used to define a class of Hamiltonians where all off-diagonal matrix elements are real and non-positive. Correspondingly a **non**-stoquastic Hamiltonian has off-diagonal elements with both signs. Many fermionic systems are non-stoquastic. The reference gives a more detailed explanation.

[2]This is, of course, an opinion; but one that seems to be pervasive in the research that the author has so far encountered.

form of the PI in Section 4.2, that I adapt to the MH algorithm. I present a primer on the statistical distributions, and their notation in Appendix A.1. I then demonstrate the different ways of implementing MH, and show that using the gaussian mixture distribution (GMD) approach is preferred. Finally, I will compare and contrast the different MH schemes in Section 4.7.

## 4.1 Basic Definitions

I begin by defining the primary components of my PIs. I work with a vibronic system split into diagonal and off-diagonal (in electronic DoF) components:

$$\hat{H} = \hat{h} + \hat{V}, \tag{4.1}$$

where the Hamiltonian $\hat{H}$ was defined in Equation (3.27) and the $\hat{h}$ operator was defined in Equation (3.29). I express Z in a path integral formulation derived in the same fashion as Section 2.1:

$$Z = \lim_{P\to\infty} \int \mathrm{d}\boldsymbol{q}_1 \int \mathrm{d}\boldsymbol{q}_2 \cdots \int \mathrm{d}\boldsymbol{q}_P \, g(\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_P) \quad = \lim_{P\to\infty} \int \mathrm{d}\boldsymbol{Q}\, g\left(\boldsymbol{Q}\right). \tag{4.2}$$

where $g\left(\boldsymbol{Q}\right)$ is

$$g\left(\boldsymbol{Q}\right) = \sum_{\mathbf{a}}^{A} \prod_{i=1}^{P} \left\langle \boldsymbol{q}_i, a_i \right| e^{-\tau\hat{h}} \left| \boldsymbol{q}_{i+1}, a_i \right\rangle \left\langle \boldsymbol{q}_{i+1}, a_i \right| e^{-\tau\hat{V}} \left| \boldsymbol{q}_{i+1}, a_{i+1} \right\rangle, \tag{4.3}$$

I will employ importance sampling as described in Section 2.2.2 with the target distribution $g\left(\boldsymbol{Q}\right)$ from Equation (4.3), and the new sampling distribution $\varrho\left(\boldsymbol{Q}\right)$

$$\varrho\left(\boldsymbol{Q}\right) = \sum_{\mathbf{a}}^{A} \prod_{i=1}^{P} \left\langle \boldsymbol{q}_i, a_i \right| e^{-\tau\hat{h}} \left| \boldsymbol{q}_{i+1}, a_i \right\rangle. \tag{4.4}$$

I define the function $W(\boldsymbol{Q}, a)$, which is a reduced density matrix,

$$W(\boldsymbol{Q}, \boldsymbol{a}) = \prod_{i=1}^{P} \left\langle \boldsymbol{q}_i, a_i \right| e^{-\tau\hat{h}} \left| \boldsymbol{q}_{i+1}, a_i \right\rangle \left\langle \boldsymbol{q}_{i+1}, a_i \right| e^{-\tau\hat{V}} \left| \boldsymbol{q}_{i+1}, a_{i+1} \right\rangle, \tag{4.5}$$

so that I can express $g\left(\boldsymbol{Q}\right)$

$$g\left(\boldsymbol{Q}\right) = \sum_{\mathbf{a}}^{A} W(\boldsymbol{Q}, \boldsymbol{a}). \tag{4.6}$$

In this way I can express $Z$

$$Z = \lim_{P\to\infty} \int \mathrm{d}\boldsymbol{Q} \sum_{\mathbf{a}}^{A} W(\boldsymbol{Q}, \boldsymbol{a}), \tag{4.7}$$

or some other property $A$ of interest

$$\langle A\rangle = \frac{1}{Z}\lim_{P\to\infty}\int \mathrm{d}\boldsymbol{Q}\sum_{\mathbf{a}}^{A} W(\boldsymbol{Q},\boldsymbol{a})A_{\mathrm{estim}}(\boldsymbol{Q}). \tag{4.8}$$

### 4.1.1 Matrix representation

As described in Section 2.4, the dominant cost of evaluating the $\boldsymbol{Q}$ integrals is in computing and diagonalizing the matrices $\mathbb{M}$, $\mathbb{O}$, $\tilde{\mathbb{O}}$. I will now define these matrices through their matrix elements. The $\mathbb{O}$ matrix elements:

$$\mathbb{O}\left(\boldsymbol{q},\boldsymbol{q}^{'}\right)_{aa'} = \left\langle\boldsymbol{q}\middle|e^{-\tau\hat{h}^{a}}\middle|\boldsymbol{q}^{'}\right\rangle\delta_{aa'}, \tag{4.9}$$

and $\mathbb{M}$ matrix elements:

$$\mathbb{M}\left(\boldsymbol{q}\right)_{aa'} = \langle a|e^{-\tau\hat{V}(\boldsymbol{q})}|a'\rangle\,. \tag{4.10}$$

Equations (4.3) and (4.5) can now be expressed as the product of $\mathbb{O}$ and $\mathbb{M}$:

$$W(\boldsymbol{Q},a) = \prod_{i=1}^{P}\mathbb{O}(\boldsymbol{q}_i,\boldsymbol{q}_{i+1})\mathbb{M}(\boldsymbol{q}_{i+1}), \tag{4.11}$$

$$g\left(\boldsymbol{Q}\right) = \sum_{a} W(\boldsymbol{Q},a) = \mathrm{Tr}\left[\prod_{i=1}^{P}\mathbb{O}(\boldsymbol{q}_i,\boldsymbol{q}_{i+1})\mathbb{M}(\boldsymbol{q}_{i+1})\right], \tag{4.12}$$

and Equation (4.4) in terms of $\mathbb{O}$

$$\varrho\left(\boldsymbol{Q}\right) = \mathrm{Tr}\left[\prod_{i=1}^{P}\mathbb{O}(\boldsymbol{q}_i,\boldsymbol{q}_{i+1})\right]. \tag{4.13}$$

The $\tilde{\mathbb{O}}$ matrix is defined in almost the same way as $\mathbb{O}$,

$$\tilde{\mathbb{O}}\left(\boldsymbol{q},\boldsymbol{q}^{'}\right)_{\tilde{a}\tilde{a}'} = \left\langle\boldsymbol{q}\middle|e^{-\tau\hat{h}^{\tilde{a}}}\middle|\boldsymbol{q}^{'}\right\rangle\delta_{aa'}, \tag{4.14}$$

however the size of the matrix *can* be larger or smaller than $\mathbb{O}$. $\tilde{\mathbb{O}}$ is of size $\tilde{A}\times\tilde{A}$ instead of $A\times A$ where $A$ is the number of electronic states of the vibronic model as defined in Section 3.4.1. $\tilde{\mathbb{O}}$ is still evaluated on the same $\boldsymbol{q}$ continuous co-ordinates but the discrete contribution $e^{-\tau\hat{h}^{\tilde{a}}}$ is chosen to optimize the statistical efficiency of evaluating the integrals. Generally the simplest choice is $\tilde{\mathbb{O}} = \mathbb{O}$ but better statistical performance can be achieved by adding fictitious surfaces or removing surfaces of low contribution.

Similarly there is

$$\tilde{\varrho}\left(\boldsymbol{Q}\right) = \mathrm{Tr}\left[\prod_{i=1}^{P}\tilde{\mathbb{O}}(\boldsymbol{q}_i,\boldsymbol{q}_{i+1})\right]. \tag{4.15}$$

## 4.2 Theory

Now that I have covered basic definitions, I can begin to layout the PI formalism in full. Four different MH schemes were explored, two of which I discuss in detail within the following Sections 4.3 to 4.5.

I will begin by describing the naive **direct** scheme, which doesn't use any fancy tricks and evaluates individual matrix elements. I will show how this approach is fundamentally flawed and a different approach is needed. Next is the **fixed** method which performed somewhat poorly. Finally we have the **selective matrix**, and the GMD or **averaged matrix** schemes.

As the reader will see in Section 4.7, the GMD scheme is statistically superior to the other schemes, in both accuracy and computational efficiency. The goal of all these schemes, is to express some property of interest in terms of a PI integral, which will be approximated through MH.

For each scheme, it is important to define what a state is, and how a trajectory is generated. Proposal states are drawn from a proposal distribution and then either accepted or rejected. Each scheme has several subtle differences in this process. Recall that the basic MH algorithm is laid out in Section 2.2.

## 4.3 Metropolis scheme 1 (Direct approach)

We describe the **Direct** approach for evaluating the path integral function through metropolis sampling. The goal of the metropolis sampling is to generate a *trajectory* (a set of states $\boldsymbol{x}_0, \boldsymbol{x}_1, \dots \boldsymbol{x}_T$) to estimate a distribution $P(\boldsymbol{x})$.

### 4.3.1 Definition of a state

We define a state of the system $\boldsymbol{x}$ for some number of beads $P$ and electronic surfaces $A$:

$$\boldsymbol{x} \sim [\boldsymbol{Q}, \boldsymbol{a}] \qquad \boldsymbol{x} = [x_1, x_2, \dots, x_P] \qquad x_i \sim [\boldsymbol{q}_i, a_i] \tag{4.16}$$

$$\boldsymbol{a} = [a_1, a_2, \dots, a_P] \qquad a_i \in \{1, 2, \dots, A\} \tag{4.17}$$

$$\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \dots, \boldsymbol{q}_P] \qquad \boldsymbol{q}_i = [q_{1,i}, q_{2,i}, \dots, q_{N,i}] \qquad \forall ij \quad q_{j,i} \in \mathbb{R} \tag{4.18}$$

Evaluating Equation (4.3) with these parameters looks like the following:

$$\begin{aligned} g\left(\boldsymbol{Q}\right) = \sum_{a_1=0}^{2}\sum_{a_2=0}^{2}\sum_{a_3=0}^{2} & \langle \boldsymbol{q}_1, a_1| \, e^{-\tau\hat{h}} \, |\boldsymbol{q}_2, a_1\rangle \, \langle \boldsymbol{q}_2, a_1| \, e^{-\tau\hat{V}} \, |\boldsymbol{q}_2, a_2\rangle \\ & \times \langle \boldsymbol{q}_2, a_2| \, e^{-\tau\hat{h}} \, |\boldsymbol{q}_3, a_2\rangle \, \langle \boldsymbol{q}_3, a_2| \, e^{-\tau\hat{V}} \, |\boldsymbol{q}_3, a_3\rangle \\ & \times \langle \boldsymbol{q}_3, a_3| \, e^{-\tau\hat{h}} \, |\boldsymbol{q}_1, a_3\rangle \, \langle \boldsymbol{q}_1, a_3| \, e^{-\tau\hat{V}} \, |\boldsymbol{q}_1, a_1\rangle \, . \end{aligned} \tag{4.19}$$

### 4.3.2 Generating a trajectory

A trajectory is a set of states $\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots \boldsymbol{x}_T$ where each state $\boldsymbol{x}_{t+1}$ is generated based on the previous state $\boldsymbol{x}_t$. The *target* or *desired* distribution is defined as $P(\boldsymbol{x})$, and the distribution from which samples are drawn is $g(\boldsymbol{x}'|\boldsymbol{x})$

At each time step, the new state $\boldsymbol{x}_{t+1}$ is either the previous state $\boldsymbol{x}_t$, or a candidate state $\boldsymbol{x}'$ which is drawn from $g(\boldsymbol{x}'|\boldsymbol{x})$. To determine which value $\boldsymbol{x}_{t+1}$ will take on, the *acceptance ratio* is used:

$$A(\boldsymbol{x}', \boldsymbol{x}) = \min\left(1, \frac{P(\boldsymbol{x}')g(\boldsymbol{x}|\boldsymbol{x}')}{P(\boldsymbol{x})g(\boldsymbol{x}'|\boldsymbol{x})}\right). \tag{4.20}$$

To obtain a new state $\boldsymbol{x}_{t+1}$ given the current state of $\boldsymbol{x}_t$:

1. Generate a random candidate state $\boldsymbol{x}'$ from $g(\boldsymbol{x}'|\boldsymbol{x}_t)$
2. Calculate the acceptance probability using Equation (4.20)
3. Generate a random number $u \in [0, 1]$
   (a) if $u \leq A(\boldsymbol{x}', \boldsymbol{x}_t)$ then *accept* the candidate state and $\boldsymbol{x}_{t+1} = \boldsymbol{x}'$
   (b) if $u > A(\boldsymbol{x}', \boldsymbol{x}_t)$ then *reject* the candidate state and $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t$

For our method, the target distribution is $g\left(\boldsymbol{Q}\right)$ from Equation (4.3) and the sampling distribution is $\varrho\left(\boldsymbol{Q}\right)$ from Equation (4.4). Note that samples drawn from $\varrho\left(\boldsymbol{Q}\right)$ are independent: $\varrho(\boldsymbol{x}'|\boldsymbol{x}_t) = \varrho(\boldsymbol{x}')$ and $\varrho(\boldsymbol{x}_t|\boldsymbol{x}') = \varrho(\boldsymbol{x}_t)$.
The *acceptance ratio* for the direct method is:

$$A(\boldsymbol{x}', \boldsymbol{x}_t) = \min\left(1, \frac{g(\boldsymbol{x}')}{g(\boldsymbol{x}_t)}\frac{\varrho(\boldsymbol{x}_t)}{\varrho(\boldsymbol{x}')}\right), \tag{4.21}$$

with probability ratio (between the proposed sample $\boldsymbol{x}'$ and the previous sample $\boldsymbol{x}_t$)

$$\frac{g(\boldsymbol{x}')}{g(\boldsymbol{x}_t)}, \tag{4.22}$$

and proposal density ratio

$$\frac{\varrho(\boldsymbol{x}_t)}{\varrho(\boldsymbol{x}')}. \tag{4.23}$$

### 4.3.3 Example of evaluating proposed sample state

For a system with two surfaces ($A = 2$, $a_i \in \{0, 1\}$) and three beads ($P = 3$), if our previous state was $\boldsymbol{x}_t \sim \Big[[\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3], [1, 1, 0]\Big]$ and we want to compute the next state $\boldsymbol{x}_{t+1}$, then:

1. Draw a candidate state $\boldsymbol{x}'$ from $\varrho\left(\boldsymbol{Q}\right)$

(a) draw $P$ random surfaces $a_1, a_2, a_3 \in \text{unif}\{0,1\}$[3](for example $a_1, a_2, a_3 = 0, 1, 0$)

(b) draw $P \times N$ random positions $\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3$ from a distribution (GMD or some $\pi_a$)

We now have a candidate state $\boldsymbol{x}' \sim \Big[[\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3], [0, 1, 0]\Big]$

2. Evaluate Equation (4.22) using $\boldsymbol{x}'$ and $\boldsymbol{x}_t$:

   Both $g(\boldsymbol{x}')$ and $g(\boldsymbol{x}_t)$ can be evaluated as shown in Equation (4.19) to give

$$\frac{g(\boldsymbol{x}')}{g(\boldsymbol{x}_t)} = \frac{\langle \boldsymbol{q}_1, 0| e^{-\tau \hat{h}} |\boldsymbol{q}_2, 0\rangle \langle \boldsymbol{q}_2, 0| e^{-\tau \hat{V}} |\boldsymbol{q}_2, 1\rangle \langle \boldsymbol{q}_2, 1| e^{-\tau \hat{h}} |\boldsymbol{q}_3, 1\rangle}{\langle \boldsymbol{q}_1, 1| e^{-\tau \hat{h}} |\boldsymbol{q}_2, 1\rangle \langle \boldsymbol{q}_2, 1| e^{-\tau \hat{V}} |\boldsymbol{q}_2, 1\rangle \langle \boldsymbol{q}_2, 1| e^{-\tau \hat{h}} |\boldsymbol{q}_3, 1\rangle} \times \frac{\langle \boldsymbol{q}_3, 1| e^{-\tau \hat{V}} |\boldsymbol{q}_3, 0\rangle \langle \boldsymbol{q}_3, 0| e^{-\tau \hat{h}} |\boldsymbol{q}_1, 0\rangle \langle \boldsymbol{q}_1, 0| e^{-\tau \hat{V}} |\boldsymbol{q}_1, 0\rangle}{\langle \boldsymbol{q}_3, 1| e^{-\tau \hat{V}} |\boldsymbol{q}_3, 0\rangle \langle \boldsymbol{q}_3, 0| e^{-\tau \hat{h}} |\boldsymbol{q}_1, 0\rangle \langle \boldsymbol{q}_1, 0| e^{-\tau \hat{V}} |\boldsymbol{q}_1, 1\rangle}$$

3. Evaluate Equation (4.23) using $\boldsymbol{x}'$ and $\boldsymbol{x}_t$

$$\frac{\varrho(\boldsymbol{x}_t)}{\varrho(\boldsymbol{x}')} = \frac{\mathbb{O}(\boldsymbol{q}_1, \boldsymbol{q}_2)\mathbb{O}(\boldsymbol{q}_2, \boldsymbol{q}_3)\mathbb{O}(\boldsymbol{q}_3, \boldsymbol{q}_1)}{\mathbb{O}(\boldsymbol{q}_1, \boldsymbol{q}_2)\mathbb{O}(\boldsymbol{q}_2, \boldsymbol{q}_3)\mathbb{O}(\boldsymbol{q}_3, \boldsymbol{q}_1)} \tag{4.24}$$

$$= \frac{\langle \boldsymbol{q}_1, 1| e^{-\tau \hat{h}} |\boldsymbol{q}_2, 1\rangle \langle \boldsymbol{q}_2, 1| e^{-\tau \hat{h}} |\boldsymbol{q}_3, 1\rangle \langle \boldsymbol{q}_3, 0| e^{-\tau \hat{h}} |\boldsymbol{q}_1, 0\rangle}{\langle \boldsymbol{q}_1, 0| e^{-\tau \hat{h}} |\boldsymbol{q}_2, 0\rangle \langle \boldsymbol{q}_2, 1| e^{-\tau \hat{h}} |\boldsymbol{q}_3, 1\rangle \langle \boldsymbol{q}_3, 0| e^{-\tau \hat{h}} |\boldsymbol{q}_1, 0\rangle} \tag{4.25}$$

4. Compute the acceptance ratio using Equation (4.21)

5. Compare the acceptance ratio to $u$ and set $\boldsymbol{x}_{t+1}$ to $\boldsymbol{x}'$ or $\boldsymbol{x}_t$ as appropriate

Z would be computed in this fashion

$$\begin{aligned} Z = \int \mathrm{d}\boldsymbol{q}_1 \int \mathrm{d}\boldsymbol{q}_2 \int \mathrm{d}\boldsymbol{q}_3 & \langle \boldsymbol{q}_1, 0| e^{-\tau \hat{h}} |\boldsymbol{q}_2, 0\rangle \langle \boldsymbol{q}_2, 0| e^{-\tau \hat{V}} |\boldsymbol{q}_2, 1\rangle \\ & \times \langle \boldsymbol{q}_2, 1| e^{-\tau \hat{h}} |\boldsymbol{q}_3, 1\rangle \langle \boldsymbol{q}_3, 0| e^{-\tau \hat{V}} |\boldsymbol{q}_3, 0\rangle \\ & \times \langle \boldsymbol{q}_3, 0| e^{-\tau \hat{h}} |\boldsymbol{q}_1, 0\rangle \langle \boldsymbol{q}_1, 0| e^{-\tau \hat{V}} |\boldsymbol{q}_1, 0\rangle . \end{aligned} \tag{4.26}$$

Unfortunately the Direct method performs abysmally and in my testing almost always is frozen in the initial state and never accepts any proposed states. Only by moving to a matrix approach did I finally see good results.

## 4.4 Metropolis scheme 2 (Averaged matrix/GMD approach)

We improve upon the **direct** scheme by changing how the states are evaluated, specifically the discrete DoF are treated differently. Instead of *stochastically* evaluating the electronic

[3]In general the distribution is unif$\{0, A-1\}$ or unif$\{1, A\}$. I treat these as equivalent given the choice of the value of the starting surface. It feels natural to refer to the ground state as 0.

surfaces, as in the **Direct** scheme, the Averaged matrix exactly evaluates them using matrix multiplication.

Another difference between this scheme and the **Direct** scheme is how the individual beads are evaluated. Previously each bead was evaluated on a specific surface $a_i$, drawn uniformly. Now surfaces $\tilde{a}_i$ are drawn, not uniformly, but with weights $w^i$, and each bead is evaluated on **all** surfaces as the AxA matrices $\mathbb{O}$ and $\mathbb{M}$.

The **Averaged matrix** scheme is as follows For some number of samples $L$:

1. Draw three $(P)$ random values $\tilde{a}_1, \tilde{a}_2, \tilde{a}_3$ with weights $[w^1, w^2, w^3]$
2. Draw three $(P)$, $(N \times L)$ samples $\underline{\boldsymbol{y}}^1 \in \varrho^{\tilde{a}_1}$, $\underline{\boldsymbol{y}}^2 \in \varrho^{\tilde{a}_2}$, $\underline{\boldsymbol{y}}^3 \in \varrho^{\tilde{a}_3}$[4]
3. Transform from collective bead co-ordinates $y^\ell_{j\lambda}$ to bead dependent co-ordinates $x^\ell_{ji}$

$$\underline{\boldsymbol{x}}^\ell = \underline{\boldsymbol{y}}^\ell (\underline{\boldsymbol{\mathcal{V}}})^\dagger, \tag{4.27}$$

4. To evaluate the matrices $\mathbb{O}$ and $\mathbb{M}$ we shift each sample to all $A$ electronic states:

$$\boldsymbol{x}_i^{\ell,a} = \boldsymbol{x}_i^\ell + \boldsymbol{d}^{\tilde{a}_\ell} - \boldsymbol{d}^a, \tag{4.28}$$

resulting in a tensor $\underline{\boldsymbol{x}}$ of dimension $L \times A \times N \times P$.

5. Evaluate $\varrho(\underline{\boldsymbol{x}}_t)$ and $\varrho(\underline{\boldsymbol{x}}')$ in a matrix fashion:

$$\mathrm{Tr}_A \left( \begin{bmatrix} \boldsymbol{q}_1\boldsymbol{q}_2 & 0 & 0 \\ 0 & \boldsymbol{q}_1\boldsymbol{q}_2 & 0 \\ 0 & 0 & \boldsymbol{q}_1\boldsymbol{q}_2 \end{bmatrix} \times \begin{bmatrix} \boldsymbol{q}_2\boldsymbol{q}_3 & 0 & 0 \\ 0 & \boldsymbol{q}_2\boldsymbol{q}_3 & 0 \\ 0 & 0 & \boldsymbol{q}_2\boldsymbol{q}_3 \end{bmatrix} \times \begin{bmatrix} \boldsymbol{q}_3\boldsymbol{q}_1 & 0 & 0 \\ 0 & \boldsymbol{q}_3\boldsymbol{q}_1 & 0 \\ 0 & 0 & \boldsymbol{q}_3\boldsymbol{q}_1 \end{bmatrix} \right) \tag{4.29}$$

6. Evaluate $g(\underline{\boldsymbol{x}}_t)$ and $g(\underline{\boldsymbol{x}}')$ in a matrix fashion:

$$\begin{aligned} \mathrm{Tr}_A \Bigg( & \begin{bmatrix} \boldsymbol{q}_1\boldsymbol{q}_2 & 0 & 0 \\ 0 & \boldsymbol{q}_1\boldsymbol{q}_2 & 0 \\ 0 & 0 & \boldsymbol{q}_1\boldsymbol{q}_2 \end{bmatrix} \times \begin{bmatrix} \mathbb{M}(\boldsymbol{q}_2)_{00} & \mathbb{M}(\boldsymbol{q}_2)_{01} & \mathbb{M}(\boldsymbol{q}_2)_{02} \\ \mathbb{M}(\boldsymbol{q}_2)_{10} & \mathbb{M}(\boldsymbol{q}_2)_{11} & \mathbb{M}(\boldsymbol{q}_2)_{12} \\ \mathbb{M}(\boldsymbol{q}_2)_{20} & \mathbb{M}(\boldsymbol{q}_2)_{21} & \mathbb{M}(\boldsymbol{q}_2)_{22} \end{bmatrix} \\ & \times \begin{bmatrix} \boldsymbol{q}_2\boldsymbol{q}_3 & 0 & 0 \\ 0 & \boldsymbol{q}_2\boldsymbol{q}_3 & 0 \\ 0 & 0 & \boldsymbol{q}_2\boldsymbol{q}_3 \end{bmatrix} \times \begin{bmatrix} \mathbb{M}(\boldsymbol{q}_3)_{00} & \mathbb{M}(\boldsymbol{q}_3)_{01} & \mathbb{M}(\boldsymbol{q}_3)_{02} \\ \mathbb{M}(\boldsymbol{q}_3)_{10} & \mathbb{M}(\boldsymbol{q}_3)_{11} & \mathbb{M}(\boldsymbol{q}_3)_{12} \\ \mathbb{M}(\boldsymbol{q}_3)_{20} & \mathbb{M}(\boldsymbol{q}_3)_{21} & \mathbb{M}(\boldsymbol{q}_3)_{22} \end{bmatrix} \\ & \times \begin{bmatrix} \boldsymbol{q}_3\boldsymbol{q}_1 & 0 & 0 \\ 0 & \boldsymbol{q}_3\boldsymbol{q}_1 & 0 \\ 0 & 0 & \boldsymbol{q}_3\boldsymbol{q}_1 \end{bmatrix} \times \begin{bmatrix} \mathbb{M}(\boldsymbol{q}_1)_{00} & \mathbb{M}(\boldsymbol{q}_1)_{01} & \mathbb{M}(\boldsymbol{q}_1)_{02} \\ \mathbb{M}(\boldsymbol{q}_1)_{10} & \mathbb{M}(\boldsymbol{q}_1)_{11} & \mathbb{M}(\boldsymbol{q}_1)_{12} \\ \mathbb{M}(\boldsymbol{q}_1)_{20} & \mathbb{M}(\boldsymbol{q}_1)_{21} & \mathbb{M}(\boldsymbol{q}_1)_{22} \end{bmatrix} \Bigg) \end{aligned} \tag{4.30}$$

7. Compute the acceptance ratio using Equation (4.21)

[4]Here the $\underline{\boldsymbol{y}}^j$ samples are 2-dimensional tensors of dimensions L (samples/batches) and N (modes). After drawing P of them we now have L x 1 x N x P samples. When we perform the shift in Equation (4.28) we "broadcast" the singular 1 dimension to A samples producing L x A x N x P samples.

8. Compare the acceptance ratio to $u$ and set $\boldsymbol{x}_{t+1}$ to $\boldsymbol{x}'$ or $\boldsymbol{x}_t$ as appropriate

Then we can repeat this process $N$ times.

## 4.5 Additional Metropolis schemes

Two additional methods were considered: the **Fixed** method, a simplification of the **Direct** method, and the **Selective matrix** method, an extension of the **Averaged matrix** method.

### 4.5.1 Fixed method

Here we "fix" all beads to the same randomly sampled surface. This is a very simplistic method and should behave poorly anytime the system is not trivial. It may be interesting to compare the fixed method against the more complex methods, for systems that are in the ground state (due to large energy separation or thermalization). This comparison may be able to indicate performance/accuracy costs incurred by the additional machinery.

The **Fixed** method for drawing a candidate state $\boldsymbol{x}'$ from $\varrho\left(\boldsymbol{Q}\right)$ is as follows:

1. Draw 1 random surface $a \in \text{unif}(0, A)$ (for example $a = 3$)
2. Draw $P \times N$ random positions $\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots$ from a distribution (either GMD or some $\pi_a$)
3. We now have a candidate state $\boldsymbol{x}' \sim \Big[[\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots], 3\Big]$

### 4.5.2 Selective matrix method

Recall from the **Averaged matrix** approach:

For each bead $i \in \{0, P-1\}$:

1. Draw $\tilde{a}_i$ with weight $w^i$ (1)
2. Chose a distribution $\varrho^{\tilde{a}_i}$ (2)
3. Draw a collective bead co-ordinate $y^{\ell}_{j\lambda}$ that had no notion of surfaces. (3)
4. Transform to bead dependent co-ordinates $x^{\ell}_{ji}$
5. Shift each sample to all $A$ electronic states $\boldsymbol{x}^{\ell,a}_{i}$

The **Selective matrix** is an attempt to use the sampling of the **Averaged matrix** approach in the **Direct** method fashion, by being more selective than simply "broadcasting" each bead-dependent sample to all surfaces and then evaluating the integral as matrices. Instead, the integral is calculated by multiplying respective matrix elements ($\mathbb{O}_{aa}$, $\mathbb{M}_{aa'}$) for each $a_i$ sample.

For a proposed state $\boldsymbol{x}' \sim [\boldsymbol{Q}, \boldsymbol{a}]$:

- Normal mode components ($\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_P]$) are still drawn as the **Averaged matrix** approach
  drawing collective co-ordinates and transforming etc.
- Electronic component ($\boldsymbol{a} = [a_1, a_2, \ldots, a_P]$) are drawn in the **Direct** method fashion: $a_i \in \text{unif}\{0, A-1\}$
- Finally the probability ratio $\frac{g(\boldsymbol{x}')}{g(\boldsymbol{x}_t)}$ and proposal density ratio $\frac{\varrho(\boldsymbol{x}')}{\varrho(\boldsymbol{x}_t)}$ are calculated as follows:
  (suppose 4 beads with $\boldsymbol{a} = [2, 1, 1, 3]$ and $\boldsymbol{a}' = [0, 3, 2, 0]$)[5]

$$\frac{g(\boldsymbol{x}')}{g(\boldsymbol{x}_t)} = \frac{\mathbb{O}(\boldsymbol{q}_1, \boldsymbol{q}_2)_{00}\mathbb{M}\,(\boldsymbol{q}_2)_{03}\,\mathbb{O}(\boldsymbol{q}_2, \boldsymbol{q}_3)_{33}\mathbb{M}\,(\boldsymbol{q}_3)_{32}\,\mathbb{O}(\boldsymbol{q}_3, \boldsymbol{q}_4)_{22}\mathbb{M}\,(\boldsymbol{q}_4)_{20}\,\mathbb{O}(\boldsymbol{q}_4, \boldsymbol{q}_1)_{00}\mathbb{M}\,(\boldsymbol{q}_1)_{00}}{\mathbb{O}(\boldsymbol{q}_1, \boldsymbol{q}_2)_{22}\mathbb{M}\,(\boldsymbol{q}_2)_{21}\,\mathbb{O}(\boldsymbol{q}_2, \boldsymbol{q}_3)_{11}\mathbb{M}\,(\boldsymbol{q}_3)_{11}\,\mathbb{O}(\boldsymbol{q}_3, \boldsymbol{q}_4)_{11}\mathbb{M}\,(\boldsymbol{q}_4)_{13}\,\mathbb{O}(\boldsymbol{q}_4, \boldsymbol{q}_1)_{33}\mathbb{M}\,(\boldsymbol{q}_1)_{32}} \tag{4.31}$$

$$\frac{\varrho(\boldsymbol{x}')}{\varrho(\boldsymbol{x}_t)} = \frac{\mathbb{O}(\boldsymbol{q}_1, \boldsymbol{q}_2)_{00}\mathbb{O}(\boldsymbol{q}_2, \boldsymbol{q}_3)_{33}\mathbb{O}(\boldsymbol{q}_3, \boldsymbol{q}_4)_{22}\mathbb{O}(\boldsymbol{q}_4, \boldsymbol{q}_1)_{00}}{\mathbb{O}(\boldsymbol{q}_1, \boldsymbol{q}_2)_{22}\mathbb{O}(\boldsymbol{q}_2, \boldsymbol{q}_3)_{11}\mathbb{O}(\boldsymbol{q}_3, \boldsymbol{q}_4)_{11}\mathbb{O}(\boldsymbol{q}_4, \boldsymbol{q}_1)_{33}} \tag{4.32}$$

It is important to highlight there is **NO** trace over the surfaces as we are multiplying individual matrix elements! This means that we are still stochastically evaluating the discrete electronic states and suffer from the sign problem.

## 4.6 Solution to manifestation of the sign problem

The uniform sampling revealed an important fact about the vibronic Hamiltonian studied here: they are non-stoquastic. A stoquastic Hamiltonian $\hat{H}$ has the general form defined in Equation (3.27) but whose matrix elements are negative [46]

$$\langle a|\, H(q)\, |a'\rangle < 0. \tag{4.33}$$

As we move from un-coupled, to weakly coupled, to strongly coupled systems, Uniform sampling breaks down because the vibronic coupling manifests as a sign problem in the distributions defined by the $V$ component of the Hamiltonian. The sign problem causes the

[5] And the $\boldsymbol{q}$'s are different but I can't be bothered to distinguish them right now.

variance to become extremely large so as to make the sampling of the problem intractable. Whereas the GMD-reduced method tackles this manifestation of the sign problem by tracing out electronic DoF **exactly**. We are left to stochastically sampling the modes, which do not have a sign problem because they are bosonic. In this way, our variance is well behaved and this makes the problem tractable. One can readily see from Equation (3.39) that the representation of a vibronic Hamiltonian, in the position representation for the modes $q_1, q_2$, and the discrete representation for the states $A$ will have alternating signs.

Now we've gone through the two methods, explaining how the MH algorithm works. In the next section I will show results of applying the GMD and GMD-reduced matrix schemes to the displaced system defined in Equation (3.38).

## 4.7 Results

I focused on testing the GMD and GMD-reduced methods using the Displaced system which is described below. I do not plot any results for the Direct method as this method simply doesn't work. It rejects almost every proposed move and therefore fails to generate a Markov Chain. I also did not plot any Fixed method results as both GMD methods are as good or better than the Fixed method.

### Efficient Diagonalization

In this section, SOS[6] results are used as to benchmark the PIMC methods. The exact implementation of these results used an operator-based diagonalization approach, that computes the action and never stores the matrix elements of the Hamiltonian. SOS results were obtained through the use of the `LinearOperator` object and the `eigsh` method provided by the `scipy.sparse.linalg` library. Since this is a sparse approach it avoids the storage issues discussed in Chapter 1. In addition, this is more efficient than a standard sparse matrix representation of the Hamiltonian. Because it never stores any matrix elements and instead uses an iterative solver approach. The Hamiltonian can be represented as a sum of `LinearOperator`s and then simply diagonalized as shown in Listing 4.1.

```
h_Ea = LinearOperator((N,N), matvec=Ea_v)
h_01 = LinearOperator((N,N), matvec=h01_v)
h_02 = LinearOperator((N,N), matvec=h02_v)
h_q1 = LinearOperator((N,N), matvec=q1_v)
h_q2 = LinearOperator((N,N), matvec=q2_v)

# Construct DVR Hamiltonian out of LinearOperators
H_total = h_Ea + h_01 + h_02 + h_q1 + h_q2

# Sparse diagonalization
evals, evecs = eigsh(H_total, k=100, which = 'SA')
```

Listing 4.1: Illustrative use of `LinearOperator`s and `eigsh` for diagonalization of a Hamiltonian.

[6]Also known as exact diagonalization (ED).

All references to SOS results were generated using the sparse LinearOperator approach just described.

### 4.7.1 Displaced System

The Displaced system's Hamiltonian is of the form defined in Equation (3.38) with the parameters given in Table 4.1. This system is useful because it has a "tunable" parameter $\gamma$ which allows us to investigate the performance over a range of coupling strengths. $\gamma_1$ there is no off-diagonal coupling and the two electronic surfaces are completely independent. Based on past experience $\gamma_2, \gamma_3$ denote weakly coupled systems, $\gamma_4$ is a turning point (neither weak, nor strong), and finally $\gamma_5, \gamma_6$ denote strongly coupled systems. The reader will see that the results of the PIMC methods agree with this assessment.

Table 4.1: Displaced system parameters (eV).

| Parameter | | Parameter | |
|---|---|---|---|
| $E^a$ | 0.0996 | $\gamma_1$ | 0.00 |
| $E^b$ | 0.1996 | $\gamma_2$ | 0.04 |
| $\omega_1$ | 0.02 | $\gamma_3$ | 0.08 |
| $\omega_2$ | 0.04 | $\gamma_4$ | 0.12 |
| $\lambda$ | 0.072 | $\gamma_5$ | 0.16 |
| | | $\gamma_6$ | 0.20 |

#### Characteristic Temperature

To illustrate, the tunability of the Displaced system, I have plotted heat capacity ($C_v$) versus temperature for each $\gamma_i$ value in Figure 4.1. These plots were obtained using SOS. In each plot, the corresponding characteristic temperature $\Theta$ is denoted with a dashed vertical orange line. The idea of high or low temperature is associated with the population of excited states. The so-called characteristic temperature is defined as

$$\Theta = \frac{\Delta E}{k_B} \tag{4.34}$$

For an uncoupled system, there will be an electronic temperature based on the spacing between the electronic states, and a temperature for each of the independent normal modes. When coupling is introduced, the mixing between the degrees of freedom blurs the distinction between the individual temperatures of each degree of freedom. Here, we use the first energy gap for the fully-coupled Hamiltonian to define the system's characteristic temperature:

$$\Theta = \frac{E_1 - E_0}{k_B}, \tag{4.35}$$

where $E_0$ is the ground state and $E_1$ is the first excited state. Each $\gamma_i$ has a different energy gap and therefore a different characteristic temperature. These values are provided in Table 4.2. Note that whenever I refer to $\Theta$, it is relative to some choice of $\gamma_i$.

In Figure 4.1, the heat capacity's relation to temperature is very consistent for $\gamma_1$, $\gamma_2$, and $\gamma_3$, with only a slight variation in the $T > 600$K range for $\gamma_3$. We see a huge change once the second mode's coefficient ($\gamma_4 = 0.12$) is much larger than the first mode's ($\lambda = 0.072$). Both the temperature ($x$) axis and the slope change drastically. This system should prove somewhat challenging. Finally, $\gamma_5$ and $\gamma_6$ both have slopes which decrease very quickly, but with temperature ranges different by several orders of magnitude: $T \leq 0.01$ for $\gamma_5$, and $T \leq 2 \times 10^{-7}$ for $\gamma_6$. The low temperatures for these systems pose an issue for completing calculations in a reasonable amount of time.

Table 4.2: Displaced system characteristic temperatures (K).

| | $\Theta$ |
|---|---|
| $\gamma_1$ | 232.09 |
| $\gamma_2$ | 231.21 |
| $\gamma_3$ | 226.39 |
| $\gamma_4$ | 33.33 |
| $\gamma_5$ | $8.94 \times 10^{-4}$ |
| $\gamma_6$ | $4.64 \times 10^{-8}$ |

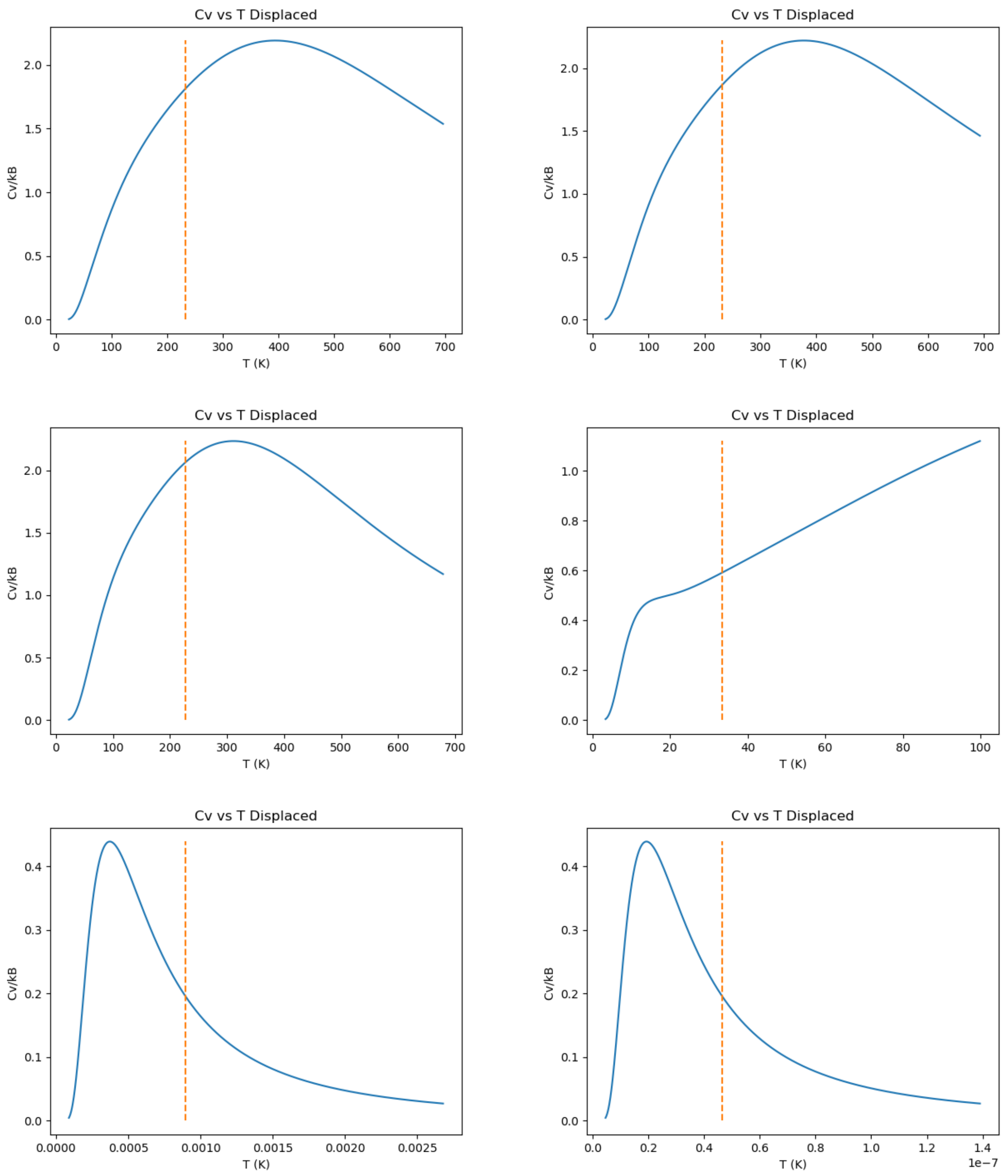

Figure 4.1: Heat Capacity vs Temperature (Kelvin) of Displaced system for each $\gamma_i$ value. The characteristic temperatures are denoted with a dashed vertical orange line. Their values are listed in Table 4.2.

### Calculation Parameters

I benchmark the GMD and GMD-reduced methods against SOS results. Each subsection (4.7.2 to 4.7.11) show computational results for a specific $q_2$ coupling ($\gamma_i$) and proposal distribution. The plots compare the distributions of the two normal modes $q_1$ and $q_2$. The histogram for each method were calculated in the same fashion as in Equation (2.31). All simulations were performed using the parameters in Table 4.3.

Table 4.3: Path Integral simulation parameters.

| | |
|---|---|
| $N_{\text{total}}$ | $1 \times 10^6$ |
| $N_{\text{skip}}$ | $1 \times 10^3$ |
| $N_{\text{burn in}}$ | $1 \times 10^2$ |
| $P$ | 16 |

For the first three coupling coefficients $\gamma_1, \gamma_2, \gamma_3$ (4.7.2 to 4.7.4) results are presented over a range of temperatures relative to their respective $\Theta$: 0.1, 1.0, 2.0, 5.0, and 10.0 times $\Theta$. Only 2.0, 5.0, and 10.0 times $\Theta$ are presented for $\gamma_4$, (4.7.5 and 4.7.6) due to divergence issues for 0.1, and 1.0 $\Theta$. Exact values are given in Table 4.4. It is expected that for 2.0$\Theta$ and above that we should start to see the effect of thermalization: that more of the eigenstates have non zero population.

For $\gamma_5$ and $\gamma_6$, we can no longer use these temperature ranges due to the extreme value of $\Theta$, that causes numerical divergence in the $\sinh(x)$ prefactor terms, as well as when computing the acceptance ratio. For the acceptance ratio, this divergence can be addressed to some degree by scaling the $\mathbb{O}$'s by a constant prefactor[7]. However, for the hyperbolic prefactors we really only have two means to handle divergence: the number of beads $P$ and the temperature $\beta = (k_B T)^{-1}$. For $\gamma_5$, we need to use thousands of beads to avoid numerical issues, and many more beads are needed for $\gamma_6$. To keep the number of beads consistent, I chose to use temperatures of 100K and 300K as shown in Table 4.5.

Table 4.4: Temperature values (in Kelvin) relative to $\Theta$ for each $\gamma_i$.

| | 0.1$\Theta$ | 1.0$\Theta$ | 2.0$\Theta$ | 5.0$\Theta$ | 10.0$\Theta$ |
|---|---|---|---|---|---|
| $\gamma_1$ | $2.32 \times 10^1$ | $2.32 \times 10^2$ | $4.64 \times 10^2$ | $1.16 \times 10^3$ | $2.32 \times 10^3$ |
| $\gamma_2$ | $2.31 \times 10^1$ | $2.31 \times 10^2$ | $4.62 \times 10^2$ | $1.16 \times 10^3$ | $2.31 \times 10^3$ |
| $\gamma_3$ | $2.26 \times 10^1$ | $2.26 \times 10^2$ | $4.53 \times 10^2$ | $1.13 \times 10^3$ | $2.26 \times 10^3$ |
| $\gamma_4$ | N/A | N/A | $6.67 \times 10^1$ | $1.67 \times 10^2$ | $3.33 \times 10^2$ |

[7]Since $\mathbb{O}$ is diagonal in surfaces $A$, we can multiply by a constant term and it will carry through to the exponential energy prefactor. Performing this both in the numerator and denominator results in the scaling factor simply cancelling out, and is equivalent to multiplying by identity.

Table 4.5: Temperature values used in strongly coupled systems $\gamma_5$ and $\gamma_6$.

| | T = 100K | | T = 300K | |
|---|---|---|---|---|
| $\gamma_5$ | $1.1187 \times 10^5$ | $\Theta$ | $3.3561 \times 10^5$ | $\Theta$ |
| $\gamma_6$ | $2.1554 \times 10^9$ | $\Theta$ | $6.4662 \times 10^9$ | $\Theta$ |

### $\tau$ convergence

The choice of temperature $T$ and bead $P$ values is linked to the Trotter error. In my previous work [44, Fig 2.], I performed $\tau$ convergence analysis on the Displaced system; note that the y axis is described in [44, Eq. 70]. Visually analyzing the previous figure, we can see that for the low coupling strength $\gamma_2$, even at very high values of $\tau$ there is only a tiny deviation between SOS and the Trotter results. In the strongly coupled case $\gamma_5$, high values of $\tau$ have substantial deviation. The initial conclusion is that a low number of beads should be sufficient for $\gamma_1$ to $\gamma_4$. If we are to consider the strongly coupled $\gamma_5$, then for "complete agreement" we should use $\tau$ values in the range $[0.2, 0.4](\text{eV}^{-1})$, as there is less than 1% difference between the Trotter and SOS. This can be relaxed a bit to $[0.4, 0.6](\text{eV}^{-1})$, if (or when) between 1% and 3% difference is acceptable. At the far end, one might consider $\tau = 1(\text{eV}^{-1})$ if $\approx 10\%$ difference is acceptable[8].

If we begin with an analysis of the characteristic temperatures, aiming for $\tau = 1$ as the low end of accuracy for $\gamma_5$, then we have a simple prescription: $P = \beta$, as shown in Table 4.6. Numbers are expressed with orders of magnitude to emphasize the fact that as the coupling strength $\gamma$ increases, the value of $\beta$ changes drastically. For most temperatures, 16 beads should be sufficient for the coupling strengths $\gamma_1$ to $\gamma_4$, as this results in $\tau$ values around 10. However, the $\beta$ values for $0.1\Theta$, as well as $1.0\Theta$ for $\gamma_4$, are quite large and may require more beads. This may explain the "flipping" of the $q_1$ and $q_2$ distributions that can be observed in the GMD results when comparing Figures 4.7 and 4.8, as well as Figures 4.12 and 4.13. Additionally, the marked difference between the GMD results in Figures 4.19 and 4.20 could be due to lack of $\tau$ convergence. Another solution to the debilitatingly high values of $\beta$ would be to perform simulations with lower $P$ values (i.e. higher $\tau$) and use the asymptotic behaviour of the Trotter error to extrapolate to $\tau = 0$.

Table 4.6: $\beta$ values $(\text{eV}^{-1})$ relative to $\Theta$ for each $\gamma_i$.

| | $0.1\Theta$ | $1.0\Theta$ | $2.0\Theta$ | $5.0\Theta$ | $10.0\Theta$ |
|---|---|---|---|---|---|
| $\gamma_1$ | $5.00 \times 10^2$ | $5.00 \times 10^1$ | $2.50 \times 10^1$ | $1.00 \times 10^1$ | 5.00 |
| $\gamma_2$ | $5.02 \times 10^2$ | $5.02 \times 10^1$ | $2.51 \times 10^1$ | $1.00 \times 10^1$ | 5.02 |
| $\gamma_3$ | $5.13 \times 10^2$ | $5.13 \times 10^1$ | $2.56 \times 10^1$ | $1.03 \times 10^1$ | 5.13 |
| $\gamma_4$ | $3.48 \times 10^3$ | $3.48 \times 10^2$ | $1.74 \times 10^2$ | $6.96 \times 10^1$ | $3.48 \times 10^1$ |
| $\gamma_5$ | $1.30 \times 10^8$ | $1.30 \times 10^7$ | $6.49 \times 10^6$ | $2.60 \times 10^6$ | $1.30 \times 10^6$ |
| $\gamma_6$ | $2.50 \times 10^{12}$ | $2.50 \times 10^{11}$ | $1.25 \times 10^{11}$ | $5.00 \times 10^{10}$ | $2.50 \times 10^{10}$ |

[8]This is of course based on the difference between partition functions, which is not a natural quantity to compare. Here we are just trying to get a sense of the approximate ranges.

For results presented here we only used 16 beads. I did run some calculations with increased number of beads, but the GMD-reduced distributions were largely the same. Specifically, I generated results for the strongly coupled $\gamma_5$ Displaced system using the bead values listed in Table 4.7. Results for Figures 4.25 and 4.26 were almost identical. There was some noticeable difference when comparing to Figures 4.27 and 4.28, but only for the GMD method, which performed markedly better: the GMD-reduced performed exactly the same. My intuition is that the density of samples becomes more important than the Trotter error for the normal mode distributions, and that we are already sufficiently converged in $\tau$. It may also be the case that another benefit of the GMD-reduced method is this improved performance at lower number of bead simulations.

Table 4.7: Possible $P$ values to obtain $\tau$ values in strongly coupled systems $\gamma_5$, $\gamma_6$.

| $\tau(\mathrm{eV}^{-1})$ | T = 100K | T = 300K |
|---|---|---|
| 1.0 | 116 | 39 |
| 0.6 | 193 | 64 |
| 0.4 | 290 | 97 |
| 0.2 | 580 | 193 |

## Proposal Distributions

Here I list three proposal-distributions [9] used in the calculations.

1. The first proposal distribution, $\pi_1$ is based on the uncoupled components $\hat{h}$ of the Hamiltonian as defined in Equations (3.27) and (3.29):

$$\hat{h}^a = E^{aa} + \frac{1}{2}\sum_j^N \omega_j(\hat{p}_j^2 + \hat{q}_j^2) + \sum_j^N g_j^{aa}\hat{q}_j.$$

   The proposal distribution $\pi_1$ is a GMD comprised of two Gaussians defined by these equations (given the values in Table 4.1):

$$\begin{bmatrix} E^a + \hat{h}_o + \lambda\hat{q}_1 & 0 \\ 0 & E^b + \hat{h}_o - \lambda\hat{q}_1 \end{bmatrix}$$

   This is a simple approach that can be replicated for any linear vibronic model by simply setting all off-diagonal terms to 0 and treating each surface as individual Harmonic Oscillators.

2. The second proposal distribution $\pi_2$ attempts to capture the $q_2$ contribution to the Hamiltonian. As $\gamma$ increases the $q_2$ mode eventually dominates the $q_1$ mode. We expect

[9]Distributions from which proposal states are drawn when constructing the Markov Chain using the MH algorithm.

$\pi_2$ to perform better for systems with stronger coupling as it captures the dominant $q_2$ contributions.

$$\begin{bmatrix} E^a + \hat{h}_o + \gamma_i \hat{q}_2 & 0 \\ 0 & E^b + \hat{h}_o - \gamma_i \hat{q}_2 \end{bmatrix}$$

3. The third proposal distribution $\pi_3$ is an attempt to combine the two prior distributions and hopefully capture both $q_1$ and $q_2$ contributions; $\pi_3$ is only used in Section 4.7.11.

$$\begin{bmatrix} E^a + \hat{h}_o + \lambda \hat{q}_1 + \gamma_i \hat{q}_2 & 0 \\ 0 & E^b + \hat{h}_o - \lambda \hat{q}_1 - \gamma_i \hat{q}_2 \end{bmatrix}$$

### 4.7.2 Uncoupled ($\gamma_1$), Proposal $\pi_1$

I plot histograms of samples generated using both the GMD-reduced and GMD matrix PIMC methods. These histograms approximate the true distributions of the normal modes $q_1$, $q_2$ which have been computed using SOS. For the Displaced system with no coupling we should expect to get exact agreement with SOS. Since the Proposal distribution $\pi_1$ is identical to the true distribution I don't explore results using an alternative proposal distribution.

in Figure 4.2 we see both methods do indeed exactly agree. Figure 4.3 sees almost exact agreement, although the GMD-reduced approach slightly overestimates the $q_1$ distribution in the $[2.5, 5.0]$ range. Increasing the temperature to $2.0\Theta$ allows the $q_1$ mode to spread out and have some appreciable probability of taking on positive values in Figure 4.4. Here we see the first difference between the GMD-reduced and GMD method. The GMD-reduced $q_1$ distribution agrees exactly with the SOS in the positive region. The GMD method has *some* samples in the positive region but appears to struggle to get over the barrier; it does reasonable in the $[0.0, 2.5]$ range, and has a few samples in the $[3, 5]$ range, but overall misses the qualitative shape of the positive component of the distribution.

Moving to $5.0\Theta$ in Figure 4.5, where it appears that we have increased the temperature high enough that the barrier doesn't prove sufficiently difficult to overcome anymore and both methods are able to recover the general shape of the distributions. The GMD-reduced method seems to produce slightly smoother distributions. To get very fine agreement with the SOS we could simply increase the number of samples here.

Both methods perform somewhat poorly in Figure 4.6 when $T = 10.0\Theta$. The GMD-reduced method does okay with the $q_1$ mode, but struggles to get the appropriate width $\sigma$ of both distributions. I think in this case we may need more samples to get nice agreement with SOS.

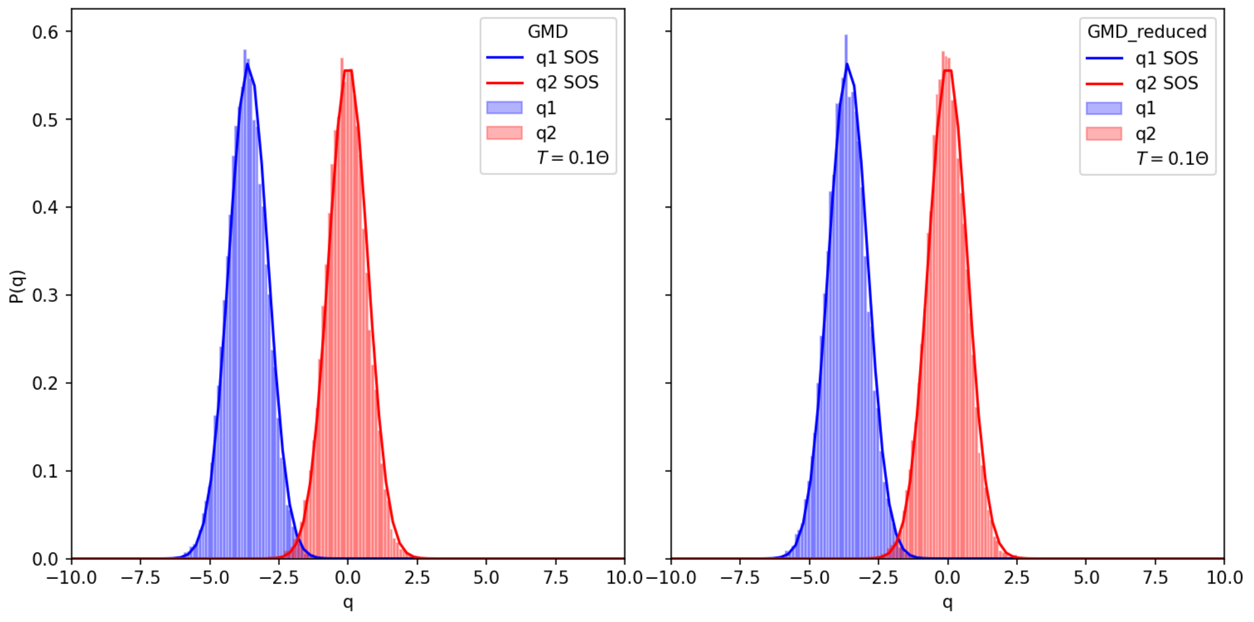

Figure 4.2: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 0.1\Theta$, $\gamma_1$, and $\pi_1$.

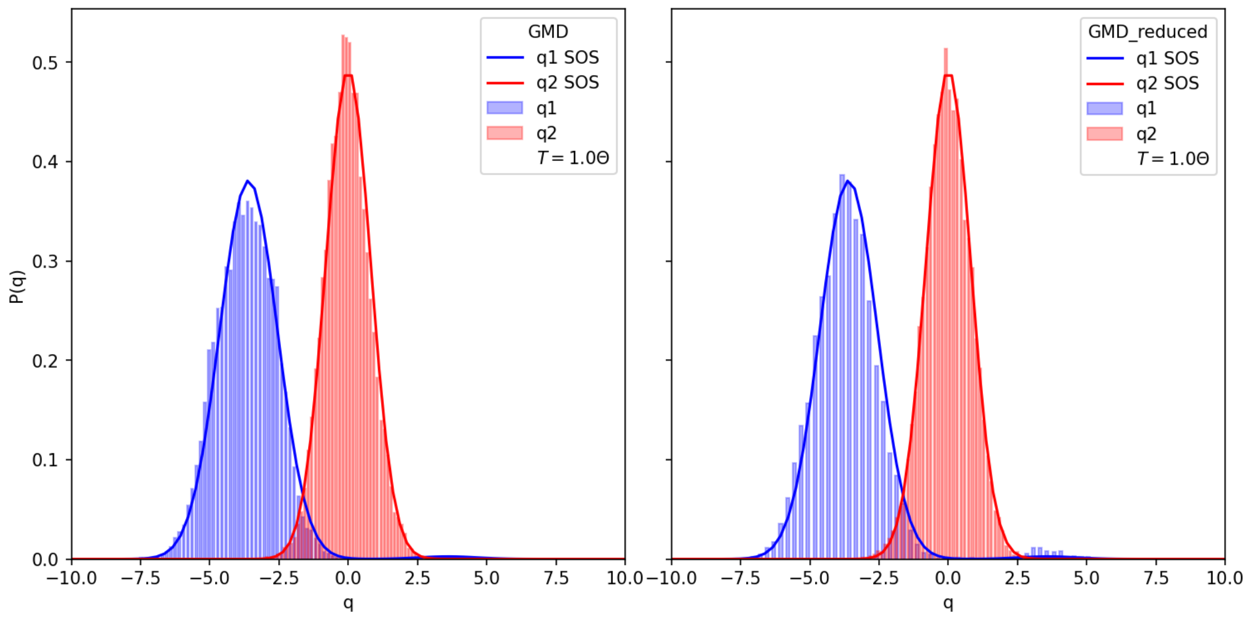

Figure 4.3: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 1.0\Theta$, $\gamma_1$, and $\pi_1$.

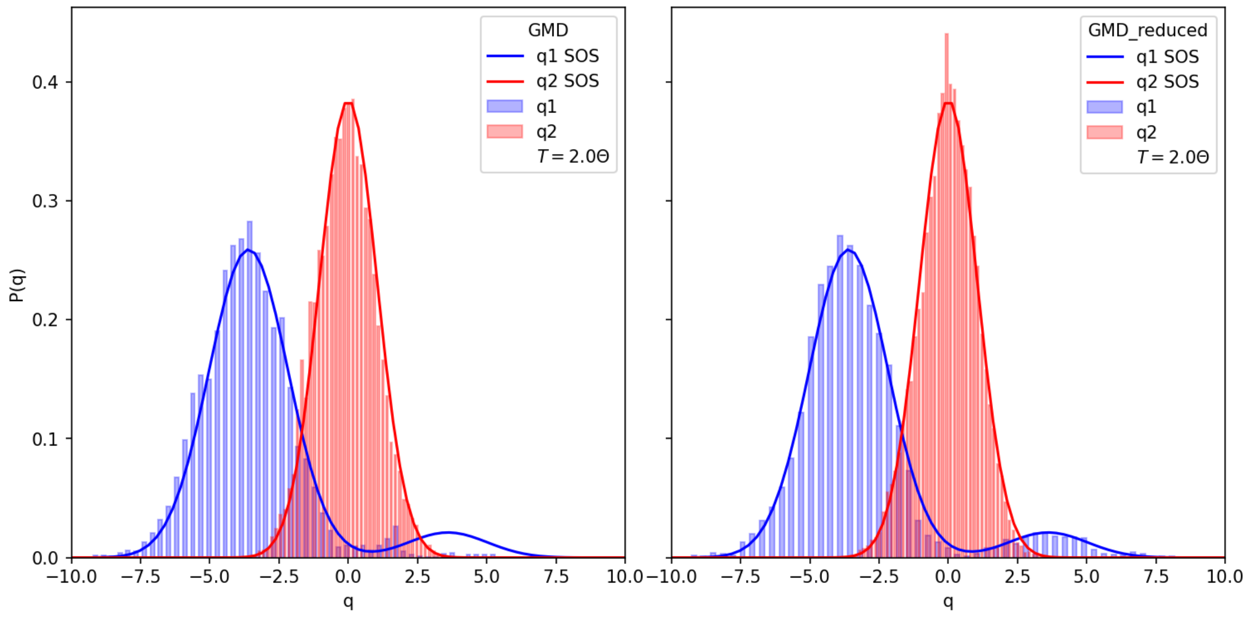

Figure 4.4: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 2.0\Theta$, $\gamma_1$, and $\pi_1$.

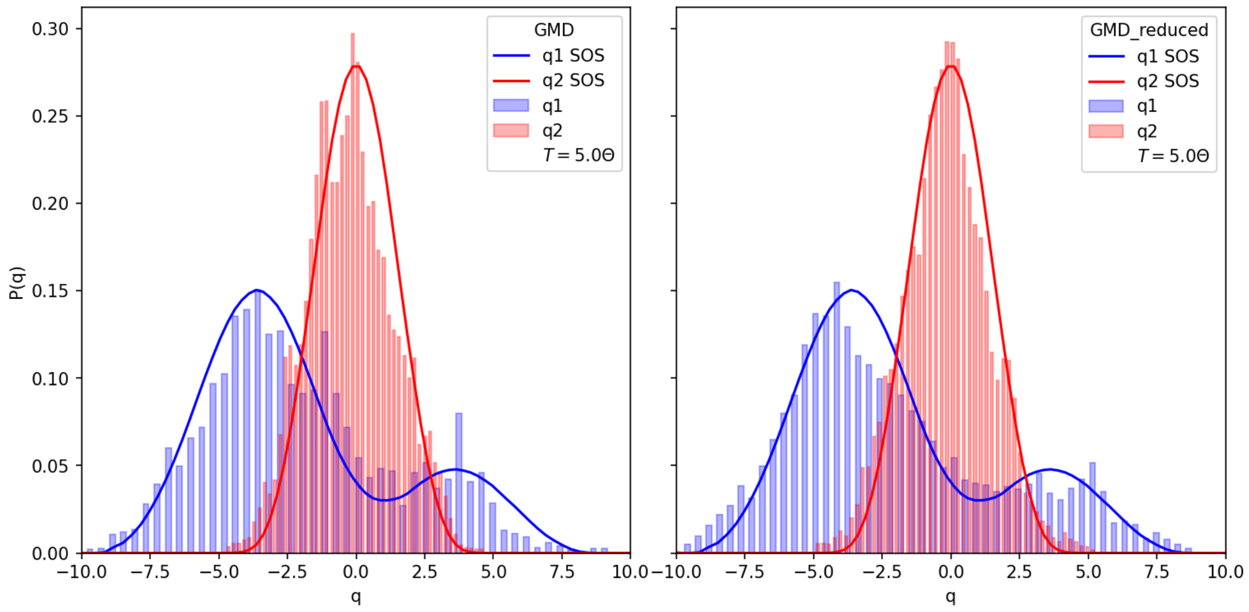

Figure 4.5: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 5.0\Theta$, $\gamma_1$, and $\pi_1$.

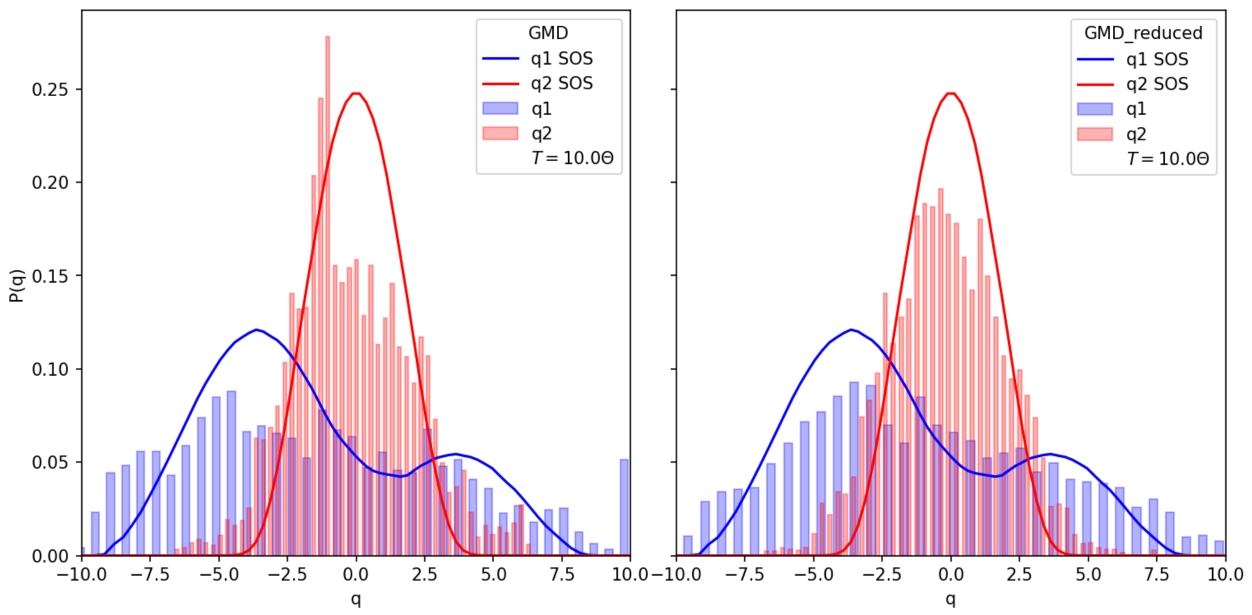

Figure 4.6: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 10.0\Theta$, $\gamma_1$, and $\pi_1$.

### 4.7.3 Weak Coupling ($\gamma_3$), Proposal $\pi_1$

For the Displaced system with weak coupling ($\gamma_3$), we should expect more difficulty getting exact agreement with SOS. In this case we use the basic proposal distribution $\pi_1$ defined by the harmonic components of the Hamiltonian.

in Figure 4.7 we see the GMD-reduced method almost exactly agrees with the SOS. While the GMD method's samples are in the correct $q$ range, the distribution's shapes are wrong.

The GMD method performs surprisingly better for $1.0\Theta$ in Figure 4.8, almost matching the SOS distribution shape exactly. The GMD-reduced approach captures the $q_1$ distribution correctly, but for $q_2$ the samples width is slightly too tight. This trend of samples being over represented in the centre of the distribution seems to also be present for the GMD-reduced approach at $T = 2.0\Theta$ and $T = 5.0\Theta$.

Increasing the temperature to $2.0\Theta$ allows the $q_1$ mode to spread out and have some appreciable probability of taking on positive values in Figure 4.9. Again we see that the GMD-reduced approach does a better job capturing the $q_1$'s distribution shape in the positive regime. Instead of exact agreement with the SOS it struggles in the $[-1.75, 1.75]$ range. The GMD method has *some* samples in the positive region but appears to struggle to get over the barrier; it does reasonably well in the $[-1.75, 1.75]$ range, and has a few samples in the $[2, 5]$ range, but overall misses the qualitative shape of the positive component of the distribution.

Moving to $5.0\Theta$ in Figure 4.10, it again appears that we have increased the temperature high enough that the barrier doesn't prove sufficiently difficult to overcome any more and both methods are able to recover the general shape of the distributions. The GMD-reduced method seems to produce slightly smoother distributions, but again has a slightly smaller width than the true distribution.

Both methods perform somewhat poorly at capturing the $q_1$ in Figure 4.11 when $T = 10.0\Theta$. Surprisingly the GMD-reduced method has better agreement with $q_2$, the width not being so narrow in this case. The GMD method does capture the $q_2$ distribution.

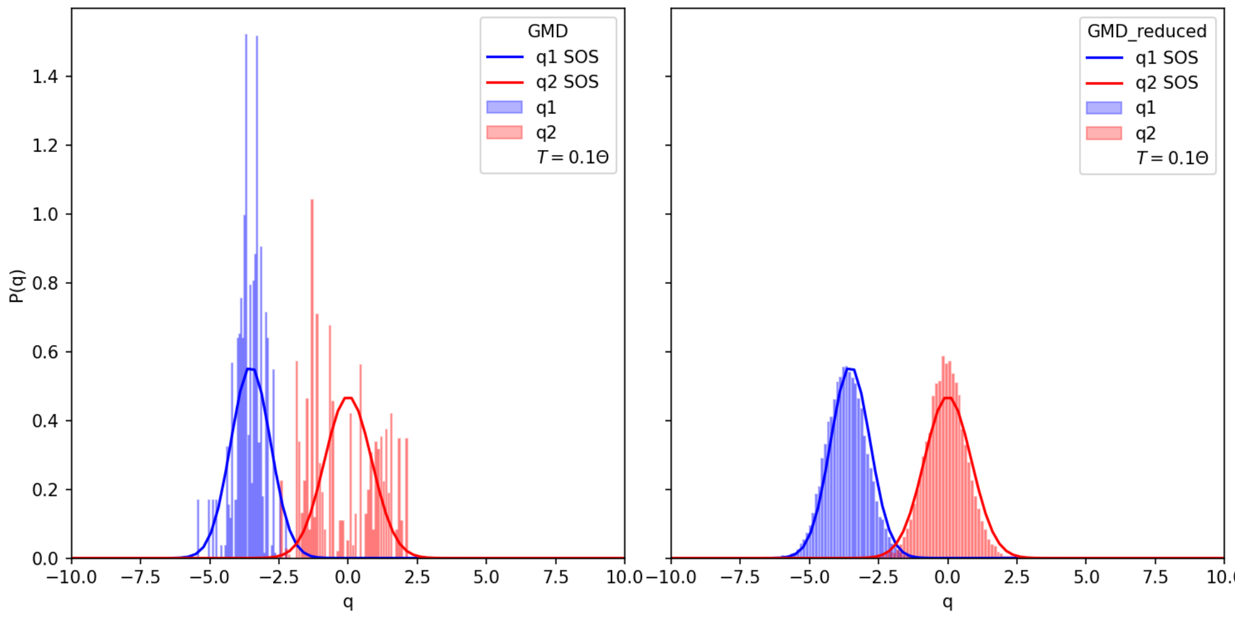

Figure 4.7: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 0.1\Theta$, $\gamma_3$, and $\pi_1$.

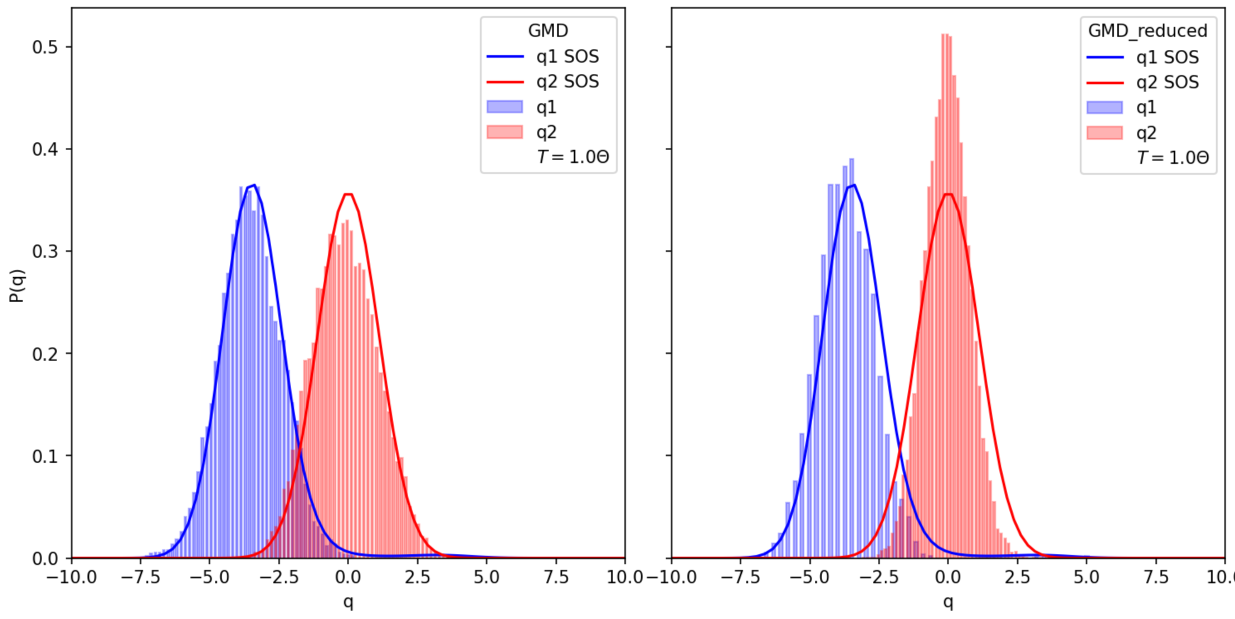

Figure 4.8: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 1.0\Theta$, $\gamma_3$, and $\pi_1$.

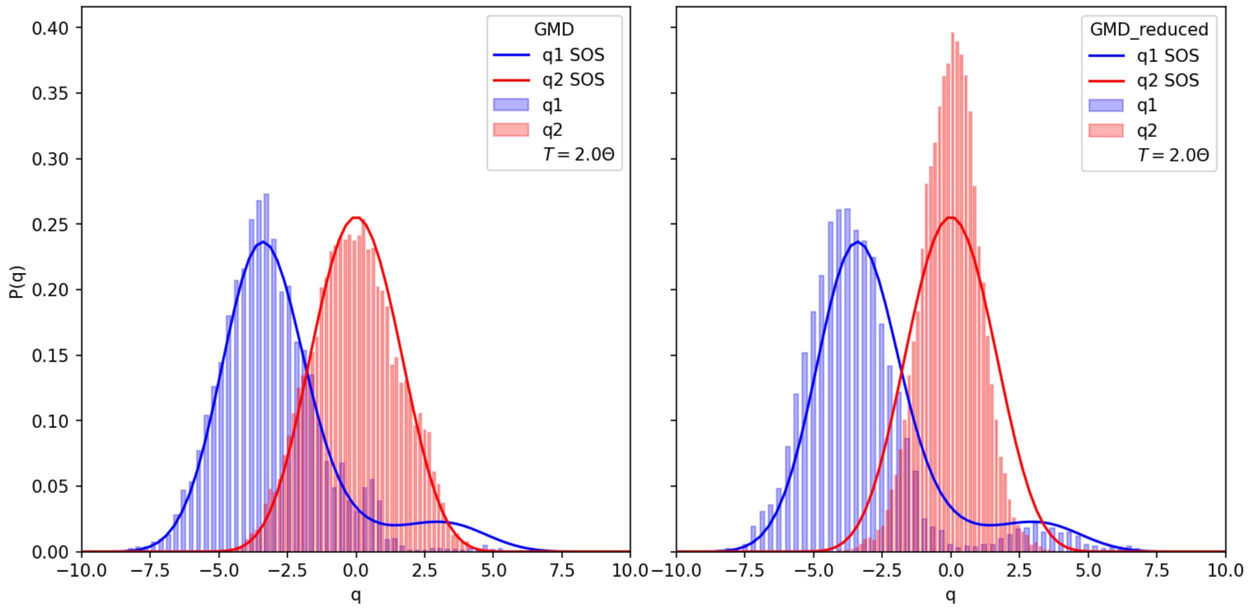

Figure 4.9: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 2.0\Theta$, $\gamma_3$, and $\pi_1$.

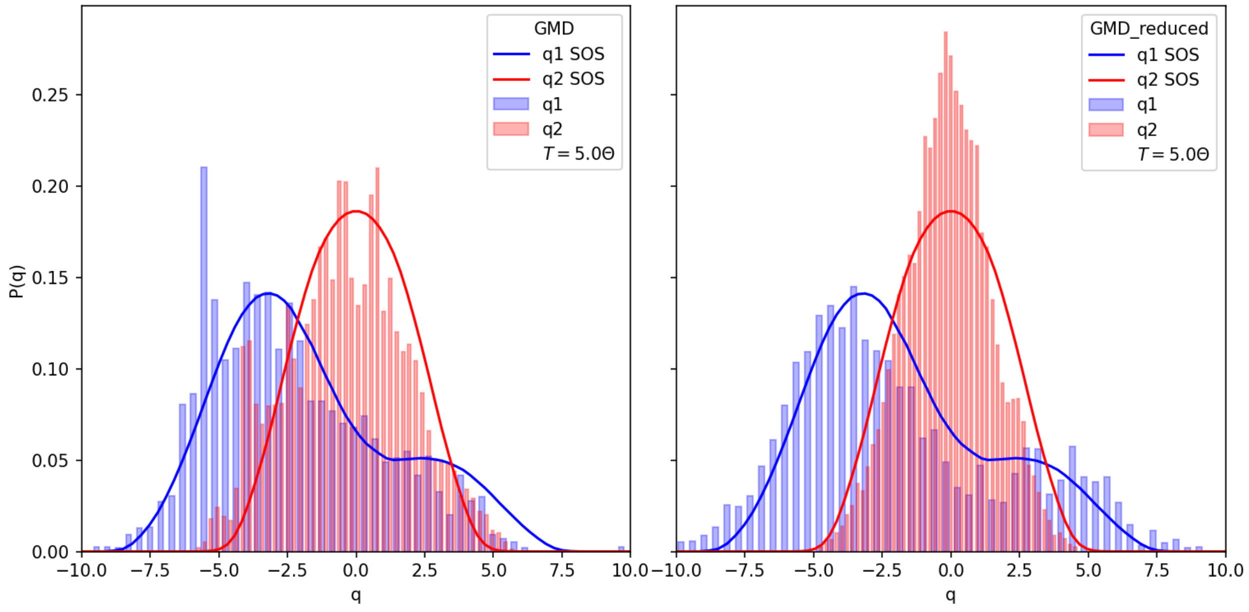

Figure 4.10: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 5.0\Theta$, $\gamma_3$, and $\pi_1$.

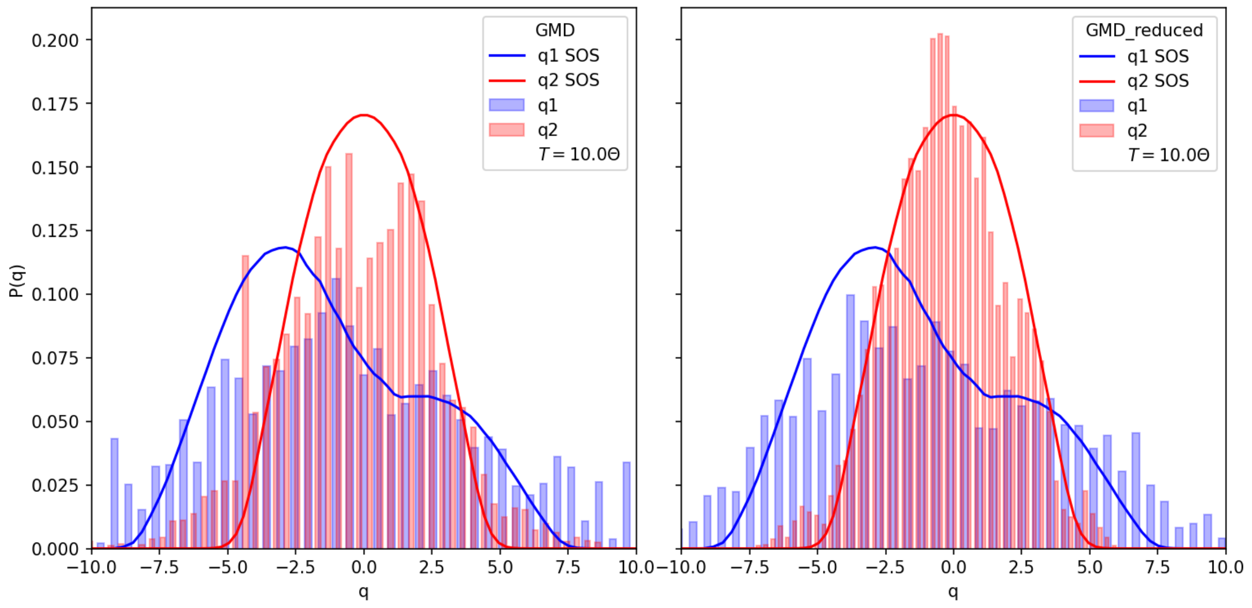

Figure 4.11: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 10.0\Theta$, $\gamma_3$, and $\pi_1$.

### 4.7.4 Weak Coupling ($\gamma_3$), Proposal $\pi_2$

For the Displaced system with weak coupling ($\gamma_3$), we should expect more difficulty getting exact agreement with SOS. In this case we use the second proposal distribution $\pi_2$ which we expect should help with capturing the $q_2$ contributions that are now present, as opposed to $\gamma_1$ which has no $q_2$ contribution.

In all the following plots we see something quite unexpected. The GMD-reduced matrix method's samples are **strongly** affected by the proposal distribution, such that for all Figures 4.12 to 4.16 the GMD-reduced method's $q_1$ and $q_2$ are seemingly "swapped". The GMD-reduced $q_1$ distribution does a good job of approximating the true $q_2$ distribution for almost all of the Figures. The GMD-reduced $q_2$ distribution does not give a good estimation of the true $q_1$ distribution does capture some of the qualitative shape, specifically for $1.0\Theta$, $T = 2.0\Theta$ and $T = 5.0\Theta$. The most striking result is in Figure 4.14, where we can clearly see the bi-modal shape of the $q_2$ distribution similar to bi-modal shape of the true $q_1$ distribution.

The GMD method does not demonstrate the same behaviour due to the change of the proposal distribution. For $T \geq 1.0\Theta$ it seems to do a fairly reasonable job of correctly capturing both the $q_1$ and $q_2$ distributions, especially in Figure 4.15 where $T = 5.0\Theta$, the results look very good. However, we see unexpected behaviour between Figures 4.12 and 4.13. It seems as we shift from $0.1\Theta$ to $1.0\Theta$ the GMD $q_1$ and $q_2$ distributions swap! My hypothesis here is that at the lower temperature the acceptance probability is affected such that the GMD method struggles to produce a good Markov Chain and instead is suffering the same issue as the GMD-reduced method where the proposal distribution is strongly affecting the results.

I think this set of figures, for $\gamma_3$ and $\pi_2$, highlights a key difference between the GMD-reduced and GMD methods. Based on the results in this Chapter it seems the GMD-reduced method, in general, produces smoother histograms and much better agreement with the SOS **when** the proposal distribution is "good". Said another way, if we have an ergodic Markov Chain then the GMD-reduced method seems to perform better. However, the GMD method, while requiring more samples to get as smooth a distribution as the GMD-reduced method, seems to be able to tease out the true distribution even with a poor proposal distribution.

Although we will see in Sections 4.7.9 and 4.7.10 that once the coupling becomes too strong that the GMD method falls apart. So it seems that the GMD method's performance in this region is due to the weak coupling.

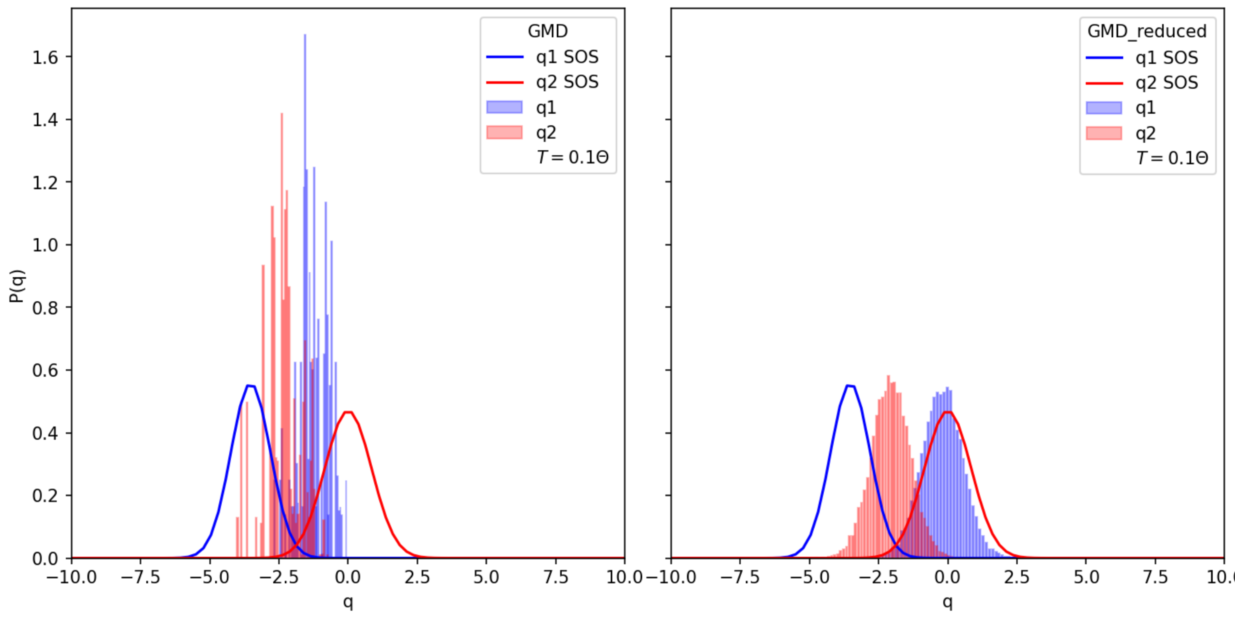

Figure 4.12: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 0.1\Theta$, $\gamma_3$, and $\pi_2$.

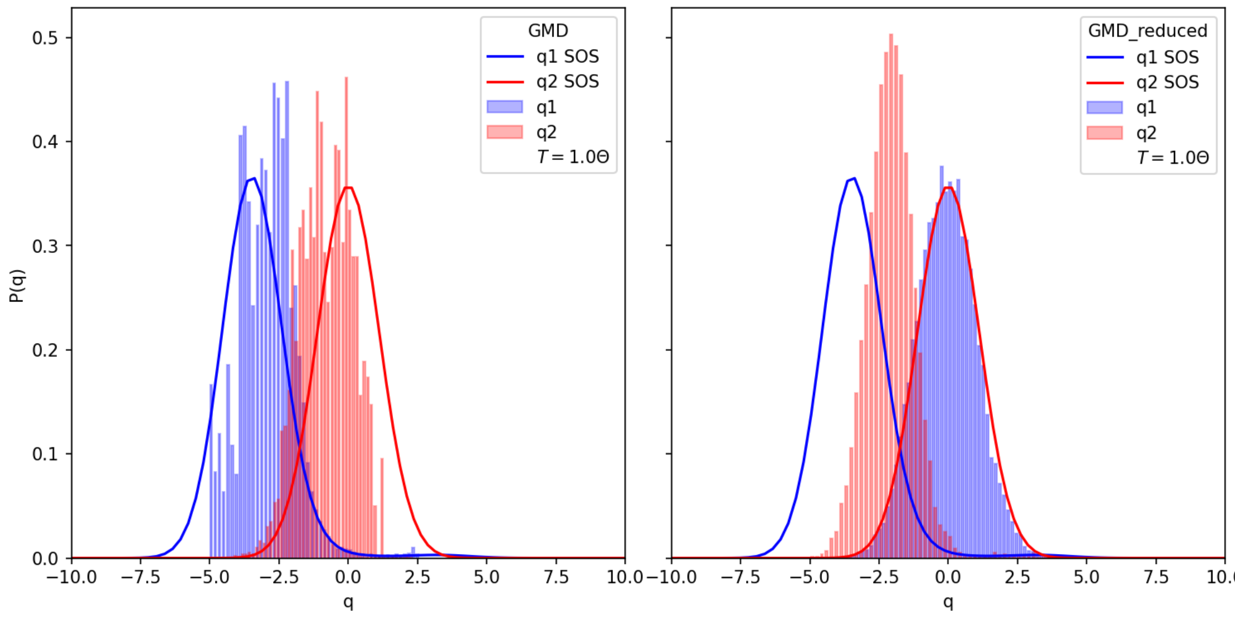

Figure 4.13: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 1.0\Theta$, $\gamma_3$, and $\pi_2$.

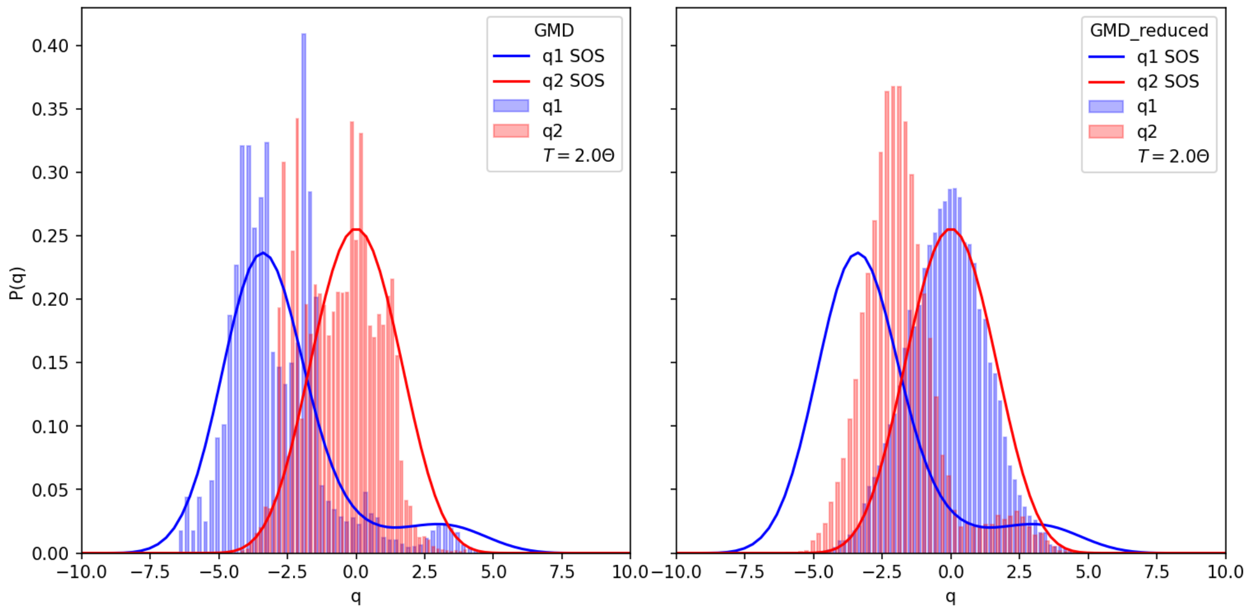

Figure 4.14: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 2.0\Theta$, $\gamma_3$, and $\pi_2$.

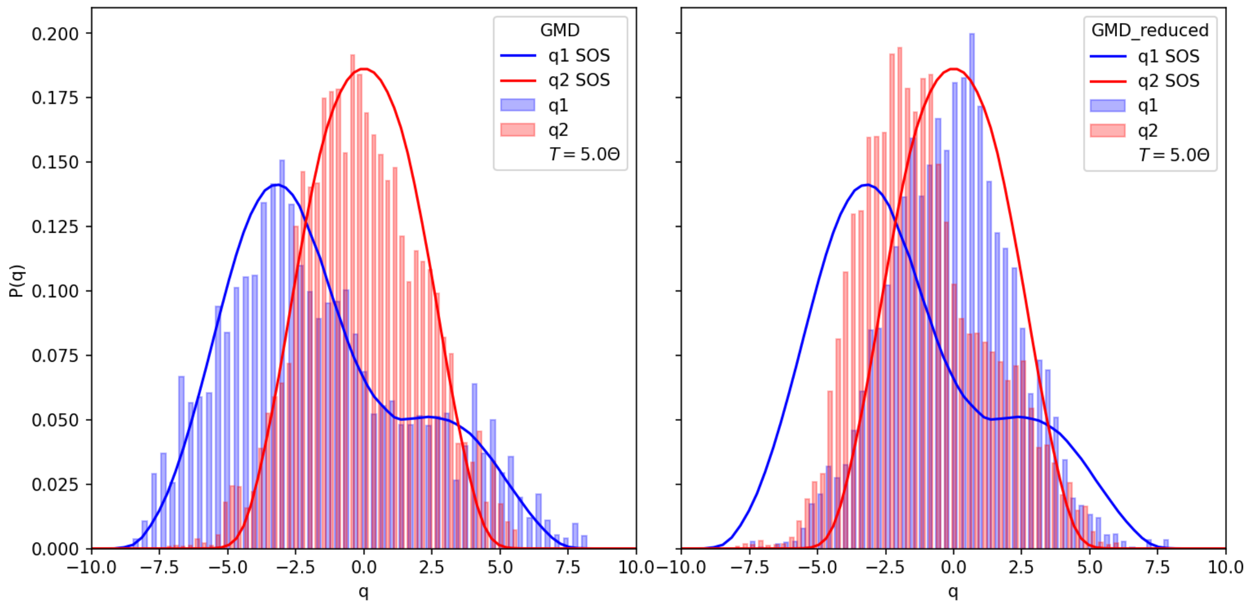

Figure 4.15: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 5.0\Theta$, $\gamma_3$, and $\pi_2$.

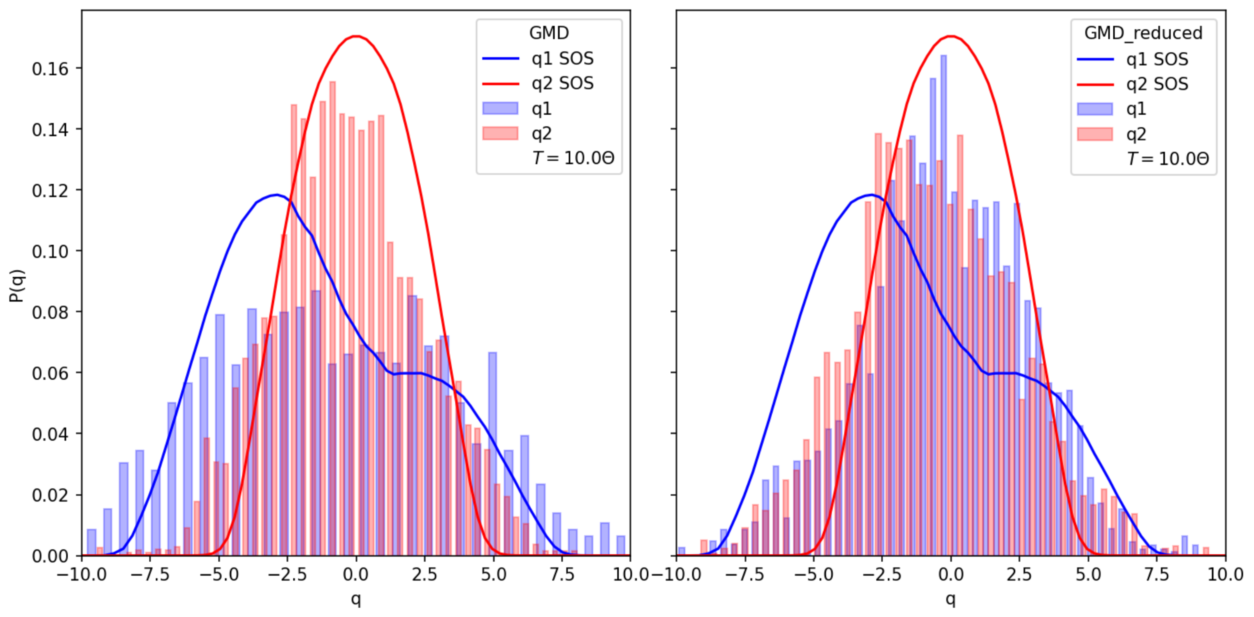

Figure 4.16: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 10.0\Theta$, $\gamma_3$, and $\pi_2$.

### 4.7.5 Intermediate Coupling ($\gamma_4$), Proposal $\pi_1$

For the Displaced system with ($\gamma_4$), both modes contribute. Now we struggle to exactly agree with the SOS, regardless of which method is used. In this case we use the basic proposal distribution $\pi_1$ defined by the harmonic components of the Hamiltonian.

For all of the following Figures in Section 3 the reader will notice two entries labelled `q2` on the legend. This is due to my treatment of the $q_2$ mode. Since the $q_2$ mode is symmetric, I divide the samples obtained by either method into two groups[10] which I plot as two separate histograms, along $+q$ and $-q$ x-axis components, respectively. Ideally the symmetry of the system should be captured in a converged simulation and this splitting treatment would not be needed. However, in reality, exceptionally high potential barriers between symmetry-equivalent regions of the phase space may trap configurations in a single region, leading to asymmetric results. An example of this barrier effect appears to be present in the $q_1$ distribution in Figures 4.4 and 4.9. A similar effect and separation can be seen in [48, Fig. 5 & 6].

The GMD method really struggles in Figures 4.17 to 4.19. It is in the "ballpark" but the shape of the distributions is not clearly defined. Once the temperature is high enough, $T \geq 5.0\Theta$ in Figure 4.20, it at least has the rough shape of the true $q_1$ and $q_2$ distributions.

The GMD-reduced method again produces much smoother distributions, but their displacement and qualitative shape is still dominated by the proposal distribution $\pi_1$ having $q_1$ displacement but no $q_2$ displacement. Interestingly, it appears to capture some of the bi-modal nature of the true $q_2$ distribution in Figure 4.17, although that seems to disappear as soon as $T \geq \Theta$.

in Figure 4.21 we continue to see the behaviour of the GMD-reduced method being able to do a better job of capturing the $q_1$ distribution density in the positive regime. The shape of the GMD $q_1$ distribution tapering off around $q_1 \approx 1.75$ and the GMD-reduced $q_1$ distribution having a dip in the $[-1.75, 1.75]$ regime is very similar to Figures 4.10 and 4.15.

Overall the GMD-reduced method fails to correctly describe the $q_1$ and $q_2$ distributions, while the GMD method does somewhat reasonable at high temperature.

[10]If I have picked sufficiently spaced samples then they should be independent. Therefore the manner in which I divide up the samples should not effect the resulting Figures.

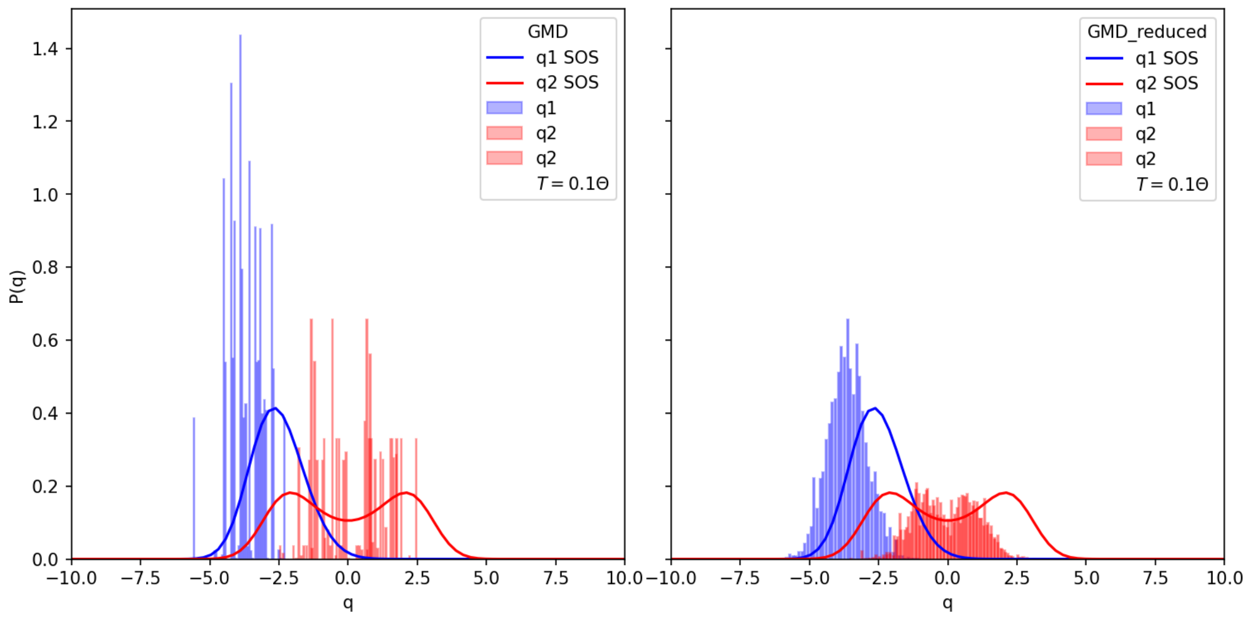

Figure 4.17: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 0.1\Theta$, $\gamma_4$, and $\pi_1$.

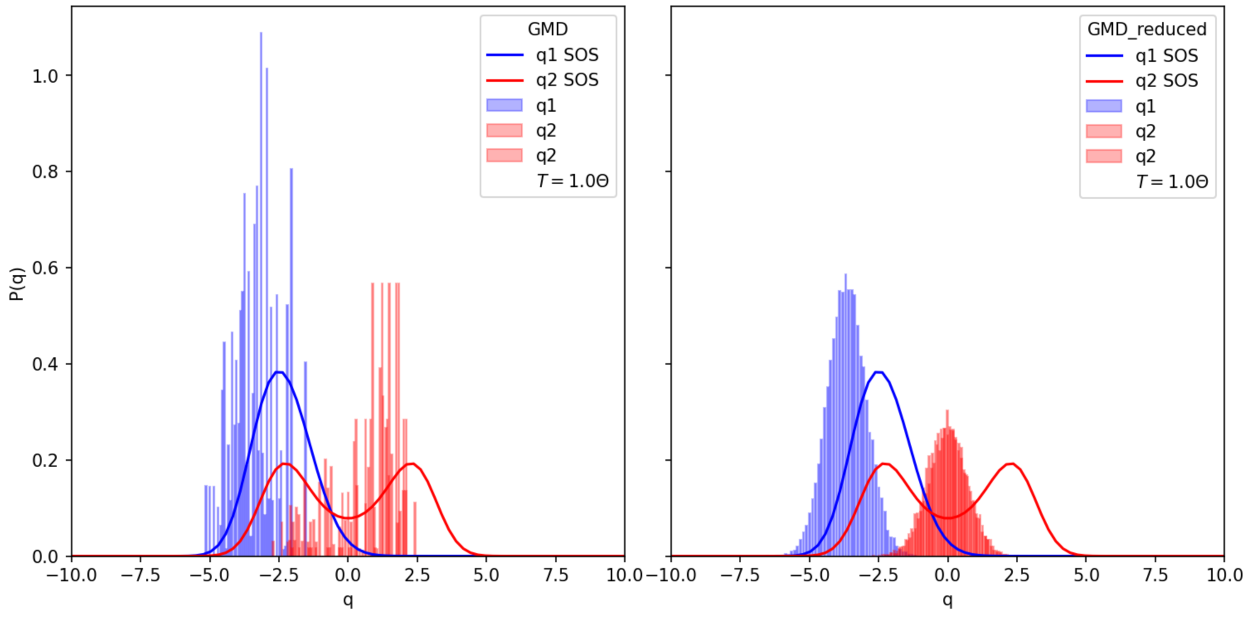

Figure 4.18: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 1.0\Theta$, $\gamma_4$, and $\pi_1$.

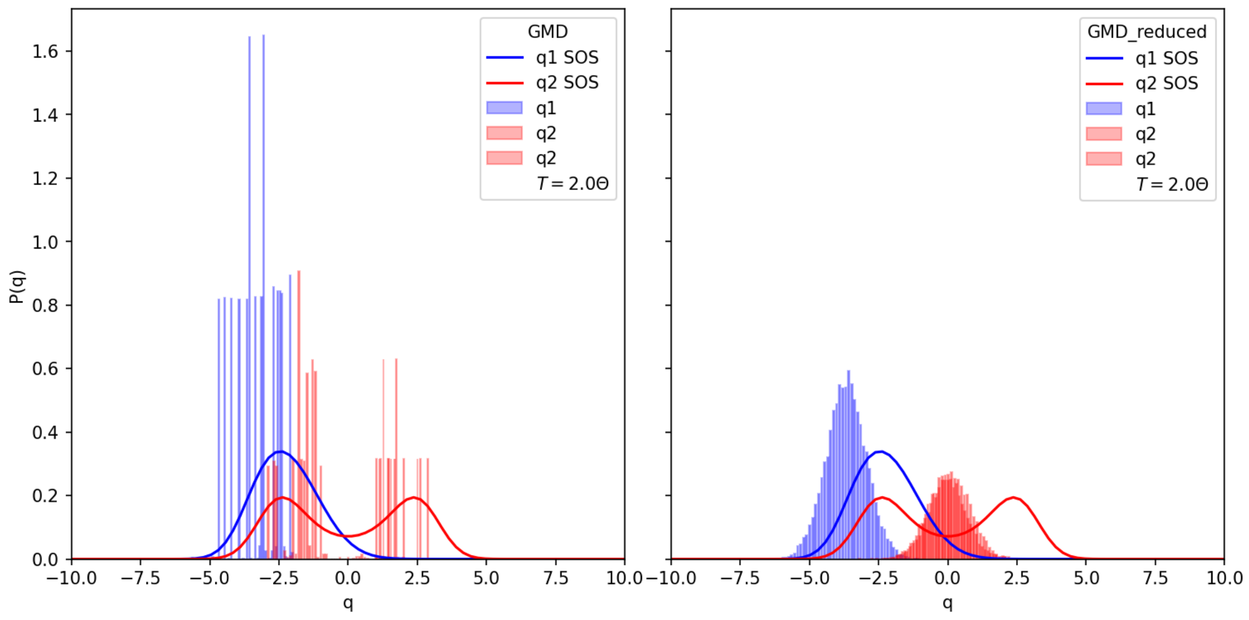

Figure 4.19: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 2.0\Theta$, $\gamma_4$, and $\pi_1$.

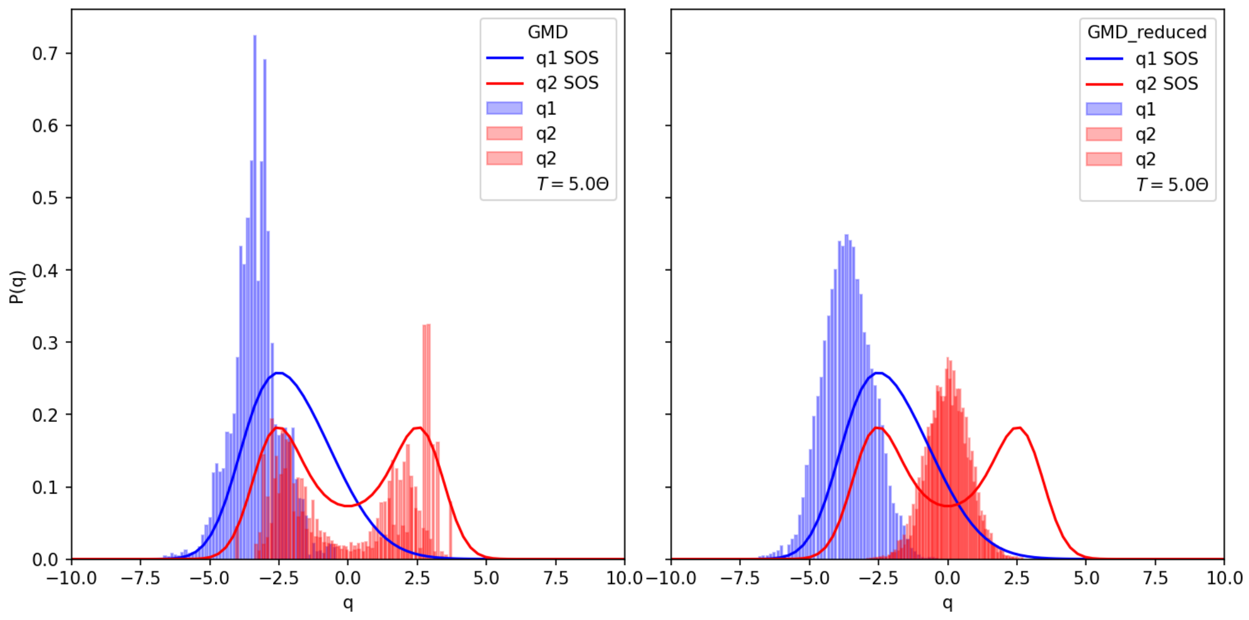

Figure 4.20: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 5.0\Theta$, $\gamma_4$, and $\pi_1$.

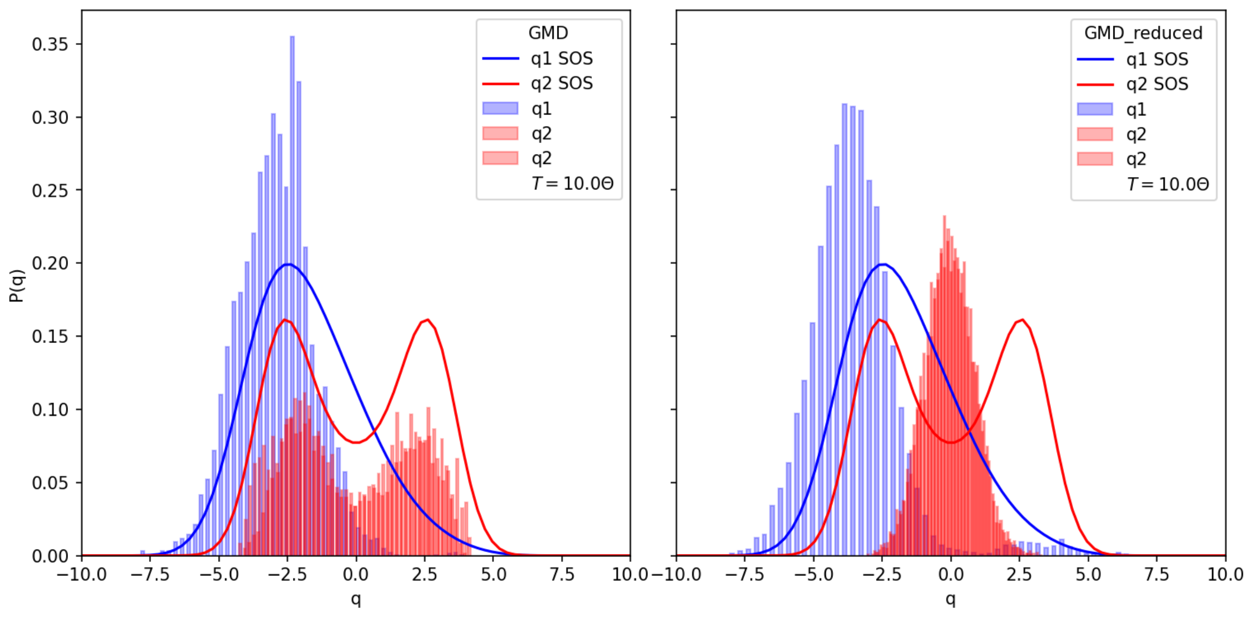

Figure 4.21: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 10.0\Theta$, $\gamma_4$, and $\pi_1$.

### 4.7.6 Intermediate Coupling ($\gamma_4$), Proposal $\pi_2$

For the Displaced system with ($\gamma_4$), both modes contribute. In this case, we use the second proposal distribution $\pi_2$ which we expect should help with capturing the $q_2$ contributions that are now present. We also only cover temperatures of 2.0Θ, 5.0Θ, 10.0Θ as the simulations at 0.1Θ, and 1.0Θ suffered numerical errors due to the low value of $T$.

It is clear in all three Figures 4.22 to 4.24 that using the second proposal distribution $\pi_2$ drastically improves both the GMD and GMD-reduced methods.

However, it seems that for this intermediate coupling $\gamma_4$, the true $q_1$ distribution still has some appreciable weight in the negative regime. Because $\pi_2$ does has no $q_1$ contributions, the GMD-reduced method can only produce a distribution centered at zero, similar to what we've seen in Section 4.7.4. We also see the same kind of behaviour in the GMD method when we compare with Section 4.7.4: that it does a much better job capturing the true distribution's shapes when using $\pi_2$. The temperature range in which it performs the best is slightly different. in Figure 4.24 the GMD method almost entirely recreates the true distribution. With more samples I expect them to agree exactly.

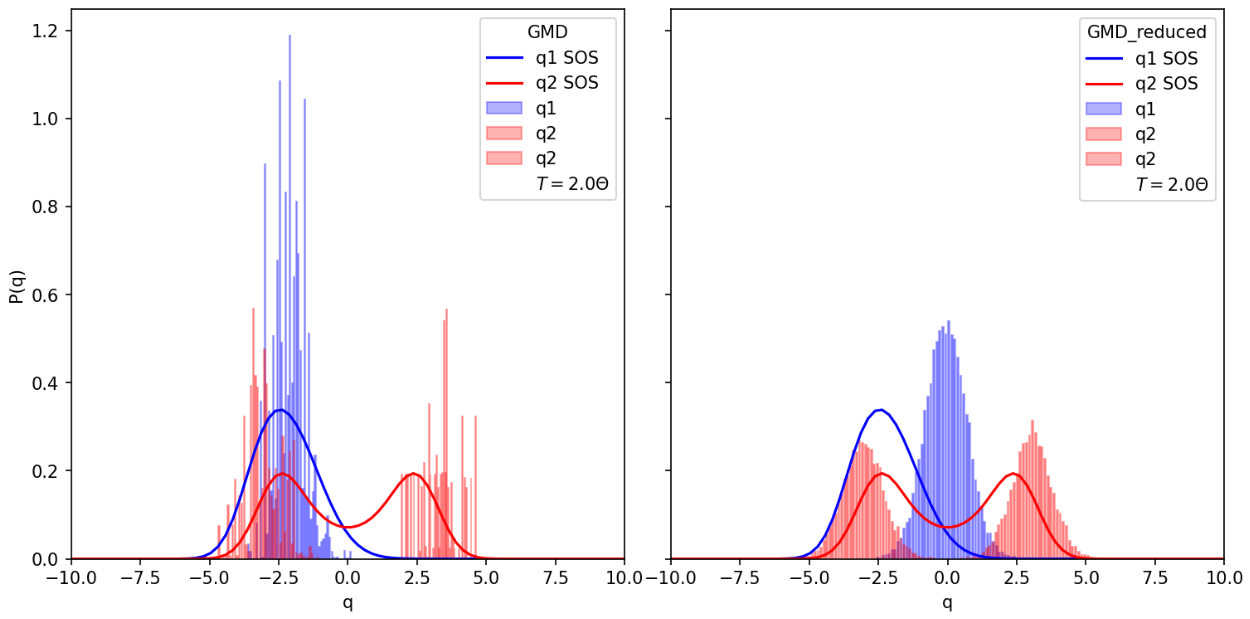

Figure 4.22: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 2.0\Theta$, $\gamma_4$, and $\pi_2$.

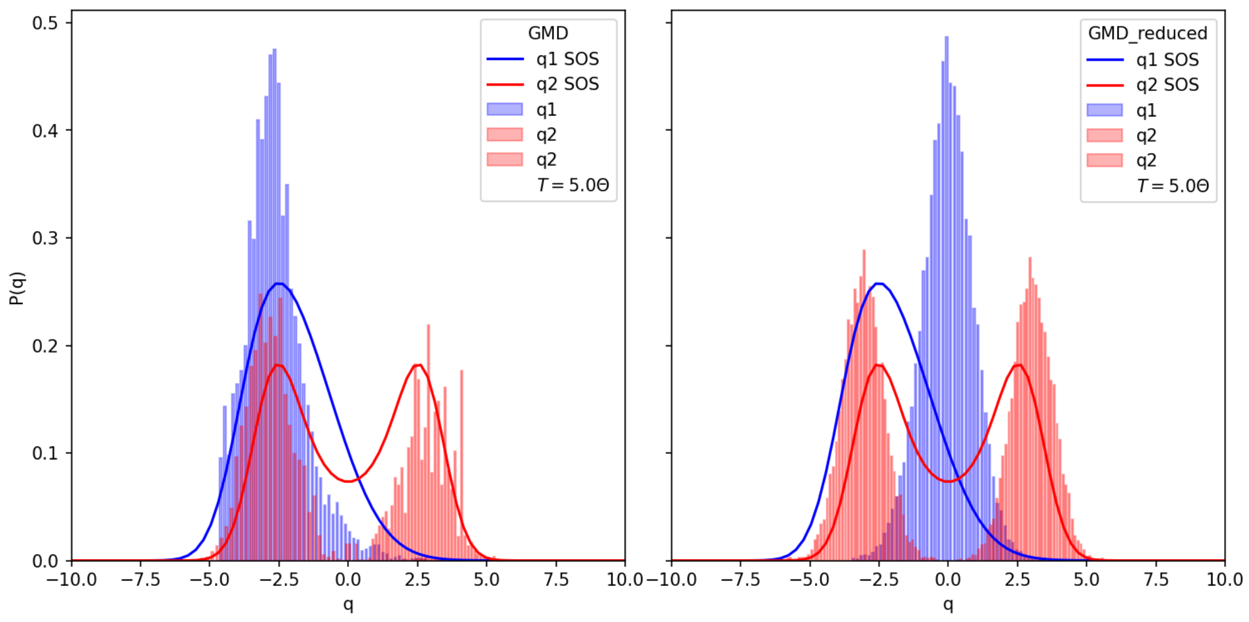

Figure 4.23: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 5.0\Theta$, $\gamma_4$, and $\pi_2$.

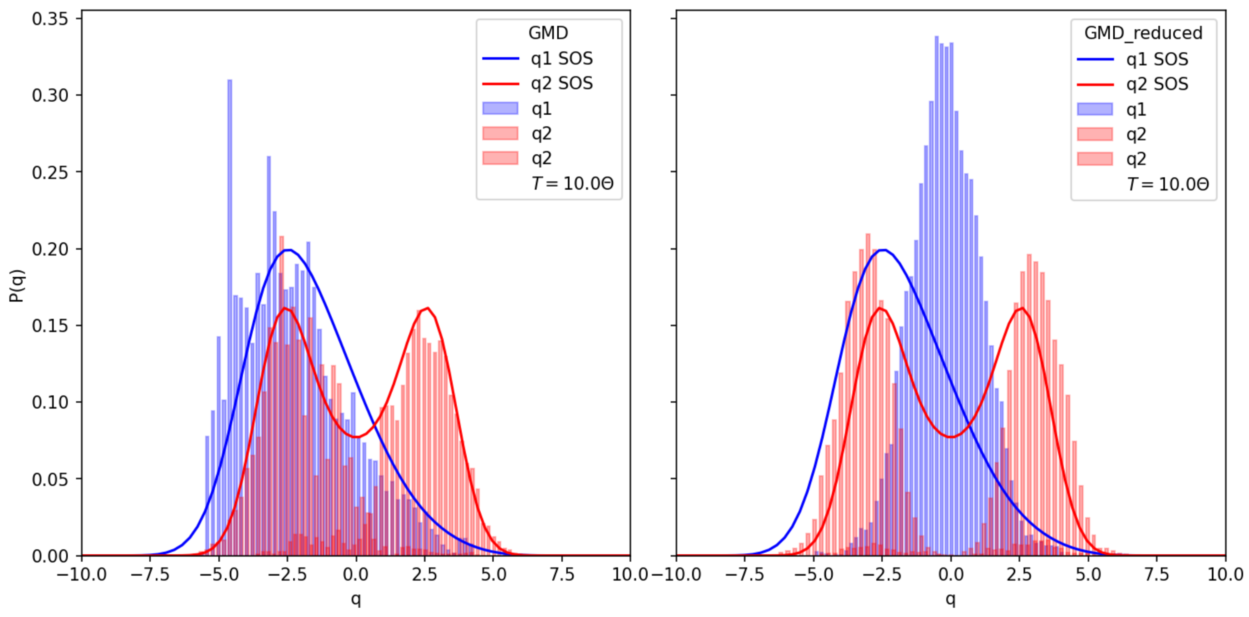

Figure 4.24: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 10.0\Theta$, $\gamma_4$, and $\pi_2$.

### 4.7.7 Strong Coupling ($\gamma_5$), Proposal $\pi_1$

For the Displaced system with strong coupling ($\gamma_5$), the $q_2$ mode dominates. In this case we use the basic proposal distribution $\pi_1$ defined by the harmonic components of the Hamiltonian. We expect this proposal distribution to perform poorly, and it does. Both the GMD and GMD-reduced methods are completely "consumed" by the proposal distribution and cannot get anywhere close to the true distributions in either Figure 4.25 or Figure 4.26. Interestingly enough, we see the same behaviour from the GMD-reduced method in the $[1.75, 5]$ regime that we've seen previously in Figures 4.4, 4.9, 4.21 and 4.26.

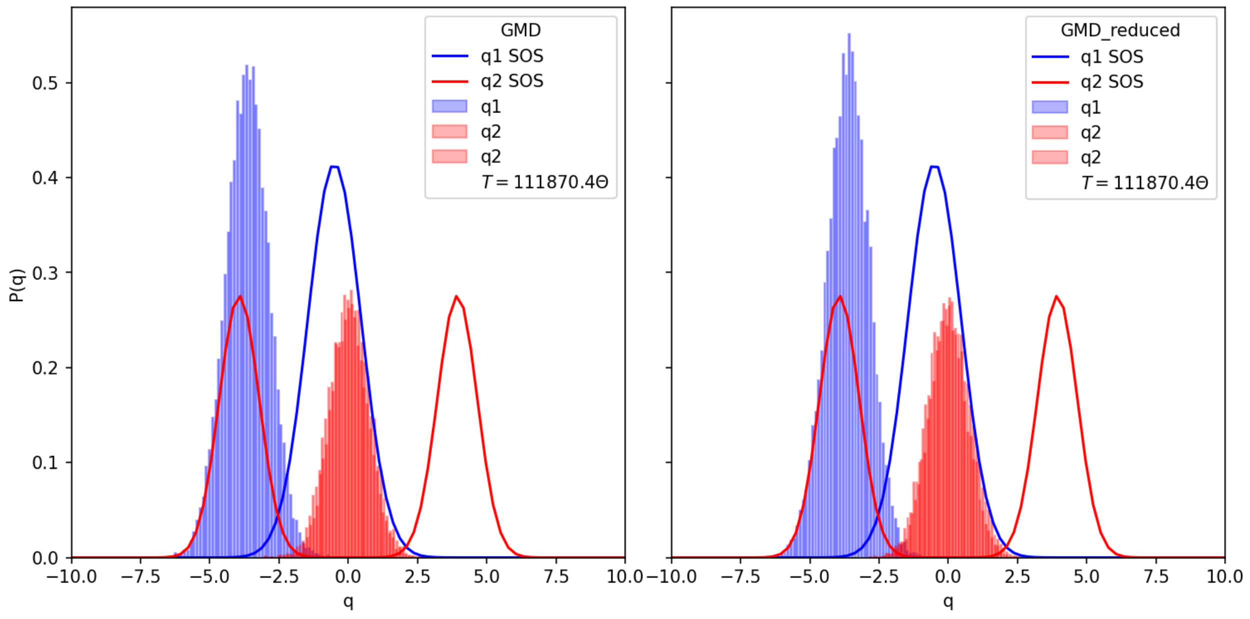

Figure 4.25: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 100K$, $\gamma_5$, and $\pi_1$.

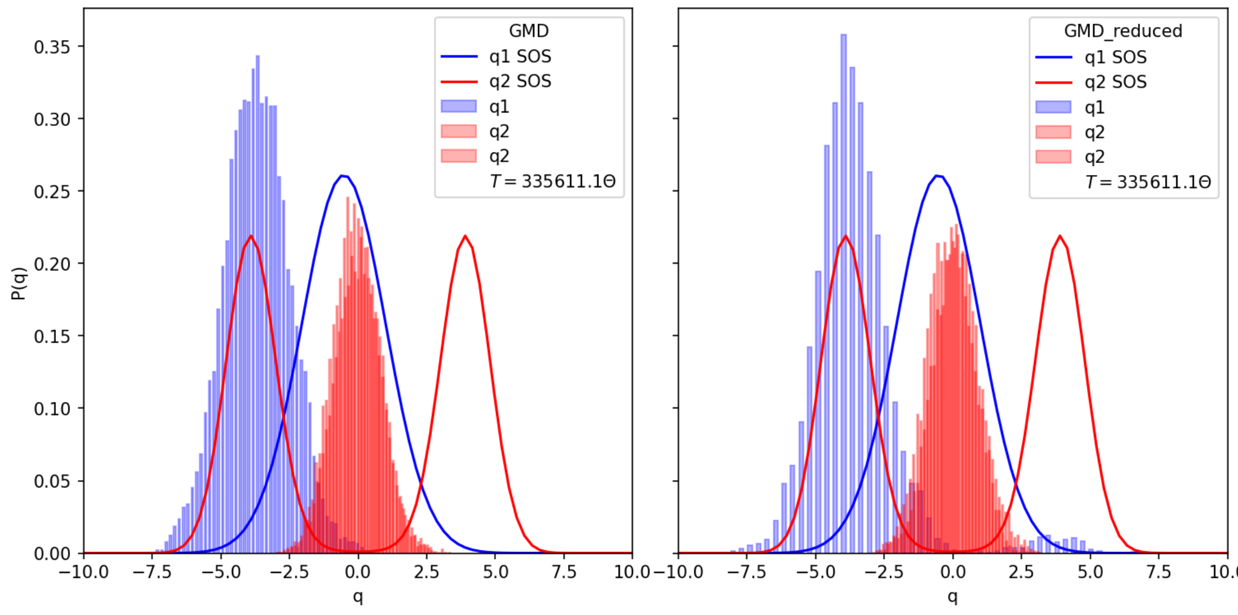

Figure 4.26: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 300K$, $\gamma_5$, and $\pi_1$.

### 4.7.8 Strong Coupling ($\gamma_5$), Proposal $\pi_2$

For the Displaced system with strong coupling ($\gamma_5$), the $q_2$ mode dominates. In this case we use the second proposal distribution $\pi_2$, which we expect should help with capturing the $q_2$ contributions that are now present. The results are actually quite spectacular, with both methods performing much better in Figures 4.27 and 4.28 than Figures 4.25 and 4.30. The GMD-reduced method really shines here, almost exactly matching the true distributions. Again we see this consistent trend to be slightly off-centre from the true distributions.

The GMD method struggles much more than the GMD-reduced method. in Figure 4.28 it does seem to have most of its $q_1$ in the same general area as the true distribution. With many more samples, it seems reasonably likely that it would eventually have the same general shape. The bi-modal nature of the $q_2$ distribution is at least present now, although the displacement and shapes are unfortunately all wrong.

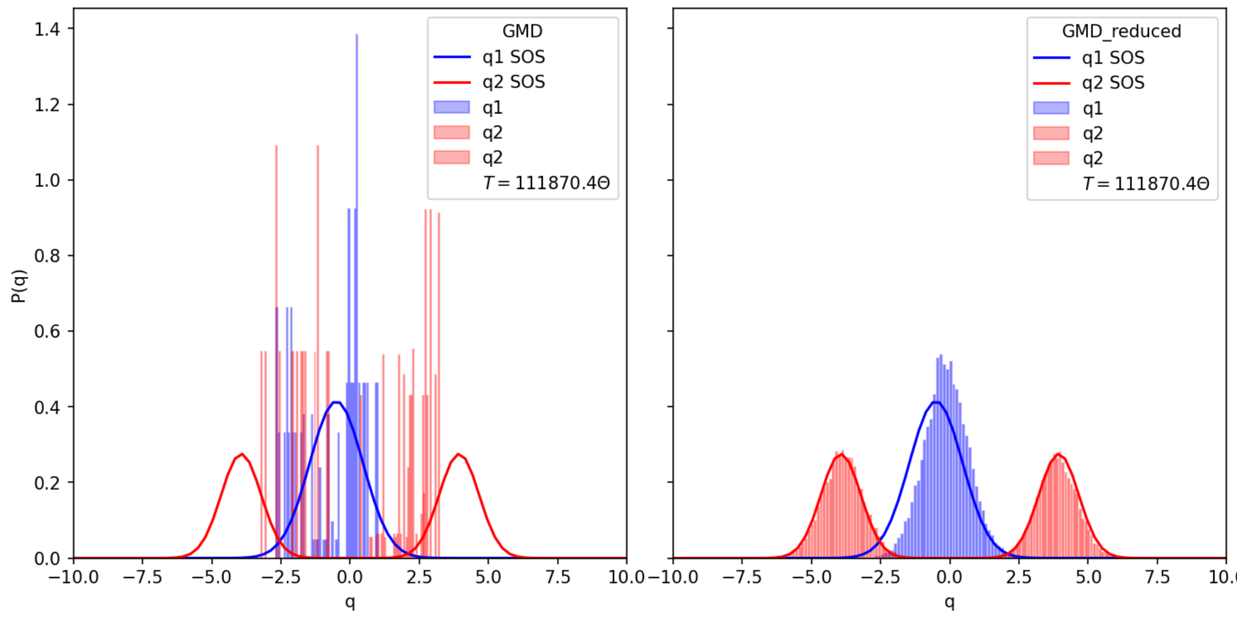

Figure 4.27: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 100K$, $\gamma_5$, and $\pi_2$.

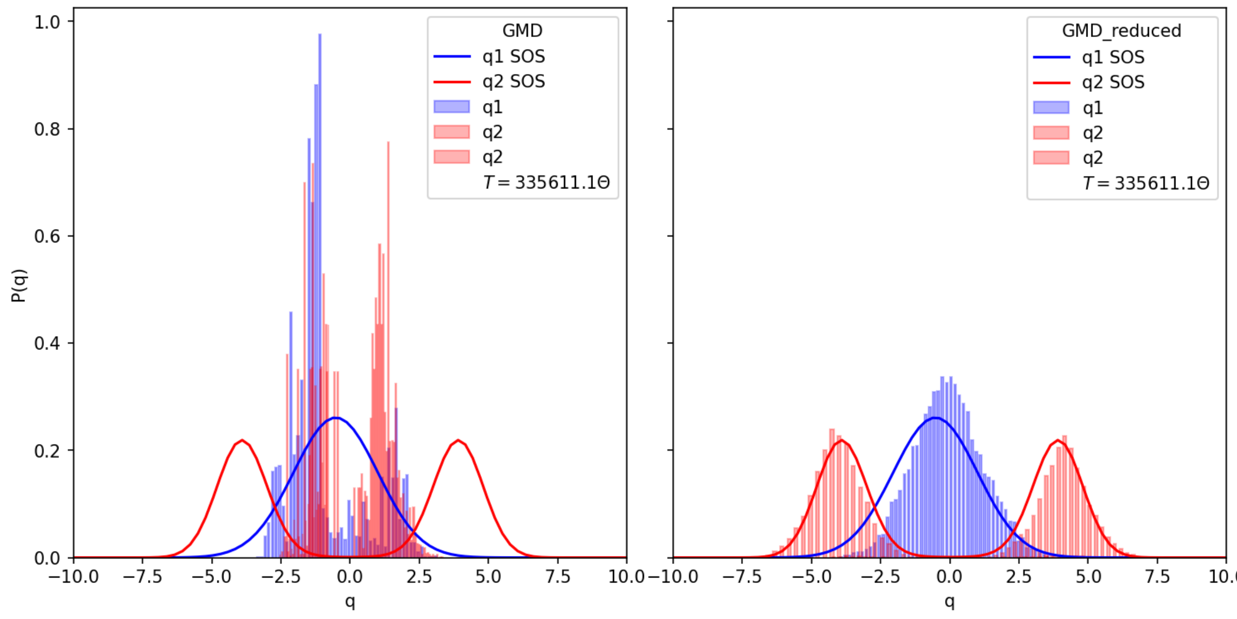

Figure 4.28: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 300K$, $\gamma_5$, and $\pi_2$.

### 4.7.9 Strong Coupling ($\gamma_6$), Proposal $\pi_1$

In this case, we use the basic proposal distribution $\pi_1$ defined by the harmonic components of the Hamiltonian. We expect this proposal distribution to perform poorly, and it does. Both the GMD and GMD-reduced methods are completely "consumed" by the proposal distribution and cannot get anywhere close to the true distributions in either Figure 4.29 or Figure 4.30. Interestingly enough, we see the same behaviour from the GMD-reduced method in the $[1.75, 5]$ regime that we've seen previously in Figures 4.4, 4.9, 4.21 and 4.26.

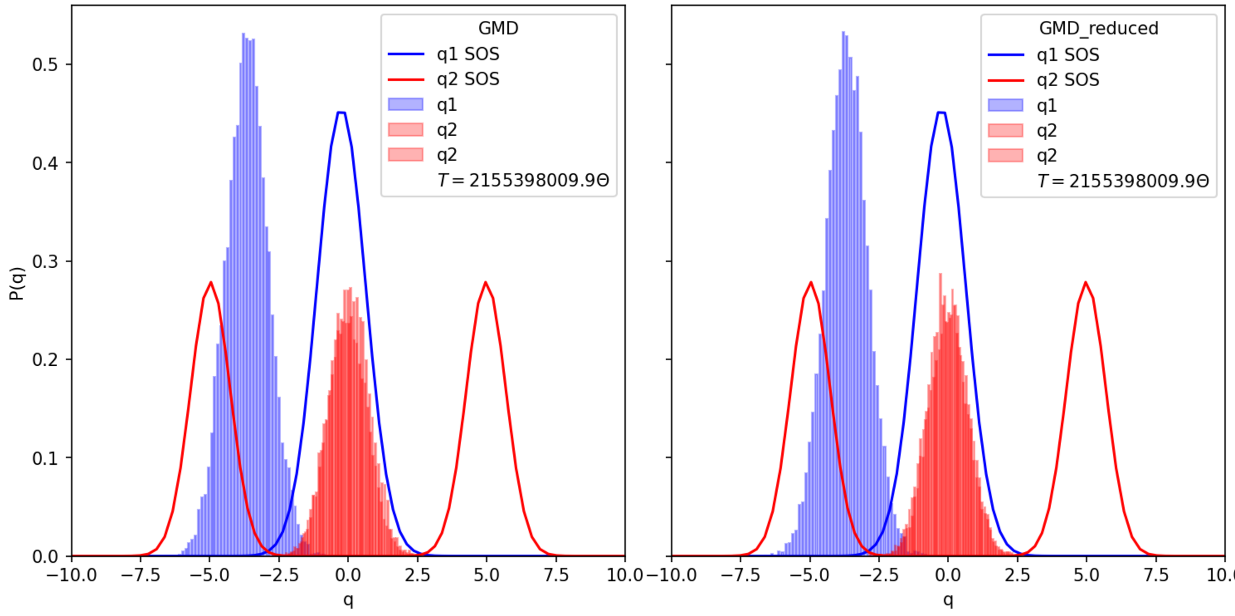

Figure 4.29: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 100K$, $\gamma_6$, and $\pi_1$.

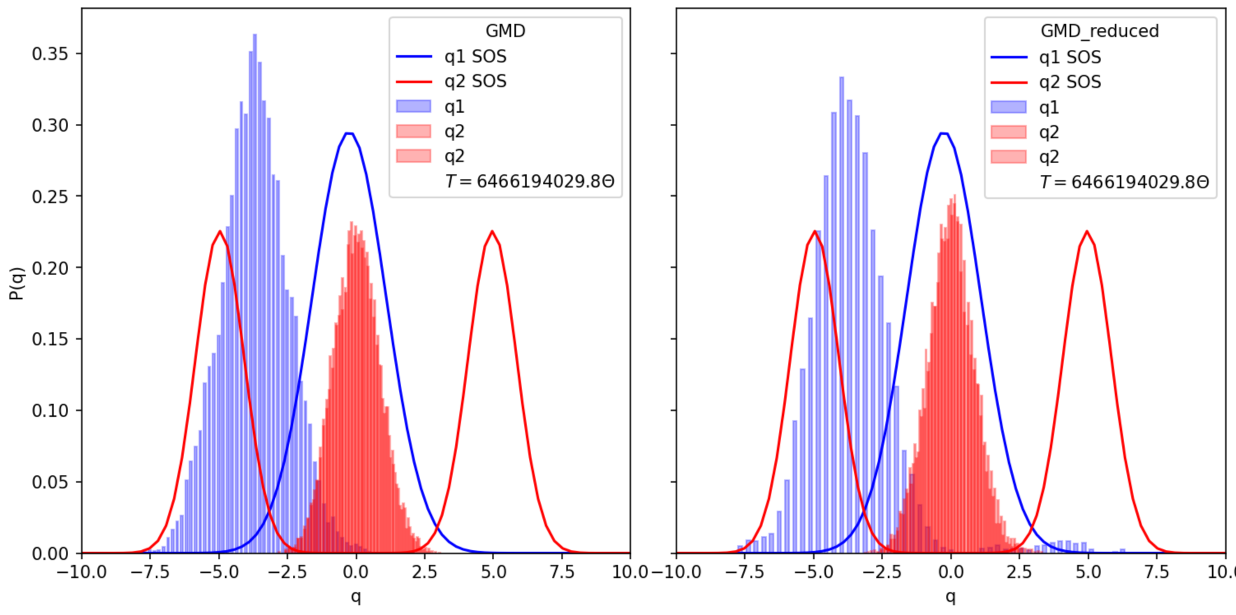

Figure 4.30: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 300K$, $\gamma_6$, and $\pi_1$.

### 4.7.10 Strong Coupling ($\gamma_6$), Proposal $\pi_2$

In this case, we use the second proposal distribution $\pi_2$, which we expect should help with capturing the dominant $q_2$ contributions. The results are actually quite spectacular, with both methods performing much better in Figures 4.31 and 4.32 than Figures 4.29 and 4.30. The GMD-reduced method really shines here, almost exactly matching the true distributions. Again we see this consistent trend to be slightly off-centre from the true distributions.

The GMD method struggles much more than the GMD-reduced method, although at least now it captures most of the $q_1$ distribution. With a much larger number of samples it seems reasonably likely that it would eventually have the same general shape. The bi-modal nature of the $q_2$ distribution is at least present now, although the displacement and shapes are unfortunately all wrong.

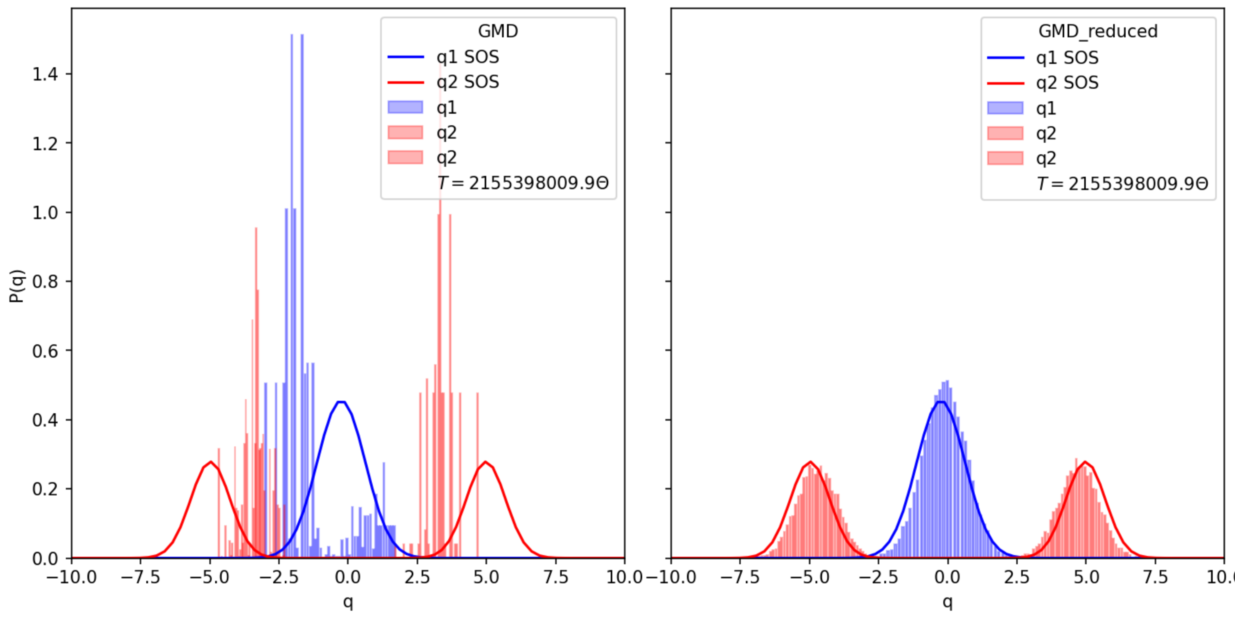

Figure 4.31: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 100K$, $\gamma_6$, and $\pi_2$.

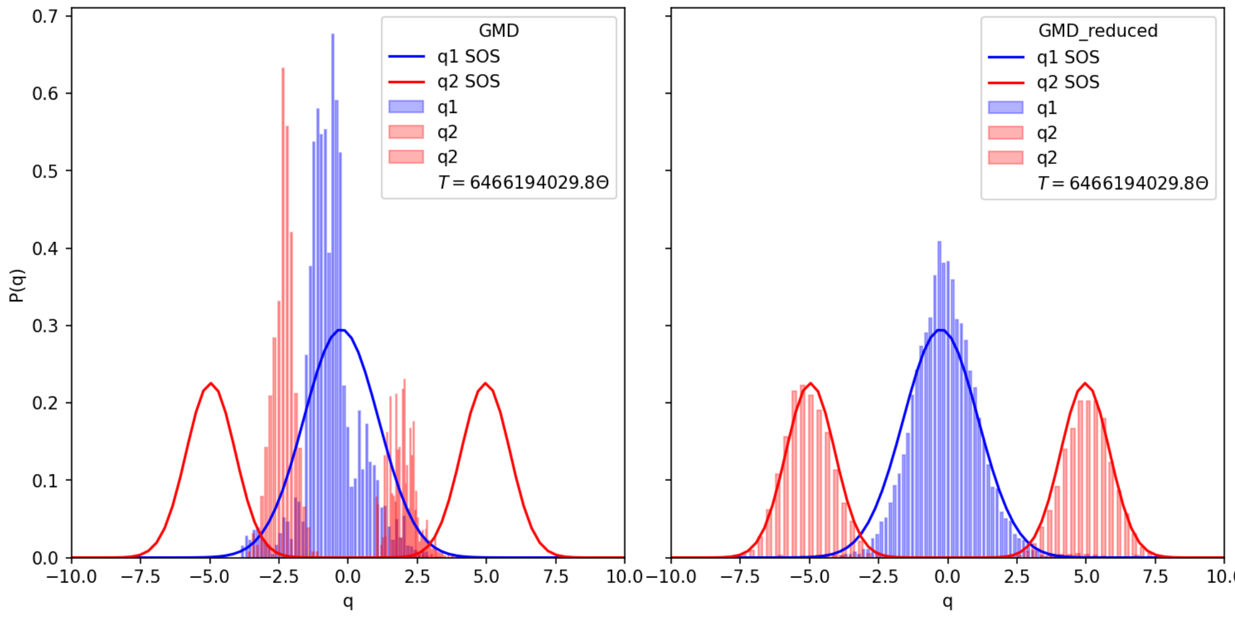

Figure 4.32: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 300K$, $\gamma_6$, and $\pi_2$.

### 4.7.11 Strong Coupling ($\gamma_6$), Proposal $\pi_3$

In this case, we use the third proposal distribution $\pi_3$, which we hope will fix the issue associated with $q_1$ while still capturing the dominant $q_2$ contributions. Overall, the attempt to use $\pi_3$ is a failure and Figures 4.33 and 4.34 show much worse results than Figures 4.31 and 4.32 for both methods. Again we see that the GMD-reduced method is strongly influenced by the proposal distribution with the $q_1$ histogram having the right shape but being displaced from the SOS.

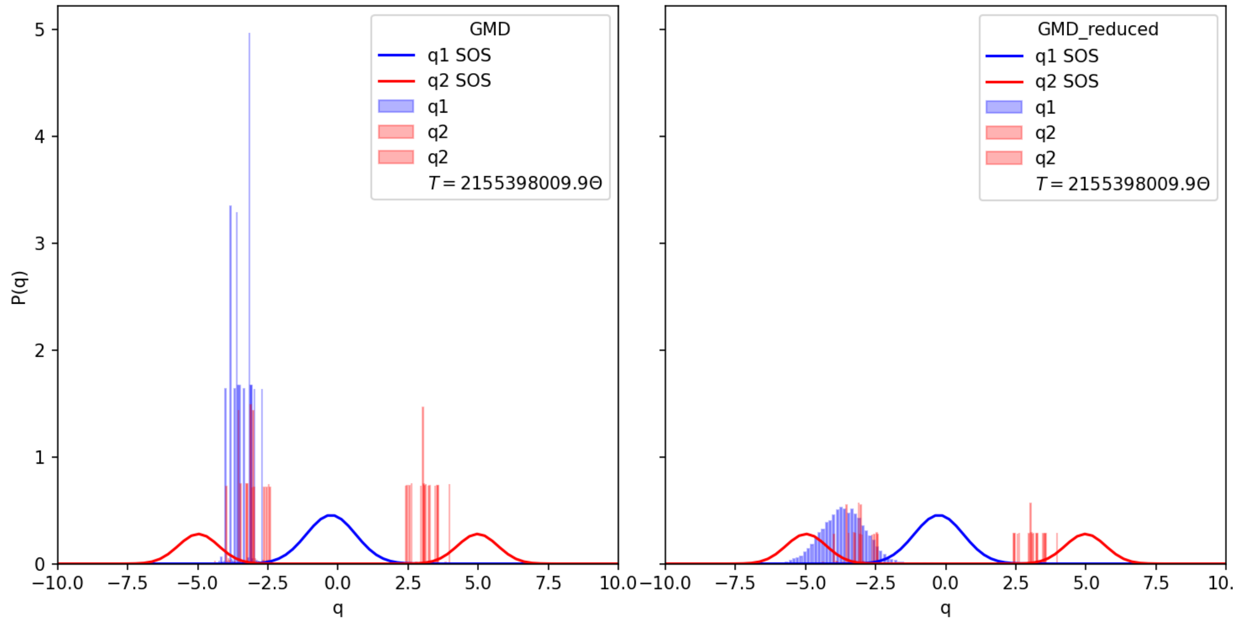

Figure 4.33: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 100K$, $\gamma_6$, and $\pi_3$.

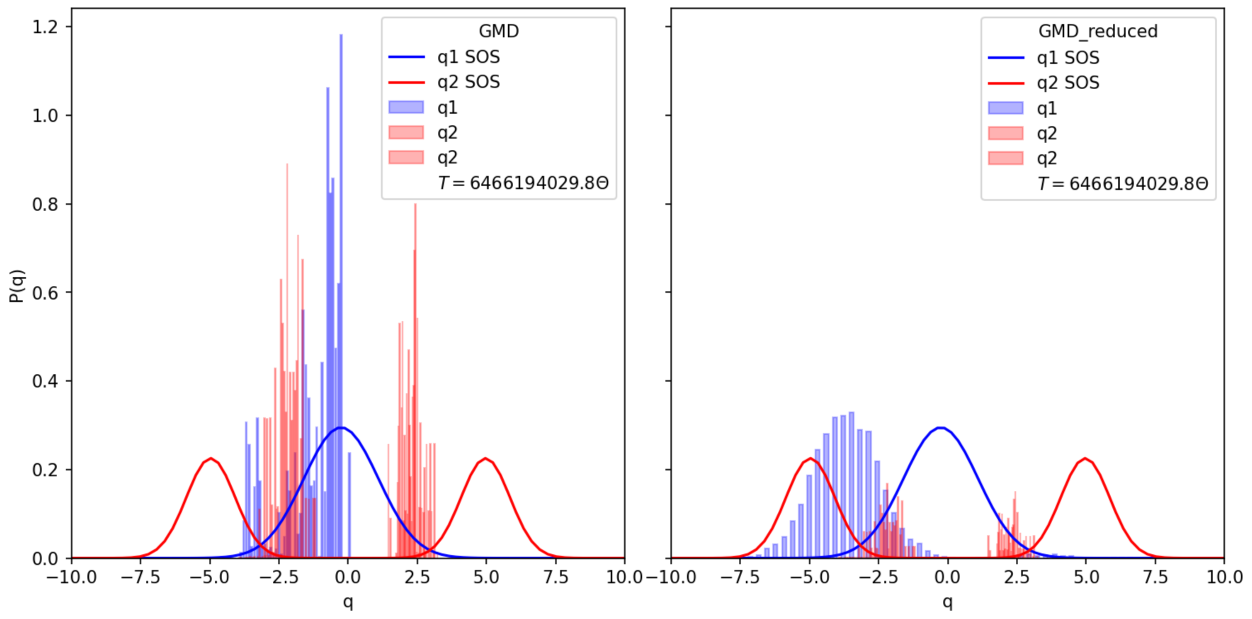

Figure 4.34: Comparison of GMD & GMD-reduced PIMC method's approximation of the true distribution of normal modes $q_1$, $q_2$. All results calculated at $T = 300K$, $\gamma_6$, and $\pi_3$.

## 4.8 Concluding Remarks

In this Chapter I presented my new MH approach for calculating thermodynamic properties of vibronic models. I derived equations for the straightforward **direct** method to explain the basic approach. However in practice this method fails to construct a Markov Chain. Uniform sampling had great difficulty dealing with the *non-stoquastic* nature of the Hamiltonian. I then presented a new approach: the GMD method which evaluates exactly, instead of *stochasitcally*, the electronic surfaces.

Both the GMD and GMD-reduced methods were compared against SOS results for the small Displaced system. In the Results section it was clear that GMD-reduced method was strongly impacted by the choice of proposal distribution. At times this meant the GMD method preformed better. In the strongly coupled systems, however, the exact evaluation of the electronic surfaces produced drastically different results to the GMD method, as seen in Figure 4.31. I purposely presented results where the proposed sampling methods work well and when they fail. Doing so allows one to understand the applicability of the approach and identify areas where the method can be improved.

In light of the present results, it is clear that better proposal distributions are required in order to simulate the full range of coupling strengths. One possibility is to use a larger number of Gaussian distributions in the GMD to distribute the sampling over a broader range of configurations. As the number of Gaussian distributions will be greater than the actual number of electronic states, new transition rules will have to be derived in the MH algorithm. This line of investigation is currently underway and will be the focus of future work.

This MH can be extended to computing the expectation value of a systems energy in the near future. This method also can be applied to new vibronic models, particularly those that incorporate spin-orbit coupling in a plug-and-play fashion.

# Chapter 5

# Distinguishing pairings with `termfactory`

## STATEMENT OF CONTRIBUTION

As this section discusses work that was a collaboration with two other researchers it is important to clearly state my contributions.

I solely developed and implemented the software package `termfactory` [49] for generating LaTeX equations, and Python equations, and for evaluating those Python equations. [1]

The `t-amplitudes` [50] software package was developed collaboratively between S. Bao and myself. Over the course of development, I was responsible for the majority of the programming and project management components. Songhao's contribution was focused primarily on derivation of equations, testing, and prototyping new ansatz.

The theoretical foundation for both of these packages and the method for generating spectra was developed by M. Nooijen and S. Bao. I did not directly contribute to that theory; I provided a robust software ecosystem to facilitate the development of the theory implemented in `t-amplitudes` as well as a technical collaborator in accelerating ansatz prototyping.

## Outline

In this chapter, I will detail my approach to solving a general problem. in Chapter 6 I will give context for applying that solution to a specific case, and discuss how that allows me to generate theoretical spectra of vibronic models. A large portion of this chapter is detailing the mathematical problems addressed by `termfactory`.

It is important to emphasize to the reader than the ongoing goal of the generator is to write a robust software package that removes natural human error. A primary motivation behind the automation is that there are a number of variations on the theme and a code/equation generator is very beneficial in such situations. The `termfactory` package was

[1]The `termfactory` and `t-amplitudes` packages were migrated to GitHub with the assistance of Lucas Price.

fundamental to the success of the `t-amplitudes` software package, and therefore to the goal of reliable spectra generation. Key contributions from this software are the drastic reduction in time required to produce vibrational electronic coupled cluster (VECC) equations of motion (EOM), as well as the code required to compute those equations. It generally takes on the order of a few seconds to generate the necessary EOM using `termfactory`, even for hundred's of equations, whereas manually deriving these equations by hand can easily take days, or weeks if any mistakes are made [2]. `termfactory` offers time savings and a guarantee of certainty that the equations are consistent.

## 5.1 Motivating Problem

To simplify what would otherwise be a lengthy and dense explanation of the theoretical background, I will explain the problem in the simplest form and give very general definitions. How this connects with the background theory and leads to the computation of simulated spectra will be covered in Chapter 6. Additional background has been provided in Section 2.3.

In general, the motivating problem can be expressed as computing an expectation value:

$$\left\langle \hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}}\cdots \right\rangle . \tag{5.1}$$

Each operator, say $\hat{\mathcal{A}}$, has electronic DoF

$$\hat{\mathcal{A}} = \sum_{a,b} |a\rangle \langle b| \, \hat{A}^{a}_{b}, \tag{5.2}$$

expanding one particular component:

$$\hat{A}^{a}_{b} = \sum_{M+N\leq R} \hat{A}^{a,M}_{b,N}\,, \tag{5.3}$$

where $M$ counts the number of vibrational creation operators, $N$ counts the number of vibrational annihilation operators and $R$ defines the maximum rank of an operator [3]. When one of the labels $M$, $N$ are zero they will be omitted: $\hat{A}^{a}_{b,N}$ or $\hat{A}^{a,M}_{b}$, and if both labels are zero then a lower label of zero will be used: $\hat{A}^{a}_{b,0}$.

I can now give a full expansion in terms of the general operators $\hat{A}$ in Equation (5.3):

$$\hat{A}^{a,M}_{b,N} = \frac{1}{N!}\frac{1}{M!} \sum_{i_1,i_2,\ldots,i_M} \sum_{j_1,j_2,\ldots,j_N} \left( A^{a,i_1,i_2,\ldots,i_M}_{b,j_1,j_2,\ldots,j_N} \right) \left( \hat{i}^{\dagger}_1 \hat{i}^{\dagger}_2 \ldots \hat{i}^{\dagger}_M \, \hat{j}_1 \hat{j}_2 \ldots \hat{j}_N \right), \tag{5.4}$$

[2]These are not estimates, but in fact recounting of actual time spans during this research.
[3]The basic definition of creation and annihilation operators is in Section 2.3.1.

which, in general, can be treated as three components:

$$\hat{A}^{a,M}_{b,N} = \Big(\text{prefactors}\Big) \sum_{\text{labels}} \Big(\text{coefficients}\Big)\Big(\text{operators}\Big), \tag{5.5}$$

where the two primary components are the complex valued coefficient amplitudes and the collection of creation and annihilation operators. The amplitudes are symmetric under both, permutations of the $i$ labels with themselves, and permutations of the $j$ labels with themselves. Sums run over all normal modes $N_{\text{modes}}$ in the system $i, j \in \{1, 2, \ldots, N_{\text{modes}}\}$. Equation (5.4) therefore provides a full prescription for any operator $\hat{\mathcal{A}}$ in Equation (5.1). Electronic labels $a, b$ will always act as in matrix multiplication and will be suppressed for the remainder of this chapter.

Equations of the form 5.1 can most easily be evaluated by applying Wick's theorem. This process involves pairing all annihilation and creation operators. These pairings occur in two ways. For normal modes $i$ and $j$, the pairings are:

$$\overset{\sqcap}{\hat{i}^\dagger \hat{j}} = \bar{f}_i \delta_{ij}, \tag{5.6a}$$

$$\overset{\sqcap}{\hat{j} \hat{i}^\dagger} = f_i \delta_{ij}, \tag{5.6b}$$

where

$$f_i = 1 + \bar{f}_i. \tag{5.7}$$

In general $\bar{f}_i$ and therefore $f_i$ are complex numbers but in specific cases

$$\bar{f}_i = 0 \qquad f_i = 1. \tag{5.8}$$

The application of Wick's theorem means that for the product of two normal ordered [4] operators $\hat{\mathcal{A}}\hat{\mathcal{B}}$ one **only** includes pairings between the operators $\hat{\mathcal{A}}$, $\hat{\mathcal{B}}$ and **never** pairings within an operator, where both $\hat{i}^\dagger$ and $\hat{j}$ come from the same normal ordered operator $\hat{\mathcal{A}}$. Pairing occurs between a $\hat{i}^\dagger$ and a $\hat{j}$ operator, each coming from a different normal ordered operator $\hat{\mathcal{A}}, \hat{\mathcal{B}}, \hat{\mathcal{C}}, \hat{\mathcal{D}}, \cdots$. Evaluating equations of this form involves summing over all possible normal mode labels:

$$\left\langle \sum_{\text{all labels}} \hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}}\cdots \right\rangle. \tag{5.9}$$

Since every pairing of two normal mode labels $i$ and $j$ produces a pair-like Kronecker delta $\delta_{ij}$, then the labels can simply be equated, while reducing the summation labels.

### 5.1.1 Example

We will examine a sizeable example to see how this pairing and equating process unfolds. Consider three operators $\hat{\mathbf{A}}^M_N, \hat{\mathbf{B}}^M_N, \hat{\mathbf{C}}^M_N$, that we know are fully described by Equation (5.4).

[4] See Equation (3.37).

Given a specific choice of $M$ and $N$ values for each operator:

$$\hat{\mathbf{A}}^{2}_{1}, \qquad \hat{\mathbf{B}}^{1}_{2}, \qquad \hat{\mathbf{C}}^{1}_{1}, \tag{5.10}$$

having amplitudes and operators

$$\mathbf{A}^{i_1,i_2}_{j_1}\left(\hat{i}^{\dagger}_1\hat{i}^{\dagger}_2\hat{j}_1\right), \qquad \mathbf{B}^{i_3}_{j_2,j_3}\left(\hat{i}^{\dagger}_3\hat{j}_2\hat{j}_3\right), \qquad \mathbf{C}^{i_4}_{j_4}\left(\hat{i}^{\dagger}_4\hat{j}_4\right), \tag{5.11}$$

we can evaluate them as follows:

$$\frac{1}{1!2!}\frac{1}{2!1!}\frac{1}{1!1!}\sum_{\substack{i_1,i_2,i_3,i_4\\ j_1,j_2,j_3,j_4}}\mathbf{A}^{i_1,i_2}_{j_1}\left(\hat{i}^{\dagger}_1\hat{i}^{\dagger}_2\hat{j}_1\right)\mathbf{B}^{i_3}_{j_2,j_3}\left(\hat{i}^{\dagger}_3\hat{j}_2\hat{j}_3\right)\mathbf{C}^{i_4}_{j_4}\left(\hat{i}^{\dagger}_4\hat{j}_4\right). \tag{5.12}$$

Consider a possible pairing: $(\hat{i}^{\dagger}_1\hat{j}_2)$, $(\hat{i}^{\dagger}_2\hat{j}_3)$, $(\hat{i}^{\dagger}_3\hat{j}_4)$, $(\hat{j}_1\hat{i}^{\dagger}_4)$; equating the labels and substituting in Equations (5.6a) and (5.6b):

$$\frac{1}{4}\sum_{\substack{i_1,i_2,i_3,i_4\\ j_1,j_2,j_3,j_4}}(\bar{f}_{i_1}\bar{f}_{i_2}\bar{f}_{i_3}f_{i_4})\mathbf{A}^{i_1,i_2}_{j_1}\mathbf{B}^{i_3}_{j_2,j_3}\mathbf{C}^{i_4}_{j_4}\delta_{i_1j_2}\delta_{i_2j_3}\delta_{i_3j_4}\delta_{i_4j_1}, \tag{5.13}$$

reducing the summation labels:

$$\frac{1}{4}\sum_{i_1,i_2,i_3,i_4}(\bar{f}_{i_1}\bar{f}_{i_2}\bar{f}_{i_3}f_{i_4})\mathbf{A}^{i_1,i_2}_{i_4}\mathbf{B}^{i_3}_{i_1,i_2}\mathbf{C}^{i_4}_{i_3}\delta_{i_1i_1}\delta_{i_2i_2}\delta_{i_3i_3}\delta_{i_4i_4}, \tag{5.14}$$

which can be simplified to

$$\frac{1}{4}\sum_{i_1,i_2,i_3,i_4}(\bar{f}_{i_1}\bar{f}_{i_2}\bar{f}_{i_3}f_{i_4})\mathbf{A}^{i_1,i_2}_{i_4}\mathbf{B}^{i_3}_{i_1,i_2}\mathbf{C}^{i_4}_{i_3}, \tag{5.15}$$

and we see that the pairing $\mathbf{A}^{i_1,i_2}_{i_4}\mathbf{B}^{i_3}_{i_1,i_2}\mathbf{C}^{i_4}_{i_3}$ will have a complex valued contribution to its associated expectation value. Note that in the specific case where $\bar{f}_i = 0$ the pairing would instead have zero contribution to its expectation value due to the presence of $\bar{f}$'s. It is evident that explicitly writing out all these pairings and terms becomes quite tedious and confusing. So I will employ a short hand of only labeling the equivalent labels with unique letters.

For the pairing $\mathbf{A}^{i_1,i_2}_{i_4}\mathbf{B}^{i_3}_{i_1,i_2}\mathbf{C}^{i_4}_{i_3}$ I instead write $\mathbf{A}^{kl}_{n}\mathbf{B}^{m}_{kl}\mathbf{C}^{n}_{m}$ simplifying[5]eq. (5.15) to

$$\frac{1}{4}\sum_{klmn}(\bar{f}_k\bar{f}_l\bar{f}_mf_n)\mathbf{A}^{kl}_{n}\mathbf{B}^{m}_{kl}\mathbf{C}^{n}_{m}, \tag{5.16}$$

We can immediately see the usefulness of this compact notation when we consider all possible

[5]Replacing $i_1 \to k$, $i_2 \to l$, $i_3 \to m$, $i_4 \to n$ and using letters from the alphabet $k, l, m, n, o, p, \cdots$ which do not include $i$ or $j$.

pairings for Equation (5.12):

$$(\hat{i}_1^\dagger \hat{j}_2), (\hat{i}_2^\dagger \hat{j}_3), (\hat{i}_3^\dagger \hat{j}_4), (\hat{j}_1 \hat{i}_4^\dagger) \to \mathbf{A}^{i_1,i_2}_{i_4} \mathbf{B}^{i_3}_{i_1,i_2} \mathbf{C}^{i_4}_{i_3} \to \mathbf{A}^{kl}_{n} \mathbf{B}^{m}_{kl} \mathbf{C}^{n}_{m} \tag{5.17a}$$

$$(\hat{i}_1^\dagger \hat{j}_3), (\hat{i}_2^\dagger \hat{j}_2), (\hat{i}_3^\dagger \hat{j}_4), (\hat{j}_1 \hat{i}_4^\dagger) \to \mathbf{A}^{i_1,i_2}_{i_4} \mathbf{B}^{i_3}_{i_2,i_1} \mathbf{C}^{i_4}_{i_3} \to \mathbf{A}^{kl}_{n} \mathbf{B}^{m}_{lk} \mathbf{C}^{n}_{m} \tag{5.17b}$$

$$(\hat{i}_1^\dagger \hat{j}_2), (\hat{i}_2^\dagger \hat{j}_4), (\hat{i}_3^\dagger \hat{j}_1), (\hat{j}_1 \hat{i}_3^\dagger) \to \mathbf{A}^{i_1,i_2}_{i_3} \mathbf{B}^{i_3}_{i_1,i_4} \mathbf{C}^{i_4}_{i_2} \to \mathbf{A}^{kl}_{m} \mathbf{B}^{m}_{kn} \mathbf{C}^{n}_{l} \tag{5.17c}$$

$$(\hat{i}_1^\dagger \hat{j}_4), (\hat{i}_2^\dagger \hat{j}_2), (\hat{i}_3^\dagger \hat{j}_1), (\hat{j}_1 \hat{i}_3^\dagger) \to \mathbf{A}^{i_1,i_2}_{i_3} \mathbf{B}^{i_3}_{i_2,i_4} \mathbf{C}^{i_4}_{i_1} \to \mathbf{A}^{kl}_{m} \mathbf{B}^{m}_{ln} \mathbf{C}^{n}_{k} \tag{5.17d}$$

$$(\hat{i}_1^\dagger \hat{j}_3), (\hat{i}_2^\dagger \hat{j}_4), (\hat{i}_3^\dagger \hat{j}_1), (\hat{j}_1 \hat{i}_2^\dagger) \to \mathbf{A}^{i_1,i_2}_{i_3} \mathbf{B}^{i_3}_{i_4,i_1} \mathbf{C}^{i_4}_{i_2} \to \mathbf{A}^{kl}_{m} \mathbf{B}^{m}_{nk} \mathbf{C}^{n}_{l} \tag{5.17e}$$

$$(\hat{i}_1^\dagger \hat{j}_4), (\hat{i}_2^\dagger \hat{j}_3), (\hat{i}_3^\dagger \hat{j}_1), (\hat{j}_1 \hat{i}_2^\dagger) \to \mathbf{A}^{i_1,i_2}_{i_3} \mathbf{B}^{i_3}_{i_4,i_2} \mathbf{C}^{i_4}_{i_1} \to \mathbf{A}^{kl}_{m} \mathbf{B}^{m}_{nl} \mathbf{C}^{n}_{k} \tag{5.17f}$$

It is important to reinforce that these are the only possible pairings that obey Wick's theorem. For example: $\mathbf{A}^{i_1,i_2}_{i_3} \mathbf{B}^{i_3}_{i_2,i_1} \mathbf{C}^{i_4}_{i_4}$ can never be a possible pairing because of $\mathbf{C}^{i_4}_{i_4}$ ; pairings can **never** occur within an operator.

### 5.1.2 Summary of Objective

Computing an expectation value of the form in Equations (5.3) and (5.9) requires summing over all labels of products of operators $\hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}}\cdots$. The amount of summation terms is determined by the rank $R$ of the operators. Consider the product $\hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}}$ where $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ have ranks of two ($R = 2$)

$$\hat{\mathbf{A}} = \hat{\mathbf{A}}_0 + \hat{\mathbf{A}}_1 + \hat{\mathbf{A}}^1 + \hat{\mathbf{A}}^1_1 + \hat{\mathbf{A}}_2 + \hat{\mathbf{A}}^2, \tag{5.18}$$

$$\hat{\mathbf{B}} = \hat{\mathbf{B}}_0 + \hat{\mathbf{B}}_1 + \hat{\mathbf{B}}^1 + \hat{\mathbf{B}}^1_1 + \hat{\mathbf{B}}_2 + \hat{\mathbf{B}}^2, \tag{5.19}$$

and $\hat{\mathbf{C}}$ has a maximum rank of one ($R = 1$)

$$\hat{\mathbf{C}} = \hat{\mathbf{C}}_0 + \hat{\mathbf{C}}_1 + \hat{\mathbf{C}}^1 + \hat{\mathbf{C}}^1_1. \tag{5.20}$$

To compute $\left\langle \sum\limits_{labels} \hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}} \right\rangle$ we have to consider all product terms

$$\hat{\mathbf{A}}_0\hat{\mathbf{B}}_0\hat{\mathbf{C}}_0, \hat{\mathbf{A}}_0\hat{\mathbf{B}}_0\hat{\mathbf{C}}_1, \cdots, \hat{\mathbf{A}}_1\hat{\mathbf{B}}^1\hat{\mathbf{C}}_0, \cdots, \hat{\mathbf{A}}^2\hat{\mathbf{B}}_2\hat{\mathbf{C}}^1_1, \cdots, \hat{\mathbf{A}}^2\hat{\mathbf{B}}^2\hat{\mathbf{C}}^1_1.$$

Some terms, such as $\hat{\mathbf{A}}^2\hat{\mathbf{B}}_2\hat{\mathbf{C}}^1_1$, may have multiple pairings $\hat{\mathbf{A}}^{kl}\hat{\mathbf{B}}_{km}\hat{\mathbf{C}}^m_l$, $\hat{\mathbf{A}}^{kl}\hat{\mathbf{B}}_{lm}\hat{\mathbf{C}}^m_k$. Other terms, such as $\hat{\mathbf{A}}_1\hat{\mathbf{B}}^1\hat{\mathbf{C}}_0$ may have only one pairing $\hat{\mathbf{A}}_k\hat{\mathbf{B}}^k\hat{\mathbf{C}}_0$. There are also terms, such as $\hat{\mathbf{A}}_0\hat{\mathbf{B}}_0\hat{\mathbf{C}}_1$, that cannot be paired and contribute 0 to the sum over labels.

The focus of the rest of this chapter is finding a method for determining all possible pairings and their prefactors (including the number of $f$ and $\bar{f}$) for each product term as seen in Equation (5.15). Of key importance is being able to distinguish between non-zero terms, (and their associated pairings) and zero contribution terms. In the next section I will explore how to distinguish between these terms and pairings.

## 5.2 General pairing forms

This section will compare and contrast two term forms. It will outline the general constraints which define non-zero contributions. These constraints form the logical foundation for the operation of `termfactory`.

I will now use the shorthand *valid* to refer to terms/pairings with non-zero contribution and *invalid* to refer to terms/pairings which contribute 0 to the summation over labels.

### 5.2.1 General form

In the general case where $\bar{f}_i$ and $f_i$ are complex numbers, then all pairings ($\hat{\mathbf{A}}^k\hat{\mathbf{B}}_k$ for example) arise from terms of the general form:

$$\hat{\mathbf{A}}_N^M\hat{\mathbf{B}}_N^M\hat{\mathbf{C}}_N^M\cdots\hat{\mathbf{Z}}_N^M. \tag{5.21}$$

Consider two example terms: $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^1\hat{\mathbf{C}}_1^1\hat{\mathbf{D}}^1$ and $\hat{\mathbf{A}}_3^1\hat{\mathbf{B}}_1^3$. The first term has three possible pairings:

$$\hat{\mathbf{A}}_k\hat{\mathbf{B}}_l^k\hat{\mathbf{C}}_m^l\hat{\mathbf{D}}^m,\qquad \hat{\mathbf{A}}_l\hat{\mathbf{B}}_m^k\hat{\mathbf{C}}_k^l\hat{\mathbf{D}}^m,\qquad \hat{\mathbf{A}}_m\hat{\mathbf{B}}_l^k\hat{\mathbf{C}}_k^l\hat{\mathbf{D}}^m,$$

and the second term has six possible pairings:

$$\begin{array}{ccc} \hat{\mathbf{A}}_{lmn}^k\hat{\mathbf{B}}_k^{lmn}, & \hat{\mathbf{A}}_{mln}^k\hat{\mathbf{B}}_k^{lmn}, & \hat{\mathbf{A}}_{mnl}^k\hat{\mathbf{B}}_k^{lmn}, \\ \hat{\mathbf{A}}_{lnm}^k\hat{\mathbf{B}}_k^{lmn}, & \hat{\mathbf{A}}_{nlm}^k\hat{\mathbf{B}}_k^{lmn}, & \hat{\mathbf{A}}_{nml}^k\hat{\mathbf{B}}_k^{lmn}. \end{array}$$

All of these pairings are *valid* in the general case, and need to be considered in the sum over labels.

### 5.2.2 Simplified $\bar{f}$ form

In the specific case where $\bar{f}_i = 0$ and $f_i = 1$ all *valid*
terms must be of the form

$$\hat{\mathbf{A}}_N\hat{\mathbf{B}}_N^M\cdots\hat{\mathbf{Z}}^M, \tag{5.22}$$

where the first operator $\hat{\mathbf{A}}$ cannot have any creation labels ($M = 0$) and the last operator $\hat{Z}$ cannot have any annihilation labels ($N = 0$). Additionally, upper labels can only pair with lower labels that proceed them, otherwise the pairing would result in a $\bar{f}$ and therefore be *invalid*.

Consider the previous examples $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^1\hat{\mathbf{C}}_1^1\hat{\mathbf{D}}^1$ and $\hat{\mathbf{A}}_3^1\hat{\mathbf{B}}_1^3$. For the general case where $\bar{f}_i \neq 0$ the first term had three pairings. But only one of those pairings is *valid* when $\bar{f}_i = 0$. The pairing $\hat{\mathbf{A}}_k\hat{\mathbf{B}}_l^k\hat{\mathbf{C}}_m^l\hat{\mathbf{D}}^m$ has a prefactor of $f_kf_lf_m$ and is therefore *valid*. Whereas $\hat{\mathbf{A}}_l\hat{\mathbf{B}}_m^k\hat{\mathbf{C}}_k^l\hat{\mathbf{D}}^m$ and $\hat{\mathbf{A}}_m\hat{\mathbf{B}}_l^k\hat{\mathbf{C}}_k^l\hat{\mathbf{D}}^m$ have $\bar{f}_k$ in their prefactors so they are *invalid*. For the second example $\hat{\mathbf{A}}_3^1\hat{\mathbf{B}}_1^3$ all six of the pairings are *invalid* because $\bar{f}_k = 0$.

## 5.3 Constraints

The simplified pairing form in 5.22 will be used to explain two general constraints. Additional constraints can exist depending on the definition of the overall Equation (5.1). A term or pairing must satisfy ALL given constraints to be considered *valid*.

### Balanced Operators

Applying Wick's theorem involves pairing all annihilation and creation operators, it follows then, that all *valid* terms and pairings must be *balanced*: having an equal number of creation and annihilation operators. As the $M$ and $N$ labels count those operators, we can represent this constraint in a simple fashion: for an equation of the form Equation (5.1) with some finite number of operators ($\lambda$)

$$\sum_{\ell=1}^{\lambda} M_\ell = \sum_{\ell=1}^{\lambda} N_\ell, \tag{5.23}$$

or expressed in a more concise format

$$M_{\text{tot}} = N_{\text{tot}}. \tag{5.24}$$

For example $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_0\hat{\mathbf{C}}^1$ satisfies this constraint, but $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1\hat{\mathbf{C}}^1$ does not.

### Normal Ordering

This constraint prevents the pairing of $\hat{i}^\dagger$ or $\hat{j}$ operators from the same $\hat{\mathbf{A}}_N^M$ operator. For a *valid* term $\hat{\mathbf{A}}_N\hat{\mathbf{B}}_N^M\hat{\mathbf{C}}_N^M\cdots\hat{\mathbf{Z}}^M$, each operator cannot pair with itself because all $\hat{\mathbf{A}}_N^M$ operators are normal ordered. There are three possible cases we should consider: a term where all pairings are *valid* a term where all pairings are *invalid* and the mixed case where a term has some *valid* pairings and some *invalid* pairings.

**All *valid* pairings: $\hat{\mathbf{A}}_1\hat{\mathbf{B}}^1$**

In the trivial case there is no pairing where $\hat{\mathbf{A}}$ or $\hat{\mathbf{B}}$ can pair with themselves; as they must pair with each other $\hat{\mathbf{A}}_k\hat{\mathbf{B}}^k$.

**All *invalid* pairings: $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^2$**

For all possible pairings $\hat{\mathbf{B}}$ MUST pair with itself: $\hat{\mathbf{A}}_k\hat{\mathbf{B}}_l^{kl}$, and $\hat{\mathbf{A}}_l\hat{\mathbf{B}}_k^{kl}$.

**Some *valid* pairings: $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^1\hat{\mathbf{C}}^1$**

There is at least one pairing where $\hat{\mathbf{B}}$ pairs with itself $\hat{\mathbf{A}}_l\hat{\mathbf{B}}_k^k\hat{\mathbf{C}}^l$, **but** there is also at least one pairing where no operators pair with themselves $\hat{\mathbf{A}}_k\hat{\mathbf{B}}_l^k\hat{\mathbf{C}}^l$.

The Normal Ordering constraint[6] requires that each *valid* term $\hat{\mathbf{A}}_N\hat{\mathbf{B}}_N^M\hat{\mathbf{C}}_N^M\cdots\hat{\mathbf{Z}}^M$ has at least one pairing where no operators pair with themselves. This can be expressed in two parts:

[6]In the simplified case where $\bar{f} = 0$.

1. If either $M = 0$ or $N = 0$ an operator cannot pair with itself and therefore meets the constraint.

2. Otherwise, if both inequalities

$$M \leq N_{\text{tot}} - N, \tag{5.25}$$
$$N \leq M_{\text{tot}} - M, \tag{5.26}$$

are satisfied, then the operator satisfies the constraint.

Recall the example $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^2$ term; the $\hat{\mathbf{B}}$ operator does not satisfy eq. (5.25), since $2 \leq 2-1$ is not true, and therefore the term is *invalid*. Whereas, for the other example term $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^1\hat{\mathbf{C}}^1$; each operator does meet the conditions: $\hat{\mathbf{A}}$ and $\hat{\mathbf{C}}$ trivially, and for $\hat{\mathbf{B}}$: eqs. (5.25) and (5.26) are both satisfied. Therefore $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^1\hat{\mathbf{C}}^1$ satisfies the Normal Ordering constraint[7].

Consider the prior example from Section 5.1.2, that is of the general form in Equation (5.21), which has a large number of terms (144). Choosing $\bar{f}_i = 0$ changes to the simplified form from eq. (5.22), and applying the Balanced and Normal Ordering constraints results in a reduction down to only 12 terms:

$$\begin{aligned}&\hat{\mathbf{A}}_0\hat{\mathbf{B}}_0\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_1\hat{\mathbf{B}}_0\hat{\mathbf{C}}^1 + \hat{\mathbf{A}}_2\hat{\mathbf{B}}_0\hat{\mathbf{C}}^2 + \hat{\mathbf{A}}_1\hat{\mathbf{B}}^1\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_2\hat{\mathbf{B}}^2\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_0\hat{\mathbf{B}}_1\hat{\mathbf{C}}^1 + \hat{\mathbf{A}}_0\hat{\mathbf{B}}_2\hat{\mathbf{C}}^2 \\ &+ \hat{\mathbf{A}}_2\hat{\mathbf{B}}^1\hat{\mathbf{C}}^1 + \hat{\mathbf{A}}_1\hat{\mathbf{B}}_1\hat{\mathbf{C}}^2 + \hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^1\hat{\mathbf{C}}^1 + \hat{\mathbf{A}}_2\hat{\mathbf{B}}_1^1\hat{\mathbf{C}}^2 + \hat{\mathbf{A}}_2\hat{\mathbf{B}}_2^2\hat{\mathbf{C}}^2.\end{aligned} \tag{5.27}$$

Note that even though each term in Equation (5.27) satisfies the constraints, not all pairings are *valid*. As previously described, the term $\hat{\mathbf{A}}_1\hat{\mathbf{B}}_1^1\hat{\mathbf{C}}^1$ has one *valid* pairing and one *invalid* pairing.

There is an additional simplification that occurs due to the permutational symmetry of operators $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \cdots$.

### 5.3.1 Permutation symmetry

As defined in Equation (5.4), each operator $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \cdots$ is symmetric under permutation of the upper labels with themselves, and permutations of the lower labels with themselves; Since we are summing over all labels, we can compute only one of these $M!$ upper or $N!$ lower permutations, the lexicographically sorted permutation, and then account for the other permutations by multiplying by an additional factor. This approach is of interest because it reduces the computational cost of evaluating Equation (5.1).

Consider the term $\hat{\mathbf{A}}_2\hat{\mathbf{B}}^2$, which has two pairings $\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{kl}$, $\hat{\mathbf{A}}_{lk}\hat{\mathbf{B}}^{kl}$. We use only one term ($\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{kl}$) to represent the contribution of both pairings, and modify its prefactor by $(N!)$. A similar process can be applied to $\hat{\mathbf{A}}^2\hat{\mathbf{B}}_2$ with the pairing $\hat{\mathbf{A}}^{kl}\hat{\mathbf{B}}_{kl}$ and a factor of $(M!)$. Since all pairings are between two operators we only need to account for this permutation once. Double counting of these factors is easily handled by scanning from left-to-right across the

[7]A more in-depth explanation is provided in Appendix C.1.

operators. For example

$$\frac{1}{2!}\frac{1}{2!}\hat{\mathbf{A}}^2\hat{\mathbf{B}}_2 \Rightarrow \frac{1}{2!}\frac{1}{2!}\sum_{kl}\hat{\mathbf{A}}^{kl}\hat{\mathbf{B}}_{kl} \Rightarrow \frac{2!}{2!}\frac{1}{2!}\sum_{kl}\hat{\mathbf{A}}^{kl}\hat{\mathbf{B}}_{kl}, = \frac{1}{2!}\sum_{kl}\hat{\mathbf{A}}^{kl}\hat{\mathbf{B}}_{kl}, \tag{5.28}$$

where we account for the $kl \to lk$ permutation when considering $\hat{\mathbf{A}}$ by adding a factor of 2!; therefore we do not consider the $kl$ labels when we move on to $\hat{\mathbf{B}}$. It should be clear that using the lexicographically sorted permutation does not change the expectation value.

$$\frac{1}{2!}\sum_{kl}\hat{\mathbf{A}}^{kl}\hat{\mathbf{B}}_{kl} = \frac{1}{2!}\frac{1}{2!}\sum_{kl}\hat{\mathbf{A}}^{kl}\hat{\mathbf{B}}_{kl} + \frac{1}{2!}\frac{1}{2!}\sum_{kl}\hat{\mathbf{A}}^{lk}\hat{\mathbf{B}}_{kl}. \tag{5.29}$$

A slightly bigger example; consider the terms in Equation (5.27) which have the following pairings (where I explicitly list the prefactors):

$$\begin{aligned}
&\hat{\mathbf{A}}_0\hat{\mathbf{B}}_0\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_k\hat{\mathbf{B}}^k\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_k\hat{\mathbf{B}}_0\hat{\mathbf{C}}^k + \hat{\mathbf{A}}_0\hat{\mathbf{B}}_k\hat{\mathbf{C}}^k + \hat{\mathbf{A}}_k\hat{\mathbf{B}}^k_l\hat{\mathbf{C}}^l \\
&+ \frac{1}{2!}\left(\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^k\hat{\mathbf{C}}^l + \hat{\mathbf{A}}_{lk}\hat{\mathbf{B}}^k\hat{\mathbf{C}}^l + \hat{\mathbf{A}}_k\hat{\mathbf{B}}_l\hat{\mathbf{C}}^{kl} + \hat{\mathbf{A}}_l\hat{\mathbf{B}}_k\hat{\mathbf{C}}^{kl}\right) \\
&+ \frac{1}{2!2!}\Bigg(\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}_0\hat{\mathbf{C}}^{kl} + \hat{\mathbf{A}}_{lk}\hat{\mathbf{B}}_0\hat{\mathbf{C}}^{kl} + \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{kl}\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_{lk}\hat{\mathbf{B}}^{kl}\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_0\hat{\mathbf{B}}_{kl}\hat{\mathbf{C}}^{kl} + \hat{\mathbf{A}}_0\hat{\mathbf{B}}_{lk}\hat{\mathbf{C}}^{kl} \\
&\qquad + \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^k_m\hat{\mathbf{C}}^{lm} + \hat{\mathbf{A}}_{km}\hat{\mathbf{B}}^k_l\hat{\mathbf{C}}^{lm} + \hat{\mathbf{A}}_{lk}\hat{\mathbf{B}}^k_m\hat{\mathbf{C}}^{lm} + \hat{\mathbf{A}}_{mk}\hat{\mathbf{B}}^k_l\hat{\mathbf{C}}^{lm}\Bigg) \\
&+ \frac{1}{2!2!2!2!}\left(\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{kl}_{mn}\hat{\mathbf{C}}^{mn} + \hat{\mathbf{A}}_{lk}\hat{\mathbf{B}}^{kl}_{mn}\hat{\mathbf{C}}^{mn} + \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{kl}_{nm}\hat{\mathbf{C}}^{mn} + \hat{\mathbf{A}}_{lk}\hat{\mathbf{B}}^{kl}_{nm}\hat{\mathbf{C}}^{mn}\right).
\end{aligned} \tag{5.30}$$

We can replace them with the shorter and simpler lexicographically sorted pairings

$$\begin{aligned}
&\hat{\mathbf{A}}_0\hat{\mathbf{B}}_0\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_k\hat{\mathbf{B}}^k\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_k\hat{\mathbf{B}}_0\hat{\mathbf{C}}^k + \hat{\mathbf{A}}_0\hat{\mathbf{B}}_k\hat{\mathbf{C}}^k + \hat{\mathbf{A}}_k\hat{\mathbf{B}}^k_l\hat{\mathbf{C}}^l + \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^k\hat{\mathbf{C}}^l + \hat{\mathbf{A}}_k\hat{\mathbf{B}}_l\hat{\mathbf{C}}^{kl} \\
&+ \frac{1}{2}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{kl}\hat{\mathbf{C}}_0 + \frac{1}{2}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}_0\hat{\mathbf{C}}^{kl} + \frac{1}{2}\hat{\mathbf{A}}_0\hat{\mathbf{B}}_{kl}\hat{\mathbf{C}}^{kl} + \hat{\mathbf{A}}_{km}\hat{\mathbf{B}}^k_l\hat{\mathbf{C}}^{lm} + \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^k_m\hat{\mathbf{C}}^{lm} + \frac{1}{4}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{kl}_{mn}\hat{\mathbf{C}}^{mn}.
\end{aligned} \tag{5.31}$$

The approach described for two operators $\hat{\mathbf{A}}, \hat{\mathbf{B}}$ generalizes to multiple operators as seen by the final term $\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{kl}_{mn}\hat{\mathbf{C}}^{mn}$ in Equation (5.31). To account for the $kl \to lk$ permutation on $\hat{\mathbf{A}}_{kl}$ we include the $N_1! = 2!$ factor. When looking at $\hat{\mathbf{B}}^{kl}_{mn}$ we see that the $kl$ has thus been accounted for, but the $mn \to nm$ permutation has not, and so adds a $N_2! = 2!$ factor. Finally the $mn$ permutation on $\hat{\mathbf{C}}^{mn}$ has already been accounted for and so no factor is added for this operator. The prefactor is modified accordingly $\frac{1}{2!2!2!2!} \to \frac{1}{4}$.

Going forward I will only list the unique lexicographically sorted pairings, which are symmetric by definition. Each of these pairings will have been obtained following this left-to-right approach.

## 5.4 Projection Operator

In this section, I introduce the projection operator $\hat{P}_N^M$ which is distinct from the operators we have been previously discussing. It has no electronic DoF and is defined

$$\hat{P}_N^M = \hat{i}_1^\dagger \hat{i}_2^\dagger \cdots \hat{i}_M^\dagger \, \hat{j}_1 \hat{j}_2 \cdots \hat{j}_N. \tag{5.32}$$

It has no prefactor or coefficients and is a collection of creation and annihilation operators. The projection operator is applied to Equation (5.9) giving

$$\mathbf{R}_N^M = \left\langle \hat{P}_N^M \sum_{\text{all labels}} \hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}} \cdots \right\rangle. \tag{5.33}$$

In the simplest case, (which is effectively the general form in Equation (5.9)):

$$\mathbf{R}_0 = \left\langle \hat{P}_0 \sum_{\text{all labels}} \hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}} \cdots \right\rangle, \tag{5.34}$$

where we can see that all terms/pairings that we have been considering up till now are of the $\mathbf{R}_0$ form.

If we consider two operators ($\hat{\mathcal{A}}$, $\hat{\mathcal{B}}$) both with maximum rank of two:

$$\mathbf{R}_0 = \left\langle \hat{P}_0 \sum_{\text{all labels}} \hat{\mathcal{A}}\hat{\mathcal{B}} \right\rangle, \tag{5.35}$$

and choose $\bar{f}_i = 0$, as well as applying both constraints

$$\mathbf{R}_0 = \hat{\mathbf{A}}_0 \hat{\mathbf{B}}_0 + \hat{\mathbf{A}}_1 \hat{\mathbf{B}}^1 + \hat{\mathbf{A}}_2 \hat{\mathbf{B}}^2, \tag{5.36}$$

with the following *valid* pairings, where I colour all labels that will be traced out in blue

$$\mathbf{R}_0 = \hat{\mathbf{A}}_0 \hat{\mathbf{B}}_0 + \hat{\mathbf{A}}_k \hat{\mathbf{B}}^k + \hat{\mathbf{A}}_{kl} \hat{\mathbf{B}}^{kl}. \tag{5.37}$$

The use for this colouring becomes evident when we consider $\hat{P}_1$ (choosing $\bar{f}_i = 0$ and applying both constraints):

$$\mathbf{R}_1 = \left\langle \hat{P}_1 \sum_{\text{all labels}} \hat{\mathcal{A}}\hat{\mathcal{B}} \right\rangle, \tag{5.38}$$

with *valid* terms:

$$\mathbf{R}_1 = \hat{\mathbf{A}}_0 \hat{\mathbf{B}}^1 + \hat{\mathbf{A}}^1 \hat{\mathbf{B}}_0 + \hat{\mathbf{A}}_1^1 \hat{\mathbf{B}}^1 + \hat{\mathbf{A}}_1 \hat{\mathbf{B}}^2, \tag{5.39}$$

and the following *valid* pairings, <u>where external labels are coloured red</u>

$$\mathbf{R}_z = \hat{\mathbf{A}}_0 \hat{\mathbf{B}}^z + \hat{\mathbf{A}}^z \hat{\mathbf{B}}_0 + \hat{\mathbf{A}}_k^z \hat{\mathbf{B}}^k + \hat{\mathbf{A}}_k \hat{\mathbf{B}}^{zk} + \hat{\mathbf{A}}_k \hat{\mathbf{B}}^{kz}. \tag{5.40}$$

The label $z$ is not part of the sum in Equation (5.38) and so we call it an "external" label. The label $k$ is part of the sum and so we call it an "internal" label. The colouring is a visual aid to help distinguish between labels, as it can be quite tedious to do so otherwise. In addition I label the external labels using letters from the alphabet in reverse $(z, y, x, w, v, u, \cdots)$. [8]

Each equation's $\mathbf{R}_0, \mathbf{R}_1, \cdots$, *valid* pairings are unique, and in lexicographical order.

### 5.4.1 External Symmetrization

To compute $\mathbf{R}_0$ or $\mathbf{R}_z$ the previous left-to-right approach is sufficient. The only difference regarding $\mathbf{R}_z$ is that instead of calculating a single complex value we are calculating a vector of complex values of length $N_{\text{modes}}$ (the number of normal modes); $\hat{\mathbf{A}}_0\hat{\mathbf{B}}^z + \hat{\mathbf{A}}^z\hat{\mathbf{B}}_0$ represents an element wise addition along the $z$-labeled normal mode dimension.

Higher order terms $\mathbf{R}_{zy}, \mathbf{R}_{zyw}, \cdots$ will be symmetric if all terms in Wick's theorem are considered. However it is more efficient to only include the unique terms in lexicographical order, and symmetrize the result during the computational process. Similar to Section 5.3.1 we would like to avoid computing all permutations of external labels $(\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{klzy} + \hat{\mathbf{A}}_k\hat{\mathbf{B}}^{kyz})$ but the contributions are *not* the same in general and so we cannot simply account for these permutations using only a multiplicative factor. Instead, we first compute lexicographically ordered tensors (such as $\mathbf{R}_{zy}$) which are *not* symmetric[9], and then we numerically symmetrize them.

The symmetrized $\tilde{\mathbf{R}}_{zy}$ tensor can be computed by summing over all permutations and then normalizing:

$$\tilde{\mathbf{R}}_{zy} = \frac{1}{2}(\mathbf{R}_{zy} + \mathbf{R}_{yz}), \tag{5.41}$$

however in practice this can add a considerable computational cost. To accelerate this symmetrization process we employ a sparse matrix product approach.

$$\tilde{\mathbf{R}}_{zy} = \hat{S}(\mathbf{R}_{zy}). \tag{5.42}$$

Consider a tensor $\mathbf{R}_N$ with explicit labels $R^a_{b,j_1,j_2,\cdots,j_{N_{\text{modes}}}}$. We can define full labels $J \to j_1, j_2, \cdots, j_{N_{\text{modes}}}$ and symmetrized labels $\tilde{J} \to j_1 \leq j_2 \leq \cdots \leq j_{N_{\text{modes}}}$, and then define the matrix $S_{\tilde{J},J}$ whose elements are 1 where the index $J$ can be sorted to index $\tilde{J}$ and zeros otherwise. The matrix $S_{J,\tilde{J}}$ can be defined in a similar fashion, mapping indices $\tilde{J}$ back to $J$ Symmetrization is a two step process:

$$\mathbf{R}_{\tilde{J}} = \sum_J S_{\tilde{J},J}\,\mathbf{R}_J, \tag{5.43}$$

[8] So far this has been sufficient to uniquely label pairings. Although a different approach may be needed in the future.

[9] With respect to the external labels $zy$; they are symmetric with respect to the internal labels by definition.

and then

$$\tilde{\mathbf{R}}_J = \sum_{\tilde{J}} S_{J,\tilde{J}} \mathbf{R}_{\tilde{J}}. \tag{5.44}$$

I provide a small example: $\mathbf{R}_3$ has an explicit labelling $R^a_{b,j_1,j_2,j_3}$. For a specific index $\tilde{J}$ of $1, 2, 3$ there are $3!$ possible $J$ indices: $123, 132, 213, 231, 312, 321$ which can be sorted to $\tilde{J}$. Thus the matrix elements $S_{1,2,3,1,2,3}$, $S_{1,2,3,1,3,2}$, $S_{1,2,3,2,1,3}$, $S_{1,2,3,2,3,1}$, $S_{1,2,3,3,1,2}$, and $S_{1,2,3,3,2,1}$ are all set to 1. Then we compute $\mathbf{R}_{\tilde{J}=(1,2,3)}$ as follows

$$\tilde{\mathbf{R}}_{(1,2,3)} = \sum_J S_{(1,2,3),J} \mathbf{R}_J = \mathbf{R}_{(1,2,3)} + \mathbf{R}_{(1,3,2)} + \mathbf{R}_{(2,1,3)} + \mathbf{R}_{(2,3,1)} + \mathbf{R}_{(3,1,2)} + \mathbf{R}_{(3,2,1)} \tag{5.45}$$

Thus $S_{\tilde{J},J}$ maps from the $3!$ $J$ indices to the single $\tilde{J}$ index. Similarly $S_{J,\tilde{J}}$ will map from the single $\tilde{J}$ index to the $3!$ $J$ indices.

Two matrix products as opposed to summing over $N!$ permutations is computationally attractive. These symmetrization matrices only need to be computed once for a fixed number of indices. They can also be pre-calculated for a large range of indices and stored to disk for repeated use at a later date. This process works the same if the matrices $S_{\tilde{J},J}$, and $S_{J,\tilde{J}}$ are dense or sparse. But for performance reasons we have implemented them as sparse matrices. In practice this means the cost of calculating them is negligible and only the sparse matrix product need be considered when measuring computational performance. in Section 6.2.2 I provide some performance analysis on the advantage of using this sparse matrix symmetrization.

I introduced a projection operator $\hat{P}^M_N$ which provides the ability to generate various orders of tensors: $\mathbf{R}_0, \mathbf{R}_1, \mathbf{R}^1, \cdots$. Labels are coloured either blue, to indicate they will be traced out, or red to indicate that they are external labels and therefore are indexed to the tensor $\mathbf{R}^M_N$. Higher order tensors $\mathbf{R}_2, \mathbf{R}^2, \cdots$ are symmetrized using a sparse matrix projection approach.

## 5.5 Solving the Problem in practice

I have laid out the theoretical foundation for computing $\mathbf{R}^M_N$ equations. A detailed summary of the general contraction scheme is provided in Appendix C.2.

Now I turn to addressing the problem of determining these terms and pairings in practice. in Section 5.6 I will discuss in broad strokes how `termfactory` tackles the problem of determining these pairings. I have provided two substantial examples of output generated by `termfactory` in Appendices E.1 and E.2

While developing both `termfactory` and `t-amplitudes` it became apparent that having a more generalized or “formal” approach to distinguishing/generating *valid* terms and pairings would be useful. It seemed intuitive that one should be able to express both terms and pairings as a formal language, and consequently be able to leverage the work of the field. My exploration of this approach is presented in Section 5.7.

## 5.6 `termfactory` scheme

Having covered the theoretical building blocks describing terms and their pairings, I will give a broad overview of how they are generated by `termfactory`. I will first explain the specific representations that are used. Then I will discuss how the output of `termfactory` is contextually dependent on the theoretical definition of the equations $\mathbf{R}_0, \mathbf{R}_1, \cdots$. Finally, I can outline the general protocol taken by `termfactory` to produce terms and their pairings.

### 5.6.1 Term and pairing representation

**Tuples** It is very natural to represent operators $\hat{\mathbf{A}}_N^M$ using tuples `(m,n)` of length two; where the first element is the $M$ label, and the second element is the $N$ label. A term with multiple operators such as: $\hat{\mathbf{A}}_1^0\hat{\mathbf{B}}_0^1$, can be represented by multiple tuples: `(0,1)`, `(1,0)`. As one of the goals of `termfactory` is robustness, the immutability of tuples is advantageous compared to other data types. Listing 5.1 provides an example of how tuples can be used to encode the constraints defined in Section 5.3. Specifically, the constraints defined in Equations (5.25) and (5.26) are handled on line 14 and 15.

```
 1 def valid_term(term):
 2     """ Return True if term is valid, otherwise False """
 3     A, B = term
 4 
 5     M_tot = A[0] + B[0]
 6     N_tot = A[1] + B[1]
 7 
 8     # balanced operators
 9     If not (M_tot == N_tot):
10         return False
11 
12     # normal ordering
13     for Op in [A, B]:
14         if not (Op[0] == 0 or Op[1] == 0):
15             if not (Op[0] <= N_tot - Op[1]) and  (Op[1] <= M_tot - Op[0]):
16                 return False
17 
18     return True
```

Listing 5.1: Illustrative use of tuples for checking constraints of terms $\hat{\mathbf{A}}_N^M\hat{\mathbf{B}}_N^M$

However, embedding the constraints and logic of pairing generation using *just* tuples leads to code which is harder to read, debug, and work with. I speak from personal experience as the early prototypes of `termfactory` suffered from these issues. Instead `termfactory` now uses namedtuples.

**Namedtuples** Namedtuples are a subclass of tuples available in the Python standard library. They are used extensively in `termfactory` for three primary reasons:

1. They allow for writing code which encodes logic in a English-like manner.

2. Their fields can be explicitly named as to their purpose.

3. They are tuples and therefore immutable.

Consider the term $\hat{\mathbf{A}}^1_0$ with a tuple representation of `(1,0)`. The corresponding simple namedtuple representation is `A = Aop(rank=1, m=1, n=0)`; although we will use more complicated namedtuples later on. Elements of the namedtuple can be accessed using their fieldnames, rather than their index, like so: `A.rank`, `A.m`, and `A.n`.

I treat each operator $\hat{A}, \hat{B}, \cdots$ in this fashion:

$\hat{\mathbf{A}}^2_1$: `A = Aop(rank=3, m=2, n=1)` where `A.m` $= 2$ and `A.n` $= 1$

$\hat{\mathbf{B}}^3_4$: `B = Bop(rank=7, m=3, n=4)` where `B.m` $= 3$ and `B.n` $= 4$

Terms can be represented as lists of namedtuples:

$$\hat{\mathbf{A}}^1_0\hat{\mathbf{B}}^0_1 \rightarrow \texttt{[Aop(rank=1, m=1, n=0), Bop(rank=1, m=0, n=1)]}.$$

A clear advantage of using namedtuples is accessing their fields by name instead of by index. An example of this is provided in Listing 5.2, which can be compared to Listing 5.1

```
def valid_term(term):
    """ Return True if term is valid, otherwise False """
    A, B = term

    M_tot = A.m + B.m
    N_tot = A.n + B.n

    # balanced operators
    If not (M_tot == N_tot):
        return False

    # normal ordering
    for Op in [A, B]:
        if not (Op.m == 0 or Op.n == 0):
            if not (Op.m <= N_tot - Op.n) and  (Op.n <= M_tot - Op.m):
                return False

    return True
```

Listing 5.2: Illustrative use of namedtuples for checking constraints of terms $\hat{\mathbf{A}}^M_N\hat{\mathbf{B}}^M_N$

**Pairings** For pairings, more complicated namedtuples are used to fully specify the pairings. A fully specified namedtuple contains all information needed to write the formal representation down by hand. Consider the term

$$\hat{P}_1 \qquad \hat{\mathbf{A}}^1\hat{\mathbf{B}}^1_1\hat{\mathbf{C}}^1_1,$$

with the *valid* pairing

$$\hat{P}_z \qquad \hat{\mathbf{A}}^z\hat{\mathbf{B}}^k_l\hat{\mathbf{C}}^l_k.$$

The associated fully specified namedtuples are

```
Aop(rank=1, m=1, n=0, p_m=1, p_n=0, b_m=0, b_n=0, c_m=0, c_n=0)
Bop(rank=2, m=1, n=1, p_m=0, p_n=0, a_m=0, a_n=0, c_m=1, c_n=1)
Cop(rank=2, m=1, n=1, p_m=0, p_n=0, a_m=0, a_n=0, b_m=1, b_n=1)
```

Previously, for terms the namedtuple's fields could be directly mapped from their formal definition $\hat{\mathbf{B}}^1_1 \to$ `Bop(rank=2, m=1, n=1)`. But each pairing occurs between **two** different operators, and therefore we need additional information (`p_m=0, p_n=0, a_m=0, a_n=0, c_m=1, c_n=1`), to specify which operators those pairings are being formed with and how many pairings are being formed.

For each other operator $\hat{P}, \hat{A}, \hat{C}$, we add an upper `p_m, a_m, c_m`, and lower `p_n, a_n, c_n` field. An operator's upper field `*_m` denotes how many of its own upper labels (counted by `m`) are pairing with the lower labels of some other operator, and the same is true for the lower fields counting how many lower labels pair with other operator's upper labels. In the pairing $\hat{P}_z\hat{\mathbf{A}}^z$, $\hat{\mathbf{A}}$ contributes an upper label and therefore `p_m=1` for $\hat{\mathbf{A}}$'s namedtuple.

The other *valid* pairings would be specified as follows:

$$\hat{P}_z \qquad \hat{\mathbf{A}}^k\hat{\mathbf{B}}^z_l\hat{\mathbf{C}}^l_k$$

```
Aop(rank=1, m=1, n=0, p_m=0, p_n=0, b_m=0, b_n=0, c_m=1, c_n=0)
Bop(rank=2, m=1, n=1, p_m=1, p_n=0, a_m=0, a_n=0, c_m=0, c_n=1)
Cop(rank=2, m=1, n=1, p_m=0, p_n=0, a_m=0, a_n=1, b_m=1, b_n=0)
```

$$\hat{P}_z \qquad \hat{\mathbf{A}}^k\hat{\mathbf{B}}^l_k\hat{\mathbf{C}}^z_l$$

```
Aop(rank=1, m=1, n=0, p_m=0, p_n=0, b_m=1, b_n=0, c_m=0, c_n=0)
Bop(rank=2, m=1, n=1, p_m=0, p_n=0, a_m=0, a_n=1, c_m=1, c_n=0)
Cop(rank=2, m=1, n=1, p_m=1, p_n=0, a_m=0, a_n=0, b_m=0, b_n=1)
```

It is important to highlight that these namedtuples store the number of pairings between every operator, **not** the specific labels of the pairings $klm\cdots$, $zxy\cdots$. This representation is possible because of the label symmetry in the definition of the operators (Equation (5.1)) which arises from the fact that these equations are describing Bosons and not Fermions. For Fermions additional complications would arise due to the occurrence of signs in the expressions.

### 5.6.2 Context

Before explaining the general `termfactory` approach I introduce the shorthand *context*, which means all information necessary to fully and explicitly define a $\mathbf{R}^M_N$ equation, as well as all possible *valid* terms and pairings. It represents all require input information and baked in assumptions necessary for `termfactory`.

The notions of *valid* or *invalid* terms and pairings are only well defined withing a given *context*. Take the example of $\hat{\mathbf{A}}_3^1\hat{\mathbf{B}}_1^3$ from Section 5.2.2. In the general case where $\bar{f} \neq 0$ this term is *valid* but when $\bar{f} = 0$ this term is *invalid*. The choice of $\bar{f}$ as being 0 or some complex valued number is necessary information for distinguishing between *valid* and *invalid* terms and pairings, and is therefore part of the *context* defining a specific $\mathbf{R}_N^M$ equation which is then processed by `termfactory`.

Recall that $\mathbf{R}_N^M$ equation's are defined in Equation (5.33) as

$$\mathbf{R}_N^M = \left\langle \hat{P}_N^M \sum_{\text{all labels}} \hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}}\cdots \right\rangle .$$

The form of $\mathbf{R}_N^M$ depends on the specific operators $\hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}}\cdots$ and their definitions.

To talk about the process of determining *valid* terms and pairings in practice we must agree on a *context* under which to evaluate them. The *context* is comprised of information whose corresponding logic is "baked" into `termfactory`:

- The specific operators and their definitions that comprise the equation we are trying to compute.

and information that can be easily changed, which comprises the input variables:

- The value of $\bar{f}$; and therefore if the general form (5.21) or the simplified form (5.22) is being used.
- The maximum rank of each operator.

For example we can use `termfactory` like so:

```
python termfactory -t 1 2 1
```

where the argument `-t` is used to indicate a sequence of integers will follow; integers which are the truncation values for $\hat{P}$, $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ of

$$\mathbf{R}_N^M = \left\langle \hat{P}_N^M \sum_{\text{all labels}} \hat{\mathcal{A}}\hat{\mathcal{B}} \right\rangle .$$

To remove the terms with $\bar{f}$ prefactors (effectively setting $\bar{f} = 0$) we add the flag `-rf`:

```
python termfactory -t 1 2 1 -rf
```

I will give an example: For $\hat{A}, \hat{B}$ both with maximum ranks 2 and $\bar{f} = 0$ there are four *valid* pairings for $\mathbf{R}_1$

$$\mathbf{R}_z = \hat{\mathbf{A}}_0\hat{\mathbf{B}}^z + \hat{\mathbf{A}}^z\hat{\mathbf{B}}_0 + \hat{\mathbf{A}}_i^z\hat{\mathbf{B}}^i + \hat{\mathbf{A}}_i\hat{\mathbf{B}}^{iz}, \tag{5.46}$$

but if instead $\hat{B}$'s maximum rank is reduced to 1 then the $\hat{\mathbf{A}}_1\hat{\mathbf{B}}^2$ term does not exist and therefore the $\hat{\mathbf{A}}_i\hat{\mathbf{B}}^{iz}$ pairing is gone from $\mathbf{R}_1$

$$\mathbf{R}_z = \hat{\mathbf{A}}_0\hat{\mathbf{B}}^z + \hat{\mathbf{A}}^z\hat{\mathbf{B}}_0 + \hat{\mathbf{A}}_i^z\hat{\mathbf{B}}^i . \tag{5.47}$$

If we introduce a new operator $\hat{C}$ which has a maximum rank of 1:

$$\mathbf{R}_{z} = \hat{\mathbf{A}}_0\hat{\mathbf{B}}^{z}\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}_0\hat{\mathbf{B}}_0\hat{\mathbf{C}}^{z} + \hat{\mathbf{A}}^{z}\hat{\mathbf{B}}_0\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}^{z}_{i}\hat{\mathbf{B}}^{i}\hat{\mathbf{C}}_0 + \hat{\mathbf{A}}^{z}_{i}\hat{\mathbf{B}}_0\hat{\mathbf{C}}^{i} + \hat{\mathbf{A}}^{z}\hat{\mathbf{B}}_{i}\hat{\mathbf{C}}^{i} + \hat{\mathbf{A}}_{i}\hat{\mathbf{B}}^{i}\hat{\mathbf{C}}^{z} + \hat{\mathbf{A}}_{i}\hat{\mathbf{B}}^{z}\hat{\mathbf{C}}^{i}. \tag{5.48}$$

The *context* is therefore the input specification for `termfactory` which determines for which $\mathbf{R}^{M}_{N}$ equations, the terms and pairings will be produced.

### 5.6.3 Overall scheme

Given a specific *context*, the general scheme is as follows:

1. For each operator $\hat{\mathbf{A}} = \sum \hat{\mathbf{A}}_0 + \hat{\mathbf{A}}_1 + \hat{\mathbf{A}}^1 + \hat{\mathbf{A}}^1_1 + \cdots$ create a representative list of tuples of each term's $M$, $N$ labels: `[(0,0), (0,1), (1,0), (1,1), ...]`.

2. The Balanced Operators and Normal Ordering constraints (from Section 5.3) are applied to all permutations of the product of these lists. This produces a list of all *valid* terms, where each of these terms are represented by lists of namedtuples which fully specify their $M$, $N$ labels. More detail will be provided in Section 5.6.4.

3. Remove all duplicates from this list; we only need unique lexicographically ordered terms.

4. For each *valid*
term, generate all possible pairings and apply the Normal Ordering constraint to remove *invalid*
pairings. Each remaining term in the list is a *valid*
pairing, fully specified by a namedtuple. More detail will be provided in Section 5.6.5.

5. Finally, various processing routines can be applied to generate either a LaTeX (`*.tex`) or Python (`*.py`) file.

LaTeX To generate the LaTeX for $\hat{\mathbf{A}}^{z}_{k}\hat{\mathbf{B}}^{yk}$: various functions analyze the namedtuples and produce the string `\hat {\textbf {A}}^{\red {z}}_{\blue {k}}\hat {\textbf {B}}^{\red {y}\blue {k}}`. By repeating this process for every pairing and having a pre-built header and footer `termfactory` can produce a multi-page LaTeX file. A relatively simple example of the output is provided in Appendix E.1 and a more extreme example is provided in Appendix E.2.

Python Similarly for generating Python code, various functions can process the namedtuples and produce Python code to compute a single pairing $\hat{A}^{z}_{k}\hat{B}^{yk}$: `np.einsum('zk,yk -> zy', A[(1,1)], B[(2,0)])`. These can then be "glued" together to form blocks of code which calculate $\mathbf{R}^{M}_{N}$. By using similar pre-build header and footers a full Python module is created to calculate various $\mathbf{R}_0, \mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3, \cdots$. This module is then imported and utilized to compute EOM as further described in Chapter 6.

### Small example

I present a small example of a few terms used in calculating $R^{a}_{zy}$:

$$R_{zy} \rightarrow \hat{A}_k\hat{B}^z\hat{C}^{ky} + \hat{A}_k\hat{B}^{zk}\hat{C}^{y} + \hat{A}_k\hat{A}_l\hat{B}^{lz}\hat{C}^{ky} + \frac{1}{2}\left(\hat{A}_k\hat{B}^k\hat{C}^{zy} + \hat{A}_k\hat{B}^{zy}\hat{C}^{k}\right) + \frac{1}{4}\left(\hat{A}_k\hat{A}_l\hat{B}^{zy}\hat{C}^{kl} + \hat{A}_k\hat{A}_l\hat{B}^{ij}\hat{C}^{zy}\right) \tag{5.49}$$

and the corresponding code excerpt in Listing 5.3

```
R += np.einsum('k,z,ky->zy', A[(0, 1)], B[(1, 0)], C[(2, 0)])
R += np.einsum('k,kz,y->zy', A[(0, 1)], B[(2, 0)], C[(1, 0)])
R += np.einsum('k,l,lz,ky->zy', A[(0, 1)], A[(0, 1)], B[(2, 0)], C[(2, 0)])
R += (1 / 2) * (
    np.einsum('k,k,zy->zy', A[(0, 1)], B[(1, 0)], C[(2, 0)]) +
    np.einsum('k,zy,k->zy', A[(0, 1)], B[(2, 0)], C[(1, 0)])
)
R += (1 / 4) * (
    np.einsum('k,l,zy,kl->zy', A[(0, 1)], A[(0, 1)], B[(2, 0)], C[(2, 0)]) +
    np.einsum('k,l,kl,zy->zy', A[(0, 1)], A[(0, 1)], B[(2, 0)], C[(2, 0)])
)
```

Listing 5.3: Excerpt of Python code from a function which generates $R_{zy}$

## 5.6.4 Basic Exclusion (Step 2/5)

Here I will describe in more detail the process by which I apply the constraints and produce all permutations of the products of each operators expansion. We revisit the prior example from Section 5.1.2, but choose $\bar{f}_i = 0$. There are three operators $\hat{A}$, $\hat{B}$, $\hat{C}$, where $\hat{A}$, and $\hat{B}$ have maximum ranks of two, and $\hat{C}$ has a maximum rank of one:

$\hat{A}$: `[(0,0), (0,1), (0,2)]`

$\hat{B}$: `[(0,0), (0,1), (1,0), (1,1), (0,2), (2,0)]`

$\hat{C}$: `[(0,0), (0,1), (1,0)]`

This will produce $3 \times 6 \times 3 = 54$ possible triplets, such as `[(0,1), (1,1), (1,0)]`. Some of these terms are *invalid*
and therefore we want to **exclude** them from our collection of terms, so as to be left with only *valid*
terms.

In general, I begin by looping over each projection term in $\hat{P}^M_N$; equivalent to processing each equation $\mathbf{R}_0, \mathbf{R}_1, \mathbf{R}_2$, one-by-one. For a given equation I loop over each term and apply the Balanced and Normal Ordering constraints in a manner similar to Listing 5.2. Any term that does not meet these basic constraints is excluded immediately. For $\mathbf{R}_0$ this results in only twelve terms as previous seen in Equation (5.27). The associated list of simple namedtuples is illustrated in Listing 5.4:

```
[
    [A(rank=0, m=0, n=0), B(rank=0, m=0, n=0), C(rank=0, m=0, n=0)],
    [A(rank=1, m=0, n=1), B(rank=0, m=0, n=0), C(rank=1, m=1, n=0)],
    [A(rank=2, m=0, n=2), B(rank=0, m=0, n=0), C(rank=2, m=2, n=0)],
    [A(rank=1, m=0, n=1), B(rank=1, m=1, n=0), C(rank=0, m=0, n=0)],
    [A(rank=2, m=0, n=2), B(rank=2, m=2, n=0), C(rank=0, m=0, n=0)],
    [A(rank=0, m=0, n=0), B(rank=1, m=0, n=1), C(rank=1, m=1, n=0)],
    [A(rank=0, m=0, n=0), B(rank=2, m=0, n=2), C(rank=2, m=2, n=0)],
    [A(rank=2, m=0, n=2), B(rank=1, m=1, n=0), C(rank=1, m=1, n=0)],
    [A(rank=1, m=0, n=1), B(rank=1, m=0, n=1), C(rank=2, m=2, n=0)],
    [A(rank=1, m=0, n=1), B(rank=1, m=1, n=1), C(rank=1, m=1, n=0)],
    [A(rank=2, m=0, n=2), B(rank=1, m=1, n=1), C(rank=2, m=2, n=0)],
    [A(rank=2, m=0, n=2), B(rank=2, m=2, n=2), C(rank=2, m=2, n=0)]
]
```

Listing 5.4: Simple namedtuple representation of *valid*terms for $\mathbf{R}_0$

After filtering out *invalid*
terms using the constraints, I am left with a list of *valid*
permutations. Every remaining product term $\hat{\mathbf{A}}_N^M \hat{\mathbf{B}}_N^M \hat{\mathbf{C}}_N^M$ obeys both the Balanced Operators and Normal Ordering constraints. At this point there is no notion of pairing. If all possible terms are *invalid*
for a projector $\hat{P}_N$ then we skip to the next one $\hat{P}_{N+1}$ and so on. Terms for $\mathbf{R}_1$ and $\mathbf{R}_2$ are treated in the same fashion as was described for $\mathbf{R}_0$.

### 5.6.5 Generating all valid pairings (Step 4/5)

In this step we explicitly define all possible *valid*
pairings. We begin with a list of unique lexicographically ordered terms in simplified namedtuple form.

In general, I begin by looping over each projection term in $\hat{P}_N^M$; similar to Step 2 in the previous section. For a given equation, say $\mathbf{R}_3$, I loop over each term in the list

#### Illustrative Example

The first step in generating the full specified namedtuple representations is to enumerate over all permutations of pairings. Consider the term $\hat{P}_3 \quad \hat{\mathbf{A}}_2 \hat{\mathbf{B}}^3 \hat{\mathbf{C}}^3$.

The valid normal-ordered pairings are

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_{kl} \hat{\mathbf{B}}^{zyx} \hat{\mathbf{C}}^{kl}$$

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_{kl} \hat{\mathbf{B}}^{zyk} \hat{\mathbf{C}}^{xl}$$

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_{kl} \hat{\mathbf{B}}^{zkl} \hat{\mathbf{C}}^{yx}$$

I enumerate over all internal and external pairing permutations, for all but one of the operators; the final operator's pairings are pre-determined by the enumeration over all possible pairings, for each of the previous operators. In this example, I enumerate over $\hat{\mathbf{A}}$, and $\hat{\mathbf{B}}$, but not $\hat{\mathbf{C}}$. The pairing with $\hat{P}$ is determined as a byproduct of processing $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$, and $\hat{\mathbf{C}}$. In general a left-to-right approach, similar to Section 5.3.1, can be used to narrow down all possible pairings[10].

Choosing $\hat{\mathbf{A}}$ to "pivot" around, I represent all possible/partial permutations as tuples `(p, a, b, c)` whose numbers represent pairings with $\hat{P}$, $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$ respectively[11]. We consider possible pairings using only the upper labels first, which is trivially `(0,0,0,0)` as $\hat{\mathbf{A}}$ has no upper labels. Considering the possible pairings using the lower labels:

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_2\hat{\mathbf{B}}^3\hat{\mathbf{C}}^2 : \texttt{(0, 0, 2, 0)} \rightarrow \quad \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{1,kl}\hat{\mathbf{C}}^2$$

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_2\hat{\mathbf{B}}^3\hat{\mathbf{C}}^2 : \texttt{(0, 0, 1, 1)} \rightarrow \quad \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{2,k}\hat{\mathbf{C}}^{1,l}$$

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_2\hat{\mathbf{B}}^3\hat{\mathbf{C}}^2 : \texttt{(0, 0, 0, 2)} \rightarrow \quad \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{3}\hat{\mathbf{C}}^{kl}$$

In the same fashion I choose $\hat{\mathbf{B}}$ as a "pivot" point, with `(p, a, b, c)`, whose upper pairings are

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_2\hat{\mathbf{B}}^3\hat{\mathbf{C}}^2 : \texttt{(3, 0, 0, 0)} \rightarrow \quad \hat{\mathbf{A}}_{2}\hat{\mathbf{B}}^{zyx}\hat{\mathbf{C}}^2$$

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_2\hat{\mathbf{B}}^3\hat{\mathbf{C}}^2 : \texttt{(2, 1, 0, 0)} \rightarrow \quad \hat{\mathbf{A}}_{1,k}\hat{\mathbf{B}}^{zyk}\hat{\mathbf{C}}^2$$

$$\hat{P}_{zyx} \quad \hat{\mathbf{A}}_2\hat{\mathbf{B}}^3\hat{\mathbf{C}}^2 : \texttt{(1, 2, 0, 0)} \rightarrow \quad \hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{zkl}\hat{\mathbf{C}}^2$$

The lower pairings for $\hat{\mathbf{B}}$ are trivially `(0,0,0,0)`. We have now enumerated over all possible upper and lower pairings for $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$. At this point we stop since all *valid*
pairings with $\hat{\mathbf{C}}$ can be determined with the possible pairings we just considered in $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$.

### Enumerate all combinations

The rest of the process is as follows:

1. We form a list `combined_m_perms` of the product of all possible upper permutations for each operator we "pivoted" around. In this case, there are only three upper permutations because $\hat{\mathbf{A}}$ has no upper labels. Then we remove all elements which are not possible (without considering the lower labels).

2. We do the same for the lower labels: form a list `combined_n_perms` of the product of all possible lower permutations and remove all elements which are not possible (without considering the upper labels).

[10]For specific equations $\mathbf{R}_N^M$ with particular operators $\hat{A}, \hat{B}, \cdots$ different strategies can be employed, but the overall principle is the same: enumerate all pairings.

[11]In practice, I omit the number corresponding to a tuple's own operator, as it may never pair with itself. However this "full" form tuple allows for a better visual explanation.

3. Finally, we loop over all upper permutations combined with all lower permutations, keeping only *valid* pairings, and throwing away any other combinations

   In this specific case, each upper permutation only forms one *valid* pairing with a single lower permutation; for example `(0,0,2,0)` from $\hat{\mathbf{A}}$ only pairs with `(1,2,0,0)` from $\hat{\mathbf{B}}$. This can be thought of as adding/combining the two partial pairings

$$\begin{array}{ccccc} \texttt{(0, 0, 2, 0)} & + & \texttt{(1, 2, 0, 0)} & & \rightarrow \textit{valid} \\ \hat{P}_{zyx}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{1,kl}\hat{\mathbf{C}}^{2} & + & \hat{P}_{zyx}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{zkl}\hat{\mathbf{C}}^{2} & & \rightarrow \hat{P}_{zyx}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{zkl}\hat{\mathbf{C}}^{yx} \end{array} \tag{5.50}$$

   As a counter-example, consider the following combination, which is not a well-defined pairing. $\hat{\mathbf{A}}$ is trying to pair with $\hat{\mathbf{C}}$ twice, but $\hat{\mathbf{B}}$ is **also** trying to pair with $\hat{\mathbf{A}}$, which doesn't have enough labels for all three attempted pairings:

$$\begin{array}{ccccc} \texttt{(0, 0, 0, 2)} & + & \texttt{(2, 1, 0, 0)} & & \rightarrow \textit{invalid} \\ \hat{P}_{zyx}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{3}\hat{\mathbf{C}}^{kl} & + & \hat{P}_{zyx}\hat{\mathbf{A}}_{1,k}\hat{\mathbf{B}}^{zyk}\hat{\mathbf{C}}^{2} & & \rightarrow \cancel{\hat{P}_{zyx}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{zyk}\hat{\mathbf{C}}^{kl}} \end{array} \tag{5.51}$$

   Clearly, in this case the resulting combination is not possible, and is recognized as such by `termfactory`, and therefore discards this term, continuing on with other permutations.

The explicit connections for each operator are assigned, giving this final output

$\hat{P}_{zyx}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{zyx}\hat{\mathbf{C}}^{kl}$:

```
Pop(rank=3, m=0, n=3, m_a=0, n_a=0, m_b=0, n_b=3, m_c=0, n_c=0)
Aop(rank=2, m=0, n=2, m_p=0, n_p=0, m_b=0, n_b=0, m_c=0, n_c=2)
Bop(rank=3, m=3, n=0, m_p=3, n_p=0, m_a=0, n_a=0, m_c=0, n_c=0)
Cop(rank=2, m=2, n=0, m_p=0, n_p=0, m_a=2, n_a=0, m_b=0, n_b=0)
```

$\hat{P}_{zyx}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{zyk}\hat{\mathbf{C}}^{xl}$:

```
Pop(rank=3, m=0, n=3, m_a=0, n_a=0, m_b=0, n_b=0, m_c=0, n_c=0)
Aop(rank=2, m=0, n=2, m_p=0, n_p=0, m_b=0, n_b=1, m_c=0, n_c=1)
Bop(rank=3, m=3, n=0, m_p=2, n_p=0, m_a=1, n_a=0, m_c=0, n_c=0)
Cop(rank=2, m=2, n=0, m_p=1, n_p=0, m_a=1, n_a=0, m_b=0, n_b=0)
```

$\hat{P}_{zyx}\hat{\mathbf{A}}_{kl}\hat{\mathbf{B}}^{zkl}\hat{\mathbf{C}}^{yx}$:

```
Pop(rank=3, m=0, n=3, m_a=0, n_a=0, m_b=0, n_b=0, m_c=0, n_c=0)
Aop(rank=2, m=0, n=2, m_p=0, n_p=0, m_b=0, n_b=2, m_c=0, n_c=0)
Bop(rank=3, m=3, n=0, m_p=1, n_p=0, m_a=2, n_a=0, m_c=0, n_c=0)
Cop(rank=2, m=2, n=0, m_p=2, n_p=0, m_a=0, n_a=0, m_b=0, n_b=0)
```

## 5.7 The Language of Pairing Terms

The motivation for the title of this section rests in my desire to express the pairing of terms as a language, such as would be defined in Formal Language Theory. It seemed evident to me that this representation would lend itself to distinguishing between *valid* and *invalid* terms and pairings. I also hoped to leverage the work of that field by finding this alternative representation. I was not successful in this original goal, but an offshoot of this investigation did produce results in the form of a Integer linear programming (ILP) approach, that I plan to integrate into `termfactory` in the near future, improving the robustness and ansatz flexibility. There is a small section with more detail on Formal Language Theory in Appendix C.3.

### 5.7.1 Motivation

My work on automatically generating pairings has been successful, within specific constrained problem sets. However, I identified an important limitation in the approach developed to date. The straightforward approach to capturing the constraints embeds those constraints in control flow. In other words, a rule such as $a < b$ is captured via an explicit programming if-statement that checks `if a < b`. This can be clearly seen on lines 8, 13 and 14 in Listing 5.2. While this seems a natural and straightforward method of implementing the rule, it results in several problems:

1. An experienced programmer is needed to change rules, and requires going through a full edit-compile-test cycle every time we change, modify or extend the rule set which defines an ansatz
2. When we have many rules that all interact at the same time, our program can become quite complex to read and modify
3. The program control flow tends to solidify around one single type or class of problem; that is, we embed assumptions, heuristics, tradeoffs, and other elements into the control flow, thus making the program difficult to extend to other domains. I have discussed `termfactory` with other developers working with Logic Programming languages and there was difficulty adapting my code to their problem area.

I am thus motivated to find a more general method for expressing the constraints in the series expansion generator, such that I can avoid large software overhead, and make the software more extensible. Ideally, such a method would not use control flow to represent constraints, but would represent constraints as data, which we expect to be both more malleable and more readily understandable by a user. By taking a declarative approach, defining the problem that needs to be solved (in the form of constraints) and solving that problem without defining the logic required to find a solution

There are three primary objectives for a generalized approach to general term pairing:

1. Express constraints as data rather than as control flow
2. Develop a more general solution approach
3. Employ existing established software to quickly and correctly generate expansions

Additionally, along with these objectives, consideration should be applied to the difficulty of "pivoting" with the associated approach. If the definition of the $\mathbf{R}_0, \mathbf{R}_1, \cdots$ are changed, and therefore the associated equations are different, we would like it to be the case that modifying the program to produce the full set of equations should not be prohibitively difficult for a reasonably skilled user.

To this end, I investigated three different areas that I thought might provide an answer: Formal language theory, Logic Programming, and Integer linear programming (ILP). During my research the most viable approach, given the time constraints, seems to be the ILP approach. While Formal language theory felt the most natural approach, it unfortunately appears to be just as complex as using a software-based control-flow approach. Logic programming does appear to be quite general, and powerful! However the overhead associated with developing the relevant skills to design such a piece of software suggested that this approach would be troublesome to use even by reasonably skilled users. In the end ILP seemed the best fit for this problem.

### 5.7.2 Logic Programming

One method for describing constraints as data is to express constraints in formal logic, and use a logic programming language to automatically compute the implication of the constraints. Programs written in logic programming languages are sets of sentences in logical form, expressing rules and facts about problem domains. Logic programming can be done in a variety of languages, two of which I investigated: Prolog, and Maude[51, 52, 53]

#### Constraints in Prolog

Prolog is a declarative programming language. Prolog uses Horn clause logic to define a problem's constraints, and then allows the user to ask Prolog to determine whether statements are true, based on the constraints (Clocksin & Mellish, 1981). For example, in Listing 5.5 I present a basic Prolog program to calculate all possible combinations of specific coins that add up to 25 cents.

```
change([Q,D,N,P]) :-
    member(Q,[0,1,2,3,4]),                  /* quarters     */
    member(D,[0,1,2,3,4,5,6,7,8,9,10]) ,    /* dimes        */
    member(N,[0,1,2,3,4,5,6,7,8,9,10,       /* nickels      */
              11,12,13,14,15,16,17,18,19,20]),
    S is 25*Q +10*D + 5*N,
    S =< 25,
    P is 25-S.
```

Listing 5.5: Basic Prolog program

The program defines the change from a dollar as being made up of half-dollars, quarters, dimes, nickels, and pennies; each coin can occur up to a maximum of `n` times, where `n` times the coin $= 25$, and the sum `S` of the coins must be 25.

Having defined the constraints on the coins, we can use the Prolog to explore the solution space for all valid solutions. If after loading the above program into the Prolog interpreter, we input the query `change([H,Q,D,N,P])` and then we press the semi-colon key `;` repeatedly, we get the following solutions:

```
Q = D, D = N, N = 0,
P = 25 ;
```

```
Q = D, D = 0,
N = 1,
P = 20 ;
```

```
Q = D, D = 0,
N = 2,
P = 15 ;
```

```
Q = D, D = 0,
N = 3,
P = 10 ;
```

```
Q = D, D = 0,
N = 4,
P = 5 ;
```

```
Q = D, D = P, P = 0,
N = 5 ;
```

```
Q = N, N = 0,
D = 1,
P = 15 ;

```

```
Q = 0,
D = N, N = 1,
P = 10 ;
```

```
Q = 0,
D = 1,
N = 2,
P = 5 ;
```

```
Q = P, P = 0,
D = 1,
N = 3 ;
```

```
Q = N, N = 0,
D = 2,
P = 5 ;
```

```
Q = P, P = 0,
D = 2,
N = 1 ;
```

```
Q = 1,
D = N, N = P, P = 0 ;
```

Now let us consider a Prolog program that might solve constraints applicable to our expansion. Consider a general term $\hat{P}^M_N \hat{\mathbf{A}}^M_N \hat{\mathbf{B}}^M_N$. In the Prolog code I use alphabetic characters as variables to represent the $M$, $N$ labels; assigned from top-to-bottom, left-to-right:

$$\hat{P}^M_N \hat{\mathbf{A}}^M_N \hat{\mathbf{B}}^M_N \to P^a_b B^c_d C^e_f \tag{5.52}$$

I then define the constraints from Section 5.3 in terms of these variables. The Balanced operators constraint ($M_{\text{tot}} = N_{\text{tot}}$) is as simple as

$$a + c + e = b + d + f. \tag{5.53}$$

The Normal ordering constraints

$$M \leq N_{\text{tot}} - N$$
$$N \leq M_{\text{tot}} - M$$

expressed in alphabetic variables

- $\hat{P}$

$$a \leq (b + d + f) - b = d + f \quad (5.54)$$
$$b \leq (a + c + e) - a = c + e \quad (5.55)$$

- $\hat{\mathbf{B}}$

$$c \leq (b + d + f) - d = b + f \quad (5.56)$$
$$d \leq (a + c + e) - c = a + e \quad (5.57)$$

- $\hat{\mathbf{C}}$

$$e \leq (b + d + f) - f = b + d \quad (5.58)$$
$$f \leq (a + c + e) - e = a + c \quad (5.59)$$

in Listing 5.6 I provide a Prolog program which solves for these constraints. However, I restrict it to the case where $\hat{P}$ has a maximum rank of 1, both $\hat{A}$, $\hat{B}$ have maximum ranks of 2, and $\bar{f} = 0$ to reduce the number of output terms. Note that variables in Prolog begin with a capital letter, otherwise I would have used lowercase letters $a, b, c, d, e, f$.

```
legal([A,B,C,D,E,F]) :-
    % define M, N labels for each operator
    member(A, [0, 1]),
    member(B, [0, 1]),
    member(C, [0]),
    member(D, [0, 1, 2]),
    member(E, [0, 1, 2]),
    member(F, [0]),

    %! Balanced Operators constraint
    (A + C + E) =:= (B + D + F),

    %! --- Normal Ordering constraints ---

    %! P operator can't pair with itself
    A =< (D + F),
    B =< (C + E),

    %! A operator can't pair with itself
    C =< (B + F),
    D =< (A + E),

    %! B operator can't pair with itself
    E =< (B + D),
    F =< (A + C).
```

Listing 5.6: Prolog code which finds the $M$ $N$ values for which $\hat{P}^M_N \hat{\mathbf{A}}^M_N \hat{\mathbf{B}}^M_N$ is a *valid* term.

If we run this program in the Prolog interpreter and search for all solutions, we obtain:

```
A = B, B = C, C = D, D = E, E = F, F = 0 ;
```

$P^0_0 \hat{\mathbf{A}}^0_0 \hat{\mathbf{B}}^0_0$

```
A = B, B = C, C = F, F = 0,
D = E, E = 1 ;
```

$P^0_0 \hat{\mathbf{A}}^0_1 \hat{\mathbf{B}}^1_0$

```
A = B, B = C, C = F, F = 0,
D = E, E = 2 ;
```

$P^0_0 \hat{\mathbf{A}}^0_2 \hat{\mathbf{B}}^2_0$

```
A = C, C = D, D = F, F = 0,
B = E, E = 1 ;
```

$P^0_1 \hat{\mathbf{A}}^0_0 \hat{\mathbf{B}}^1_0$

```
A = C, C = F, F = 0,
B = D, D = 1,
E = 2 ;
```

$P^0_1 \hat{\mathbf{A}}^0_1 \hat{\mathbf{B}}^2_0$

```
A = D, D = 1,
B = C, C = E, E = F, F = 0 ;
```

$P^1_0 \hat{\mathbf{A}}^0_1 \hat{\mathbf{B}}^0_0$

```
A = E, E = 1,
B = C, C = F, F = 0,
D = 2 ;
```

$P^1_0 \hat{\mathbf{A}}^0_2 \hat{\mathbf{B}}^1_0$

```
A = B, B = D, D = E, E = 1,
C = F, F = 0 ;
```

$P^1_1 \hat{\mathbf{A}}^0_1 \hat{\mathbf{B}}^1_0$

```
A = B, B = 1,
C = F, F = 0,
D = E, E = 2 ;
```

$P^1_1 \hat{\mathbf{A}}^0_2 \hat{\mathbf{B}}^2_0$

### Issues with Prolog

The above programs show that it is possible to capture constraints in Prolog, and to obtain solutions that satisfy those constraints. However, there are several issues that make the use of Prolog less than satisfactory.

1. Prolog requires a separate interpreter. We would have to package up the interpreter with our Python routines to make a complete package, as well as provide a mechanism to supply input to Prolog and then post-process its output
2. Prolog is not in general easy to program
   (a) Most programmers are not familiar with the declarative Horn clause logic style, and find it an effort to depart from their imperative programming style
   (b) Prolog's only data structure is lists
   (c) The error diagnostics of the interpreter can be very confusing to beginning programmers
3. The Prolog community is small and it can be difficult to get help with problems
4. It is difficult to estimate the complexity of our Prolog program, and how expensive it would be as we expand the number of constraints. Since Prolog does tree searching across all possibilities, it may consume a very significant amount of time

For these reasons we discontinued further investigation of Prolog, although it still retains value for rapid prototyping of sets of constraints.

### Maude

The Maude website describes the language as:[54]

*Maude is a high-performance reflective language and system supporting both equational and rewriting logic specification and programming for a wide range of applications. Maude has been influenced in important ways by the OBJ3 language, which can be regarded as an equational logic sublanguage. Besides supporting equational specification and programming, Maude also supports rewriting logic computation. Rewriting logic is a logic of concurrent change that can naturally deal with state and with concurrent computations. [...]*

For my purposes the draw to use Maude was the rewriting logic capabilities. It seemed ideal for representing the application of Wick's theorem in a concise well defined fashion. The initial plan was to start with a basic mathematical description of the operator expansions, such as those in Equations (5.18) to (5.20), then process product terms similar to Section 5.1.2. These product terms could be rewritten according to constraints as described in Section 5.3.

To understand a bit more about how it works I present the simple vending machine example in Maude:

```
fmod VENDING-MACHINE-SIGNATURE is
  sorts Coin Item Marking .
  subsorts Coin Item < Marking .
  op __ : Marking Marking -> Marking [assoc comm id: null] .
  op null : -> Marking .
  op \( : -> Coin [format (r! o)] .
  op q : -> Coin [format (r! o)] .
  op a : -> Item [format (b! o)] .
  op c : -> Item [format (b! o)] .
endfm
```

Listing 5.7: Example from Maude Manual[53, Pg. 150]

We can modify it to match the Prolog coin example in Listing 5.5:

```
fmod VENDING-MACHINE-SIGNATURE is
  sorts Coin Item Marking .
  subsorts Coin Item < Marking .
  op __ : Marking Marking -> Marking [assoc comm id: null] .
  op null : -> Marking .
  op \) : -> Coin [format (r! o)] .
  op q : -> Coin [format (r! o)] .
  op a : -> Item [format (b! o)] .
  op c : -> Item [format (b! o)] .
endfm
```

Listing 5.8: Modification of Listing 5.7 to match Listing 5.5.

Exploring the solution space is not as simple as in Prolog, instead we must use the specific tools, in this case `reduce`. Running `reduce in META-LEVEL` we can explore the solution space like so

```
Maude> reduce in META-LEVEL : upModule('VENDING-MACHINE-SIGNATURE, false)
    .
reduce in META-LEVEL : upModule('VENDING-MACHINE-SIGNATURE, false) .
rewrites: 1 in 0ms cpu (0ms real) (~ rewrites/second)
result FModule: fmod 'VENDING-MACHINE-SIGNATURE is
  including 'BOOL .
  sorts 'Coin ; 'Item ; 'Marking .
  subsort 'Coin < 'Marking .
  subsort 'Item < 'Marking .
  op '$ : nil -> 'Coin [format('r! 'o)] .
  op '__ : 'Marking 'Marking -> 'Marking [assoc comm id('null.Marking)] .
  op 'a : nil -> 'Item [format('b! 'o)] .
  op 'c : nil -> 'Item [format('b! 'o)] .
  op 'null : nil -> 'Marking [none] .
  op 'q : nil -> 'Coin [format('r! 'o)] .
  none
  none
endfm
```

Listing 5.9: Maude vending machine result.

To give more context I present a very basic module to solve the sample problem as Listing 5.6:

```
mod RESIDUALS is

  protecting BOOL .
  protecting INT .

  *** define integer variables
  var A, B, C, D, E, F : INT


  M_tot = A + C + E .
  N_tot = A + C + E .

  *** Balanced Operators constraint
  M_tot == N_tot .

  A + B = R1 .
  C + D = R2 .
  E + F = R3 .


  *** Normal ordering constraints
  A <= N_tot - B .
  B <= N_tot - A .

  C <= N_tot - d .
  D <= N_tot - c .

  E <= N_tot - f .
  F <= N_tot - e .

endfm
```

Listing 5.10: Maude code which finds the $M$ $N$ values for which $\hat{P}_N^M \hat{\mathbf{A}}_N^M \hat{\mathbf{B}}_N^M$ is a *valid* term.

```
mod RESIDUALS is

protecting BOOL .
protecting NAT .
protecting INT .

*** define the types of terms
sorts ValidTerm InvalidTerm Term .
subsorts ValidTerm InvalidTerm < Term .

*** a variable for any term
var T : Term .
var VT : ValidTerm .
var IVT : InvalidTerm .

*** define the set of valid terms
sort ValidSet .
subsort ValidTerm < ValidSet .
op empty : -> ValidSet [ctor ].
op _;_ : ValidSet ValidSet -> ValidSet [ctor assoc comm id: empty] .

*** define membership in the set of valid terms
op _in_ : Term ValidSet -> Bool .
var SetOfValidTerms : ValidSet .

eq T in T ; SetOfValidTerms = true .
eq T in SetOfValidTerms = false [owise] .


*** define the set of invalid terms
sort InvalidSet .
subsort InvalidTerm < InvalidSet .
op empty : -> InvalidSet [ctor ].
op _;_ : InvalidSet InvalidSet -> InvalidSet [ctor assoc comm id: empty] .

*** define membership in the set of Invalid terms
op _in_ : Term InvalidSet -> Bool .
var SetOfInvalidTerms : InvalidSet .
```

Listing 5.11: Example Maude code showing set inclusion.

Clearly Maude is very powerful and has very intuitive mathematical notation. However trying to go further than simple examples becomes extremely challenging. In Maude the design paradigm is not clear, and there are many ways one could solve a problem that a certain amount of selection paralysis is present. In addition, the pros/cons of certain choices are obscure to non-experts and can have hidden implications. In the end, it seems the extra overhead associated with having a more powerful and dynamic tool is too high a cost. It may be that Maude is a good solution to this problem, but using it requires more years of experience with the tool. In the end I found it disheartening to have to abandon this approach because, at first, it appeared to be a very good fit with high flexibility.

Let us now review. While the logic programming approach does show promise for simple examples, it rapidly becomes complicated when the examples contain more constraints. Furthermore, both Prolog and Maude are relatively baroque programming environments in which basic arithmetic and output are done in a non-traditional manner. This limits the usability of such tools. For these reasons, I discontinued looking at logic programming, and pursued a third option.

### 5.7.3 Linear programming

A third area for investigation that has proved more fruitful is *linear programming.* A linear programming problem is a mathematical optimization problem[12] where all constraints are expressible by linear relationships. The linear relationships define a convex polytope over an n-dimensional space, where n is the number of variables. For example, in a two-variable $(x, y)$ case a set of six inequalities might define the following convex polytope[13]

[12]A problem where the goal is to select the "best" element, with regards to some criterion, from some set of available alternatives

[13]A line segment is 1d, a polygon 2d, a polyhedron 3d, and polytopes are $N$d.

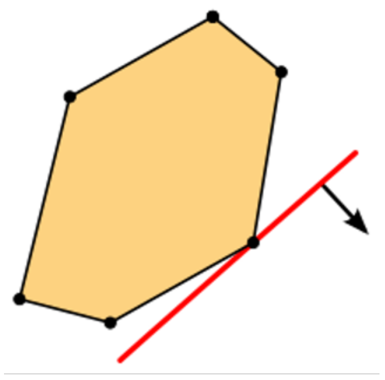

Figure 5.1: A pictorial representation of a simple linear program. The set of feasible solutions is depicted in yellow and forms a polygon, a 2-dimensional polytope. The optimum of the linear cost function is where the red line intersects the polygon. The red line represents the objective function that we are attempting to minimize, and the arrow indicates the direction in which we are optimizing [55].

Here the red line represents the objective function that we are attempting to minimize: the minimum of the objective function that is consistent with the constraints is where the red line touches the convex polytope. This linear programming problem has a unique solution.

### Integer linear programming

Integer linear programming (ILP) is a special case of linear programming, in which the variables are restricted to integer values. Linear programming is a general problem that has wide applicability in science and economics, and finds use in planning, routing, scheduling, and assignment. The problem has been studied since at least 1947, when Dantzig invented the simplex method to solve linear programming problems [56].

If we can express our problem as an ILP problem, we can take advantage of the simplex method (or other known algorithms) to solve it. The general linear programming problem expressed in canonical form is: [14]

$$\begin{array}{ll} \text{Find a vector} & \boldsymbol{x} \\ \text{that maximizes} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ \text{and} & \boldsymbol{x} \geq \mathbf{0}. \end{array}$$

Let us use the same example and constraints as defined in Equation (5.52):

$$\hat{P}^M_N \hat{\mathbf{A}}^M_N \hat{\mathbf{B}}^M_N \rightarrow P^a_b B^c_d C^e_f$$

Our goal is to find integer values for each superscript and subscript, such that the result is a valid term in our series expansion. To represent this as a linear programming problem, we

[14] Inequality comparisons are performed elementwise over tensors.

need to define $\boldsymbol{x}$,$\boldsymbol{c}$,$\boldsymbol{A}$, and $\boldsymbol{b}$. We can rewrite the constraints as a set of linear relations on their variables $a, b, c, d, e, f$, by moving all variables to one side of the equation:

Table 5.1: Canonical form of linear relations

| Constraint | Linear relation in canonical form |
|---|---|
| $a + c + e = b + d + f$ | $a - b + c - d + e - f = 0$ |
| $a \leq d + f$ | $a + 0b + 0c - d + 0e - f \leq 0$ |
| $b \leq c + e$ | $0a + b - c + 0d + e + 0f \leq 0$ |
| $c \leq b + f$ | $0a - b + c + 0d + 0e - f \leq 0$ |
| $d \leq a + e$ | $-a + 0b + 0c + d - e + 0f \leq 0$ |
| $e \leq b + d$ | $0a - b + 0c - d + e + 0f \leq 0$ |
| $f \leq a + c$ | $-a + 0b - c + 0d + 0e + f \leq 0$ |

Then $\boldsymbol{x}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ will be

$$\boldsymbol{x} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{5.60}$$

I choose $\boldsymbol{c} = \boldsymbol{b}$ since I am not attempting to minimize any linear objective function and so can set $\boldsymbol{c}$ to zeroes. The constraints will be split into two matrices. The equality constraint $A_{\text{eq}}$ will be captured this way:

$$\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}, \tag{5.61}$$

and our inequality constraints $A_{\text{ub}}$ will be captured this way:

$$\begin{array}{cccccc} a & b & c & d & e & f \\ \end{array}$$
$$\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & -1 & 1 & 0 \\ -1 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}. \tag{5.62}$$

This is the complete set of specifications of our term pairing problem, expressed as an integer linear programming problem.

- Our first objective has been achieved: we have been able to express the constraints as data instead of control flow.

- Our second objective has also achieved, because we can use this technique to express a wide variety of problems, over any possible set of variables and constraints, so long as we are limited to linear combinations.
- We next show that our third objective can also be achieved, because there already exists a rich set of library functions that can solve our integer linear programming problem.

The third-party Python library **Scipy** [57] has a module `optimize` which provides "functions for minimizing (or maximizing) objective functions, possibly subject to constraints". Specifically it contains a general solver for linear programming problems: `optimize.linprog`. This function takes as input the exact vectors and matrices we have outlined above, and produces a set of values for our variables $a, b, c, d, e, f$. Since `linprog` will produce only one "optimal" value for a given set of input, we can explore the solution space by controlling the values of our individual variables. Mechanically, this means that given a set of solutions $\mathcal{S}_1$ which has some optimal solution $k_1$ we produce a new set of solutions $\mathcal{S}_2 = \mathcal{S}_1 \setminus k_1$ which will have some optimal solution $k_2$, and we can repeat this process until we end up with no solutions left $\emptyset = \mathcal{S}_n \setminus k_n$.

The following Python script gives an example:

```python
# import packages
import numpy as np
from scipy import optimize

# We have no linear objective function to optimize
c = np.zeros(6)

# AEQ is the equality constraint matrix
# AUB is the inequality constraint matrix
# They have one row per constraint, and one column per variable
#    { a, b, c, d, e, f }
AEQ = np.array([
    [1, -1, 1, -1, 1, -1]
])

AUB = np.array([
    [1, 0, 0, -1, 0, -1],
    [0, 1, -1, 0, -1, 0],
    [0, -1, 1, 0, 0, -1],
    [-1, 0, 0, 1, -1, 0],
    [0, -1, 0, -1, 1, 0],
    [-1, 0, -1, 0, 0, 1]
])

# beq is the equality constraint vector
beq = np.array([[0]])

# bub is the inequality constraint vector
bub = np.array([[0, 0, 0, 0, 0, 0]])

# bnds defines the minimum and maximum value for each variable
# we will redefine it in each iteration of the loop
bnds= [(0,0), (0,0), (0,0), (0,0), (0,0), (0,0)]


# We run a nested loop over the values min to max for each variable,
# making sure we test all values of each variable
min=0
max=2
```

```python
for x, y, z in it.product(range(min,max), repeat=3):
    for X, Y, Z in it.product(range(min,max), repeat=3):
        bnds = [(x,max), (y,max), (z,max), (X,max), (Y,max), (Z,max)]
        result = optimize.linprog(
            c,
            A_ub=AUB, A_eq=AEQ,
            b_eq=beq, b_ub=bub,
            bounds=bnds,
            method="revised simplex"
        )
        print(result.x)
```

Listing 5.12: Python script for solving linear programming problem which is analogous to finding the $M$ $N$ values for which $\hat{P}^M_N \hat{\mathbf{A}}^M_N \hat{\mathbf{B}}^M_N$ is a *valid* term.

The above program will generate output in rows like so: `(1.0, 0.0, 0.0, 0.0, 0.0, 1.0)`, where row is a set of of values for the variables $a, b, c, d, e, f$ that satisfies the constraints expressed in Listing 5.12. In the case where each operator has a maximum rank of 3 the ILP approach finds 729 solutions, of which 199 are unique. Changing to maximum rank of 2 for each operator produces a much more reasonable 64 solutions, of which 29 are unique. By adding a bit of processing code I can print those 29 solutions out as simple LaTeX:

| | | | | | |
|---|---|---|---|---|---|
| $P^0_0 A^0_0 B^0_0$ | $P^0_1 A^1_0 B^0_0$ | $P^1_0 A^0_0 B^0_1$ | $P^1_1 A^0_0 B^1_1$ | $P^1_1 A^1_1 B^0_0$ | $P^1_2 A^1_1 B^1_0$ |
| $P^0_0 A^0_1 B^1_0$ | $P^0_1 A^1_0 B^1_1$ | $P^1_0 A^0_1 B^0_0$ | $P^1_1 A^0_1 B^1_0$ | $P^1_1 A^1_1 B^1_1$ | $P^2_0 A^0_1 B^0_1$ |
| $P^0_0 A^1_0 B^0_1$ | $P^0_1 A^1_1 B^1_0$ | $P^1_0 A^0_1 B^1_1$ | $P^1_1 A^0_1 B^2_1$ | $P^1_1 A^1_2 B^1_0$ | $P^2_1 A^0_1 B^1_1$ |
| $P^0_0 A^1_1 B^1_1$ | $P^0_1 A^1_1 B^2_1$ | $P^1_0 A^1_1 B^0_1$ | $P^1_1 A^1_0 B^0_1$ | $P^1_1 A^2_1 B^0_1$ | $P^2_1 A^1_1 B^0_1$ |
| $P^0_1 A^0_0 B^1_0$ | $P^0_2 A^1_0 B^1_0$ | $P^1_0 A^1_1 B^1_2$ | $P^1_1 A^1_0 B^1_2$ | $P^1_2 A^1_0 B^1_1$ | |

Each line above is a valid set of values for the variables $a, b, c, d, e, f$. Note the following:

1. Constraints expressed as linear combinations are directly usable as data
2. No constraint is expressed as control flow; indeed, there are no conditional statements at all in our program
3. The program is very short and easy (for Python programmers) to read
4. The result is efficient; the results are printed in 1 to 2 seconds

A comment on computational performance and the scaling with respect to the number of variables. I expect this ILP approach to be the same order of magnitude as the original control-state-flow Python implementation, which takes on the order of a few seconds. Even if either approach took a minute or even ten minutes that runtime pales in comparison to the cost of computing the integral **using** the equations that are generated with this approach. For small molecules like $N_2O$, $CH_2O$, $H_2O$, a 50fs propagation takes on the order of a few minutes for Z2 or Z3 truncation, moving to medium-sized molecules like vinyl and cytosine are 10's of minutes, and finally hexahelicene is a few hours.

Scipy's `optimize.linprog` is a very general method that has many options, and is designed for use with large linear optimization problems. It fits very well with our problem, but if it did not, there are a variety of other third-party libraries which provide functionality for solving these types of problems, such as `cvxpy` and Numpy's `linalg` module [39].

A more general discussion of the simplex and revised simplex methods for solving systems of linear equations is beyond the scope of our work, but this is a well-studied problem in numerical analysis and so discussion can be found in good textbooks on numerical analysis. Suffice it to say that by recasting the term pairing problem as an instance of linear programming, we can take advantage of the very extensive amount of work done to solve linear programming problems, without needing to understand how they produce such solutions.

Thus, our third objective, of quickly and correctly generating series expansion terms, has been achieved.

The simplex method should be sufficient to solve our ILPs in short order. in Appendix C.4 I discuss the complexity of linear programming.

## 5.8 Concluding Remarks

The goal of this chapter was to develop a robust software package for generating and calculating derivative terms in the second quantization formalism of the change of the wavefunction of the time-dependent Schrödinger equation (TDSE). I was able to successfully accomplish these goals, and the resulting software package `termfactory` is publicly available on GitHub. This work allows the `t-amplitudes` software package to be used by experimentalists to simulate and compare spectra with real-world results.

This package also provides a foundation for future research into thermal properties of vibronic systems or spectra of excited state systems. Both of these require solving the problem for the general form, as opposed to the simplified form. Producing a software package that only supported the simplified form with very restricted contractions would have been a much simpler task. By anticipating future expansion, I reduced the overall development time and helped accelerate the ansatz prototyping process, which was a key element in the success or failure of the `t-amplitudes` software package. This somewhat higher overhead at the outset of the project was seen to have a significant future payoff.

With regards to the exploration of solving the problem using ILP, I have a proof-of-principle implementation. As the current implementation is sufficient to complete the job, and due to time constraints, adding the ILP approach to `termfactory` has not been a priority. I do plan in the near future to integrate this approach into the code base.

# Chapter 6

# Application of `termfactory` and `t-amplitudes` to VECC methodology and calculation of Vibronic Spectra

In this chapter equations for some quite involved variants of the VECC methodology will be presented. The equations are cast in general operator form, and the task of our software tools is to derive detailed working equations and provide implementations for procedures that are then assembled in computer code that can perform actual simulations of spectra. The actual implementation is compute-intensive, and thus I will discuss computational improvements which allow for these expensive EOM to be solved within a reasonable time frame. Finally, I will present spectra that I generated for various molecules.

The equations to be discussed are superficially quite different from the equations discussed in the previous chapter. However, there are clear pathways to connect the formulations, and to provide the relevant equations to the newly developed software tools.

In the previous chapter the problem was outlined as computing an expectation value,

$$\left\langle \hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\mathcal{C}}\cdots \right\rangle . \tag{6.1}$$

It was shown in Section 2.3.1, specifically Equation (2.49), that quantum mechanical (QM) EOM can be expressed in terms of these expectation values. Thus, by calculating all non-zero contributions and evaluating them, we solve these EOM. This allows us to propagate a wavefunction in real time. With a method for propagating our wavefunction we have a pathway to calculating an ACF. As was shown in Section 2.3.2 we can obtain a power spectra from an ACF.

The operator equations derived in this chapter can be evaluated using the tools in termfactory and they are then implemented in actual computer code using t-amplitudes. In the final computer code one solves an ordinary differential equation for the t-amplitudes, which can be used to obtain the ACF. As was shown in Section 2.3.2 we can obtain a power spectra from an ACF.

Therefore, Chapter 5 is all about computing QM EOM so that we can calculate an ACF from which we can obtain our theoretical spectra.

## 6.1 Derivation of Vibrational Electronic Coupled Cluster approaches

The goal is the generation of vibrationally-resolved electronic spectra through the Fourier transform of an auto-correlation function (ACF)

$$ACF(\tau) = \sum_{a,b} X_a \mathcal{U}_{ab}(\tau) X_b = \sum_{a,b} X_a \langle 0, a| e^{-i\hat{H}\tau} |0, b\rangle X_b, \tag{6.2}$$

by numerically integrating equations of motion over the parameter $\tau$. Work with S. Bao is in progress to also evaluate thermal properties, in which case $\tau$ will refer to inverse temperature $\beta$. In this chapter, real-time propagations are of interest, and $\tau$ is used to denote time $t$. We can apply a fixed ansatz that is based on second quantized operators to parameterize the time-dependent wavefunction. Let us first recall the definition of the vibronic hamiltonian

$$\begin{aligned}\hat{H} = \sum_{ab} |a\rangle \langle b| \Bigg( h^a_b + \sum_a h^{ai}_b \{\hat{i}^\dagger\} + \sum_i h^a_{bi} \{\hat{i}\} + h^{ai}_{bj}\{\hat{i}^\dagger \hat{j}\} \\ + \frac{1}{2}\sum_{ij} h^{aij}_b \{\hat{i}^\dagger \hat{j}^\dagger\} + \frac{1}{2}\sum_{ij} h^a_{bij}\{\hat{i}\hat{j}\}\Bigg),\end{aligned} \tag{6.3}$$

where labels $a, b, c, \cdots$ denote electronic surfaces and $i, j, k, \cdots$ denote vibrational modes.

The relevant matrix $\mathcal{U}$

$$\mathcal{U}_{ab}(\tau) = \langle 0, a| e^{-i\hat{H}\tau} |0, b\rangle\,, \tag{6.4}$$

can be obtained by obtaining wavefunctions $\Psi_b$

$$e^{-i\hat{H}t} |0, b\rangle \equiv |\Psi_b(\tau)\rangle\,, \tag{6.5}$$

that satisfy the TDSE

$$i \left| \frac{\mathrm{d}\Psi_b}{\mathrm{d}\tau} \right\rangle = \hat{H} |\Psi_b(\tau)\rangle\,, \tag{6.6}$$

which allow us to express each matrix element of $\mathcal{U}_{ab}$ in terms of the wavefunction

$$\mathcal{U}_{ab} = \langle 0, a|\Psi_b(\tau)\rangle\,. \tag{6.7}$$

Thus, the full $\mathcal{U}_{ab}$ matrix is computed through independent propagations for each electronic state $|\Psi_b(\tau)\rangle$. Having the full $\mathcal{U}_{ab}$ matrix, one can compute the ACF using Equation (6.2), and then produce a vibrationally-resolved electronic spectra. Computing these propagations requires choosing a representation, or ansatz, for a wavefunction to approximate the 'true' time-dependent wavefunction. Over the course of working on this research project, many different representations were explored. Below, I provide a breakdown of a simple linear representation, and then we discuss the scheme that will be used in actual simulations, and that has proven most successful thus far. The ansatz applies second quantization and has classical DoF. By increasing the truncation order of the ansatz, accuracy can be systematically improved. In the limit the ansatz is an exact solution to the TDSE.

**Linear wavefunction representation**

The simplest approach using second quantization is a linear operator

$$|\Psi_b(\tau)\rangle = \sum_c \hat{Z}_c |0, c\rangle , \tag{6.8}$$

where

$$\hat{Z}_c = z_c^0(\tau) + \sum_k z_c^k(\tau)\hat{k}^\dagger + \frac{1}{2}\sum_{k,l} z_c^{kl}(\tau)\hat{k}^\dagger\hat{l}^\dagger + \cdots , \tag{6.9}$$

in general

$$\hat{Z}_c^{(N)} = \frac{1}{N!}\sum_{k1,k2,\ldots,kN} z_c^{k1,\ldots,kN}\hat{k_1}^\dagger \cdots \hat{k_N}^\dagger. \tag{6.10}$$

with has the initial condition (where $\tau = 0$)

$$Z_c^0 = \delta_{cb}, \tag{6.11}$$

and all other amplitudes are zero. Substituting in the TDSE gives us the following relation

$$i\sum_c \frac{\mathrm{d}\hat{Z}_c}{\mathrm{d}\tau} |0, c\rangle = \sum_c \hat{\mathbf{H}}\hat{Z}_c |0, c\rangle , \tag{6.12}$$

that we can project onto, with $\langle 0, a| \hat{\Omega}_\lambda^\dagger$

$$i \langle 0, a| \hat{\Omega}_\lambda^\dagger \sum_x \frac{\mathrm{d}\hat{Z}_c}{\mathrm{d}\tau} |0, c\rangle = \sum_c \langle 0, a| \hat{\Omega}_\lambda^\dagger \hat{\mathbf{H}}\hat{Z}_c |0, c\rangle , \tag{6.13}$$

where $\lambda$ defines the level of excitations in the bra state, and we ensure we have the same number of equations as we have parameters. Of key importance here is the presence of the term $\frac{\mathrm{d}\hat{Z}_c}{\mathrm{d}\tau}$ on the left-hand-side. Having a closed form expression for the change in the operator $\hat{Z}_c$ with respect to time $\tau$ is necessary for computing the independent propagations for each initial state $|0, b\rangle$.

Since the projection operator $\hat{\Omega}_\lambda$ does not have an electronic label, the equations simplify to

$$i \langle 0, a| \hat{\Omega}_\lambda^\dagger \frac{\mathrm{d}\hat{Z}_a}{\mathrm{d}\tau} |0, a\rangle = \sum_c \langle 0, a| \hat{\Omega}_\lambda^\dagger \hat{\mathbf{H}}\hat{Z}_c |0, c\rangle . \tag{6.14}$$

To propagate the ACF and produce a spectrum one needs to calculate $\frac{\mathrm{d}\hat{Z}_c}{\mathrm{d}\tau}$. Starting from the linear wave function representation in Equation (6.13) there is a general expression for each order $N$

$$\frac{\mathrm{d}\hat{Z}_a^N}{\mathrm{d}\tau} = \mathrm{RHS}_a^N \tag{6.15}$$

and it is this RHS that are the residual equations that are derived by the code generator. This is the root of the terminology "residual term/equation". By this point the reader should have some grasp of the relation between the time-derivatives of the form $\frac{\mathrm{d}\hat{Z}_x}{\mathrm{d}\tau} = a+b+c+\cdots$ and the overall goal of computing vibrationally-resolved electronic spectra.

## Mixed Exponentional-Linear approach with Ehrenfest projections

In Section 2.3 we discussed that in the course of this work a number of approaches have been explored, but we ran into many failures. In this section we present a fairly complicated scheme that yields fairly satisfactory results.

We introduce a special type of mixed exponential/linear parameterization for the time-dependent wavefunction

$$|\Psi(\tau)\rangle = e^{\hat{T}(\tau)} \sum_c \hat{Z}_c(\tau) |c, 0\rangle \tag{6.16}$$

where

$$\hat{T} = \sum_i t^i \hat{i}^\dagger \tag{6.17}$$

is the same for each electronic state and only contains single excitations. The purpose of this operator is essentially to provide a moving reference state that is based on a weighted average over the various electronic states. Details will follow.

The operators $\hat{Z}_c$ are the same as in the linear ansatz.

$$\hat{Z}_c = z_c^0 + \sum_i z_c^i \hat{i}^\dagger + \frac{1}{2}\sum_{ij} z_c^{ij} \hat{i}^\dagger \hat{j}^\dagger + \cdots \tag{6.18}$$

The initial conditions are $t_i = 0, z_c^0 = \delta_{bc}$ to propagate starting from surface $b$ in its initial ground state configuration. All other $z$-amplitudes vanish initially.

We substitute this ansatz in Equation (6.16) into TDSE and multiply by $e^{-\hat{T}}$ on both sides of the equation

$$\sum_c (i\frac{d\hat{T}}{d\tau}\hat{Z}_c + i\frac{d\hat{Z}_c}{d\tau}) |0, c\rangle = \sum_c e^{-\hat{T}} \hat{\mathbf{H}} e^{\hat{T}} \hat{Z}_c |0, c\rangle \tag{6.19}$$

It is convenient to define the transformed operator

$$\bar{\mathbf{H}} = e^{-\hat{T}} \hat{\mathbf{H}} e^{\hat{T}} \tag{6.20}$$

which can be computed outside the code-generator and which has the same operator rank as $\hat{\mathbf{H}}$

In the next step we need to define a projection manifold. Here we use the same moving (nuclear) reference state, but now acting on the bra side of the equation, represented by $\langle 0,a| e^{\hat{T}^\dagger}$ and using the the fact that $\hat{T}^\dagger$ commutes with $\hat{\Omega}_\nu^\dagger$ we obtain

$$\langle 0,a| \hat{\Omega}_\nu^\dagger e^{\hat{T}^\dagger}(i\frac{d\hat{T}}{d\tau}\hat{Z}_a + i\frac{d\hat{Z}_a}{d\tau})|0,a\rangle = \sum_c \langle 0,a| \hat{\Omega}_\nu^\dagger e^{\hat{T}^\dagger}\bar{\mathbf{H}}\hat{Z}_c|0,c\rangle \tag{6.21}$$

The goal of introducing this modification of projection manifold is to shift the center of the reference state (in the projection manifold) and to align it with the center of an evolving ket wave function $|\Psi\rangle$. The exponential incorporates contributions of higher-level excitations on top of the linear $\hat{Z}$ ansatz. We emphasize that the parameterization of the wave function is not affected by the presence of $e^{\hat{T}^\dagger}$, but the equations and therefore the values of the parameters will change. As a consequence, both accuracy and robustness is expected to be be improved by introducing the modification of projection manifold.

The effect of the operator $e^{\hat{T}^\dagger}$ can be treated as another similarity transform. We can define

$$\begin{aligned}\tilde{\mathbf{H}} &= e^{\hat{T}^\dagger}\bar{\mathbf{H}}e^{-\hat{T}^\dagger}\\ \tilde{Z}_c &= e^{\hat{T}^\dagger}\hat{Z}_c e^{-\hat{T}^\dagger}\\ \frac{d\tilde{Z}_a}{d\tau} &= e^{\hat{T}^\dagger}\frac{d\hat{Z}_a}{d\tau}e^{-\hat{T}^\dagger}\\ \frac{d\tilde{T}}{d\tau} &= e^{\hat{T}^\dagger}\frac{d\hat{T}}{d\tau}e^{-\hat{T}^\dagger}\end{aligned}$$

And from the fact that $e^{\hat{T}^\dagger}|0,c\rangle = |0,c\rangle$, since $\hat{T}^\dagger$ only has annihilation operators only, the equations simply reduce to

$$\langle 0,a| \hat{\Omega}_\nu^\dagger(i\frac{d\tilde{T}}{d\tau}\tilde{Z}_a + i\frac{d\tilde{Z}_a}{d\tau})|0,a\rangle = \sum_c \langle 0,a| \hat{\Omega}_\nu^\dagger\tilde{\mathbf{H}}\tilde{Z}_c|0,c\rangle \tag{6.22}$$

The parameterization of the mixed exponential/linear ansatz is redundant. There is in a sense complete freedom to choose the $\hat{T}$ operator while the $\hat{Z}$ equations are adjusted accordingly, and the approach remains exact in the limit of a complete expansion of $\hat{Z}$ operators.

To obtain a suitably averaged equation for $\hat{T}$, we manipulate the above equation as follows. We remove the time derivative $\frac{d\tilde{Z}_a}{d\tau}$, and retain only the constant (or reference) contribution $\tilde{Z}_c^0$ on both sides of the equation. Finally, we multiply each equation by $(\tilde{Z}_a^0)^*$ and sum the electronic components to obtain the averaged equation

$$\sum_a(\tilde{Z}_a^0)^*\langle 0,a| \hat{\Omega}_\nu^\dagger(i\frac{d\tilde{T}}{d\tau})\tilde{Z}_a^0|0,a\rangle = \sum_{a,c}(\tilde{Z}_a^0)^*\langle 0,a| \hat{\Omega}_\nu^\dagger\tilde{\mathbf{H}}\tilde{Z}_c^0|0,c\rangle \tag{6.23}$$

Because the operator $\hat{T}$ has no dependence on electronic labels (and it only has singles, $\hat{T}_1$), the equation simplifies to

$$i\frac{dt^i}{d\tau} = \frac{1}{C}\sum_a (\tilde{Z}_a^0)^* \sum_c \langle 0, a| \hat{\Omega}_\nu^\dagger \tilde{\mathbf{H}} \tilde{Z}_c^0 |0, c\rangle \tag{6.24}$$

where C is a positive normalization constant $C = \sum_a (\tilde{Z}_a^0)^* \tilde{Z}_a^0$ This latter equation clearly indicates the averaging over the electronic states for the amplitudes of the $\hat{T}$ operator. To implement the equations inside the code generator requires a substantial amount of work and there are still many steps that are done manually. One needs to manipulate equations further to solve for the amplitudes of the untransformed $\hat{Z}_a$ operator. All of these manipulations were done in hand-coded implementation. The code generator evaluates the basic multiplication using the transformed amplitudes in Equation (6.22).

### Discussion equation generator and Left hand side

From the main working equation in Equation (6.22), the reader can glean the general form of terms that can be generated by `termfactory`. However, many special issues are adding up that are better implemented by hand. Taking a step back, at the outset of this research the linear and pure exponential representations were investigated, and only after further development was the mixed exponential/linear representation explored. Therefore the language around residual equations originated with the $\frac{\mathrm{d}\hat{Z}_x}{\mathrm{d}\tau}$ = RHS shape. However, for the mixed representation due to the presence of two operators $e^{\hat{T}^\dagger}$ and $\hat{Z}$, what was a singular derivative term on the left hand side of the equation instead becomes multiple derivative terms. Further complications arise as the equations are cast in terms of transformed operators, and the amplitudes need to be solved for. Determining the change of an operator $\hat{A} \in \{\hat{T}, \hat{Z}\}$ is handled in the following abstract fashion: [1]

$$\frac{\mathrm{d}\hat{A}}{\mathrm{d}\tau} + \mathrm{LHS} = \mathrm{RHS}, \tag{6.25}$$

$$\frac{\mathrm{d}\hat{A}}{\mathrm{d}\tau} = \mathrm{RHS} - \mathrm{LHS}. \tag{6.26}$$

The scope of the project expanded to calculating these left-hand-side terms as well. Thankfully, the logic and procedure is **very** similar to the right-hand-side terms. Therefore the majority of the chapter will focus on the generation of the Residual equation (RHS) and their terms, with a small section at the end explaining the small changes needed to allow generation of the LHS equations and terms.

We next look at methods to improve the efficiency of the VECC approach, particularly those that can reduce the computational scaling with respect to DoFs.

[1] Where LHS is everything **except** the derivative I am trying to calculate.

## 6.2 Computational Costs

Computational Scaling is extremely important as it can make or break this tool. Consequently we looked at a number of different techniques to improve the performance of the tool when scaling up our problem sizes.

### 6.2.1 Optimized Einsum

The most computationally expensive part of the VECC approach is computing the individual terms. These operations are executed using Numpy's `einsum` function[2]. It provides a very useful interface to define mathematical operations in Einstein notation. For example:

| | |
|---|---|
| • trace | `np.einsum('ii', a)` |
| • inner product | `np.einsum('i,i', a, b)` |
| • outer product | `np.einsum('i,j', a, b)` |
| • matrix transpose | `np.einsum('ij->ji', c)` |
| • matrix multiplication | `np.einsum('ij,jk -> ik', a, b)` |

Increasing the efficiency of these computations is of great interest. We next describe some techniques used to achieve efficiency.

**Linking to BLAS libraries** One straightforward approach is to compile Numpy's library with links to a BLAS library such as IntelMKL or openBLAS [34, 35, 36, 37, 38]. This allows low-level mathematical operations (matrix products) to use optimized C code, which is much faster than any Python implementation. This is standard practice for all my calculations[3].

**Path optimization** Besides linking to optimized libraries, there is also a method for optimizing the path used by `einsum`. The principle here is that while a mathematical operation may have multiple theoretically-equivalent evaluation methods, the real-world performance can vary drastically. In particular, path optimization attempts to reduce the scaling of the big-O cost respective to the number of DoF. This is accomplished by partitioning the calculation up into a number of intermediate steps. By choosing an "optimal" partitioning the maximum scaling of ALL intermediate steps is lower than the scaling of the original calculation.

**`opt_einsum` library** A final optimization was the use of the `opt_einsum` library [58]. Their documentation is well written and so I reproduce two examples to illustrate the use of `opt_einsum` and path optimization. First from their GitHub page's `README` file: Listing 6.1:

[2] https://numpy.org/doc/stable/reference/generated/numpy.einsum.html

[3] At this time I have been using the 2019 version of IntelMKL, however it does seem that more recent versions, 2022 specifically, have significant improvements. The 2022 version seems to require more overhead to install due to the "oneAPI" so I hope to address this possible improvement in the future.

```
import numpy as np
from opt_einsum import contract

N = 10
C = np.random.rand(N, N)
I = np.random.rand(N, N, N, N)

%timeit np.einsum('pi,qj,ijkl,rk,sl->pqrs', C, C, I, C, C)
1 loops, best of 3: 934 ms per loop

%timeit contract('pi,qj,ijkl,rk,sl->pqrs', C, C, I, C, C)
1000 loops, best of 3: 324 us per loop
```

Listing 6.1: Example of tensor contraction performed with `np.einsum` and `opt_einsum`'s `contract`.

Second, from their github.io documentation where they state:

*As an example, consider the following expression found in a perturbation theory (one of* 5,000 *such expressions):*

```
'bdik,acaj,ikab,ajac,ikbd'
```

*At first, it would appear that this scales like* $N^7$ *as there are* 7 *unique indices; however, we can define a intermediate to reduce this scaling.*

```
# (N^5 scaling)
a = 'bdik,ikab,ikbd'
# (N^4 scaling)
result = 'acaj,ajac,a'
```

They go on to show the use of `oe.contract_path` to find a different path in Listing 6.2:

```
path_info = oe.contract_path('bdik,acaj,ikab,ajac,ikbd->', *views)

    print(path_info)
#>   Complete contraction:  bdik,acaj,ikab,ajac,ikbd->
#>          Naive scaling:  7
#>      Optimized scaling:  4
#>       Naive FLOP count:  2.387e+8
#>   Optimized FLOP count:  8.068e+4
#>    Theoretical speedup:  2958.354
#>   Largest intermediate:  1.530e+3 elements
#> --------------------------------------------------------------
#> scaling         BLAS            current                remaining
#> --------------------------------------------------------------
#>    4               0     ikbd,bdik->ikb      acaj,ikab,ajac,ikb->
#>    4     GEMV/EINSUM        ikb,ikab->a             acaj,ajac,a->
#>    3               0       ajac,acaj->a                     a,a->
#>    1             DOT               a,a->                        ->
```

Listing 6.2: Example `path_info` output detailing differences in naive and optimized path contraction.

We can see that the calculation was split into 4 intermediate steps which exhibited a maximal scaling of $\mathcal{O}(4)$ and a theoretical speedup of $\approx 3000$.

I have found in practice that for systems with less than 15 DoF, ($H_2O$, $CO_2$, $CH_2O$, $N_2O$, $NH_3$) optimizing the paths either incurs a performance cost or shows relatively small improvements under 10%. In this case, the improvement in scaling comes at the cost of larger prefactors, and in the case where the number of DoF is small, results in a decrease in performance. For "medium" systems (vinyl, cytosine) we start to see serious computational gains. Below I show comparisons of path optimization for a "small" system Section 6.2.1, and a "medium" system Section 6.2.1:

| Tensor rank | Un-optimized `einsum` (s) | `opt_einsum` (s) | Relative Runtime |
|---|---|---|---|
| 3rd order | 0.000875 | 0.000388 | 2x |
| 4th order | 0.07769 | 0.00730 | 11x |
| 5th order | 5.022 | 0.070 | 74x |
| 6th order | 238.861 | 2.002 | 142x |

Table 6.1: Relative and absolute runtimes of `einsum` vs. `opt_einsum` for a small system (5 electronic surfaces 8 normal modes). Results are for 1 calculation.

| Tensor rank | Un-optimized `einsum` (s) | `opt_einsum` (s) | Relative Runtime |
|---|---|---|---|
| 3rd order | 0.07 | 0.009 | 3x |
| 4th order | 27.364 | 0.142 | 193x |
| 5th order | 1200 | 3.925 | 306x |

Table 6.2: Relative and absolute runtimes of `einsum` vs. `opt_einsum` for a medium system (12 electronic surfaces, 12 normal modes). Results are for 1 calculation.

To provide some in context of performance benefits in practice we can look at Hexahelicene. The linear vibronic model of Hexahelicene, discussed in Section 6.3.9, provides a good example as it is comprised of 63 normal modes and 15 electronic surfaces.

I first will detail the performance including all optimizations. I was able to simulate the spectra with Z3 truncation in roughly 6 hours. This involves about 2800 integration steps, integrated using an explicit Runge–Kutta method of order 5(4) with a $1 \times 10^{-7}$ relative tolerance and $1 \times 10^{-8}$ absolute tolerance. This calculation was performed on 20 cpu cores, using 15GB of memory; running the same calculation with 4 cores takes about 22 hours.

The computational runtime is linear with respect to the length of the propagation, as seen in Table 6.3. In Table 6.3 the first column lists the amount of real-world time it took to compute the necessary EOM to propagate the wavefunction the associated # of femtoseconds in column two. In column three, we can see the real-world computational time it took to propagate 1fs for that particular block of the total simulation. We can see that there is a larger start up time for the initial section of the propagation, but that the majority of the entire simulation takes roughly 15 minutes per one femtosecond of propagation.

Because the runtime scales linearly we can get good estimates of the runtime by calculating only a few integration steps. The runtimes I just mentioned are using the most optimized implementation of `t-amplitudes`. For example, to propagate 28 steps, roughly 0.25fs, takes around 216 seconds with optimized `einsum` code. Using non-optimized `einsum` calls, it can take around 4582 seconds, roughly $21x$ slower.

| real time (s) | simulation time (fs) | real time per 1fs (s/fs) |
|---|---|---|
| 00:00:00 | 0.0 | – |
| 01:22:49 | 2.5557 | 00:32:24 |
| 01:59:57 | 5.0587 | 00:14:50 |
| 02:38:00 | 7.5603 | 00:15:13 |
| 03:14:59 | 10.0519 | 00:14:51 |
| 03:54:40 | 12.5517 | 00:15:52 |
| 04:31:45 | 15.0678 | 00:14:44 |
| 05:10:10 | 17.5518 | 00:15:28 |
| 05:49:52 | 20.0512 | 00:15:53 |
| 06:30:50 | 22.5515 | 00:16:23 |

Table 6.3: Runtime data of Z3 truncated calculation of the ACF of a linear vibronic model of Hexahelicene. Results are reported every 2.5 femtoseconds, or 10% of the total propagation length 25fs.

### 6.2.2 Sparse Matrix Symmetrization

Symmetrization of certain mathematical tensors is also of importance in the VECC method. By not computing all permutations of external labels, we make a trade-off: we can make fewer `einsum` calls, but the resulting tensor needs to be symmetrized as we only calculate the upper/lower triangle (effectively). Note that even though this symmetrization adds computational cost, it adds less than what the respective `einsum` calls would cost.

Instead of using a trivial approach of symmetrizing by permuting over all indices and normalizing, we choose instead to use matrix projection. This procedure is outlined in Section 5.4.1. The procedure has significant performance benefits, as we can reduce the simplistic transpose approach down to two matrix multiplications, which can be computed very efficiently by a BLAS library as opposed to $N!$ additions. To further improve on this, we use sparse matrices to perform the projection.

Below I show comparisons of the basic transposition approach to our sparse matrix approach for a "small" system section 6.2.1, and a "medium" system section 6.2.1:

| Tensor rank | Transpose (s) | Sparse Matrix (s) | Relative Runtime |
|---|---|---|---|
| 4th order | 0.125 | 0.066 | 2x |
| 5th order | 0.853 | 0.072 | 12x |
| 6th order | 10.609 | 0.093 | 114x |

Table 6.4: Relative and absolute runtimes of basic transposition vs. sparse matrix projection for a small system (2 electronic surfaces, 2 normal modes). Results are for 1000 consecutive repetitions.

| Tensor rank | Transpose (s) | Sparse Matrix (s) | Relative Runtime |
|---|---|---|---|
| 4th order | 0.195 | 0.065 | 3x |
| 5th order | 11.9 | 0.544 | 22x |
| 6th order | 1153.4 | 22.68 | 51x |

Table 6.5: Relative and absolute runtimes of basic transposition vs. sparse matrix projection for a medium system (12 electronic surfaces, 12 normal modes). Results are for 1 calculation.

## 6.3 Results

In this section, I present vibrationally-resolved spectra generated using `t-amplitudes`. Relevant technical details will be discussed in Section 6.3.2. I first benchmark my results using the Multi-configuration time-dependent Hartree (MCTDH) software package in Sections 5.3.2 to 5.3.7, and then compare hexahelicene results with a recently published paper.

### 6.3.1 MCTDH calculations

The MCTDH software package is used to generate benchmark ACF results, which are then turned into theoretical spectra through the use of `autospec84`, as described in the section below. These results were generated through wavepacket propagation calculations. Harmonic oscillator primitive basis functions (PBF) were used to describe the vibrational modes. A multi-set single-particle function (SPF) basis was used. The ground electronic state was described with one SPF, and excited electronic states were described with 3–6 SPF's depending on the strength of their associated coupling coefficients. Standard integrator settings were used: `CMF/var`, `BS/spf`, and `SIL/A`. The initial wavefunction was always placed in the lower electronic state. Calculations were run for various propagation lengths, 50, 100, 500, 1000 fs. Convergence in the PBF was determined by comparing MCTDH calculations. For most systems approximately 30 PBF's was sufficient, otherwise 100 to 300 PBF's showed convergence. Results were generated every 0.1fs. An example input file for water $H_2O$ is provided in Listing E.1.

During the calculation ACF data is written to the file `auto`. For every propagation step, in femtoseconds, the real, imaginary and absolute values of the ACF are recorded. This data was then processed by `autospec84`. Note that the MCTDH data did not need to be normalized.

### 6.3.2 Details of generating spectra

In this section I will provide the reader a more detailed explanation of how the final spectra are obtained from the ACF that I generate. The software tool that I used to produce spectra is `autospec`, specifically `autospec84`, provided in the MCTDH software package[59, 60]. A simple example of how to use `autospec84`, for determining the absorption spectrum for the photodissocia- tion of NOC, is provided at[61, Pg. 14-15].

The basic process to generate vibrationally-resolved electronic spectra such as Fig. 6.1 requires the following steps:

1. Calculate ACF using `t-amplitudes`.
2. Normalize $x, y$ data through interpolation.[4]
3. Fourier transform interpolated ACF using `autospec84`.

The ACF data needs to be interpolated and normalized for comparison to MCTDH. The raw ACF has a variable real-time spacing between data points and the values are not normalized. Whereas the MCTDH ACF data has a fixed spacing and the data is normalized. The necessary components to calculate the ACF using `t-amplitudes` are:

- A vibronic model of the form in Equation (3.37), an example is provided in Listing E.2.
- Relevant equations of motion.

As `t-amplitudes` implements a coupled cluster method we can choose various levels of truncation (Singles, Doubles, etc) which change the EOM used to propagate the wavefunction. For most purposes those EOM will already have been pre-generated. If not, then `termfactory` can be used to generate necessary EOM.

**Fourier Transformation** I used `autospec84`, provided in MCTDH, to obtain spectra from results generated by `t-amplitudes` and MCTDH. `autospec84`, quote: *"computes the spectrum by Fourier transform of the autocorrelation function which is multiplied with the weight function"*:

$$\exp\left\{-\left(\frac{t}{\tau}\right)^{\texttt{iexp}}\right\} * \cos^n\left(\frac{\pi * t}{2T}\right) \tag{6.27}$$

The arguments provided to `autospec84` to generate Figure 6.1 are shown below. Similar commands were used to produce the other spectra in this section.

- `-o ./h2o_vibronic_linear_tf50.pl`
- `-f ./ACF_ABS_CC_h2o_vibronic_linear_tf50_z3_normalized.txt`
- `-p 4000`
- `21.0 11.5 eV`
- `30`
- `1`

The command `-p 4000` changes the density of data points plotted. The limits and units of the x-axis of the plot are set using `21.0 11.5 eV`. The value of $\tau$ is set by `30`. The value of `iexp` is set by `1`. The input file path is given by `-f`, and the output file path by `-o`. For the previous commands, the general weight function in Equation (6.27) becomes

$$\exp\left\{-\left(\frac{t}{25}\right)^{1}\right\} * \cos^n\left(\frac{\pi * t}{2T}\right) \tag{6.28}$$

[4]Scipy's interpolate.interp1d was used to perform the interpolation.

For each point on the $x$ axis `autospec84` generates three $y$ values ($g0$, $g1$, $g2$) corresponding three choices of the cosine exponent $N$. In general, $n = 1$ gives the preferred shape, and so all results are plotted using the $g1$ data[61] Note that the magnitudes of calculated spectral peaks are dependent on the choice of $\tau$ in Equation (6.27). By choosing `iexp` to be 1, I am choosing to Fourier transform with a Lorentzian function. Choosing `iexp` to be 2, it would instead be using an exponential function.

### 6.3.3 Benchmark models

Results are presented for five linear vibronic model of molecules, whose Hamiltonians are of the form shown in Equation (3.37). The molecules are water ($H_2O$), carbon dioxide ($CO_2$), formaldehyde ($CH_2O$), nitrous oxide ($N_2O$), and ammonia ($NH_3$). We benchmark the VECC result of `t-amplitudes` against the MCTDH software package as described in the previous section. All spectra are generated from ACFs calculated by propagating the initial wavefunction for 50 femtoseconds. Electronic transition dipole moments are chosen to be 0.1 eV for all electronic surfaces.

We are calculating photo-electron spectra using ionized states calculated by the IP-EOMCC method [62, 63, 64, 65, 66, 67, 68]. These vibronic models were calculated with code developed in the Nooijen group, using similar protocols as described in Chapter 3 [69, 70, 71]. These protocols, as explained with reference to work by Santoro [45], were first developed in [71] and generalized to TD-DFT by Neugebauer and Nooijen [72]. The five vibronic models that are used here were developed by undergraduate student Julia Endicott [73, 74]. In these calculations, the ground state was optimized and the harmonic frequencies were obtained using coupled cluster single-double (CCSD) with a `TZ2P` atomic basis set. Following that, ionization potentials were calculated using subsequent IP-EOMCCSD calculations using the same `TZ2P` basis set. The vibronic models were then constructed in the same fashion as Equations (3.25) and (3.26), including up to quartic constants. Here we only use the linear vibronic models.

### 6.3.4 Water $H_2O$

The parameters for this vibronic model of $H_2O$ are presented in Table 6.6. This model has three electronic surfaces $A = 3$ and three normal modes $N = 3$. We expect this model's spectra to have three bands, around 14eV, 12eV and 18eV, due to the fairly large separation between the three electronic states. The second and third modes $\omega_2, \omega_3$ are almost degenerate, and the largest coefficient is $g_2^{33}$. Consequently we expect the highest energy band to be the most complicated.

In Figure 6.1, we see the VECC and MCTDH spectra exactly agree. Results for `t-amplitude` are generated in 1 to 2 minutes; 52 seconds for Z1 truncation, 82 seconds for Z2 truncation, and 102 seconds for Z3 truncation. In this case, Z1 truncation is sufficient to correctly capture the MCTDH spectra. The theoretical spectra has good agreement with the experimental spectra in Figure 6.1. The magnitude of the theoretical peak slightly below 13eV is smaller than the corresponding experimental peak.

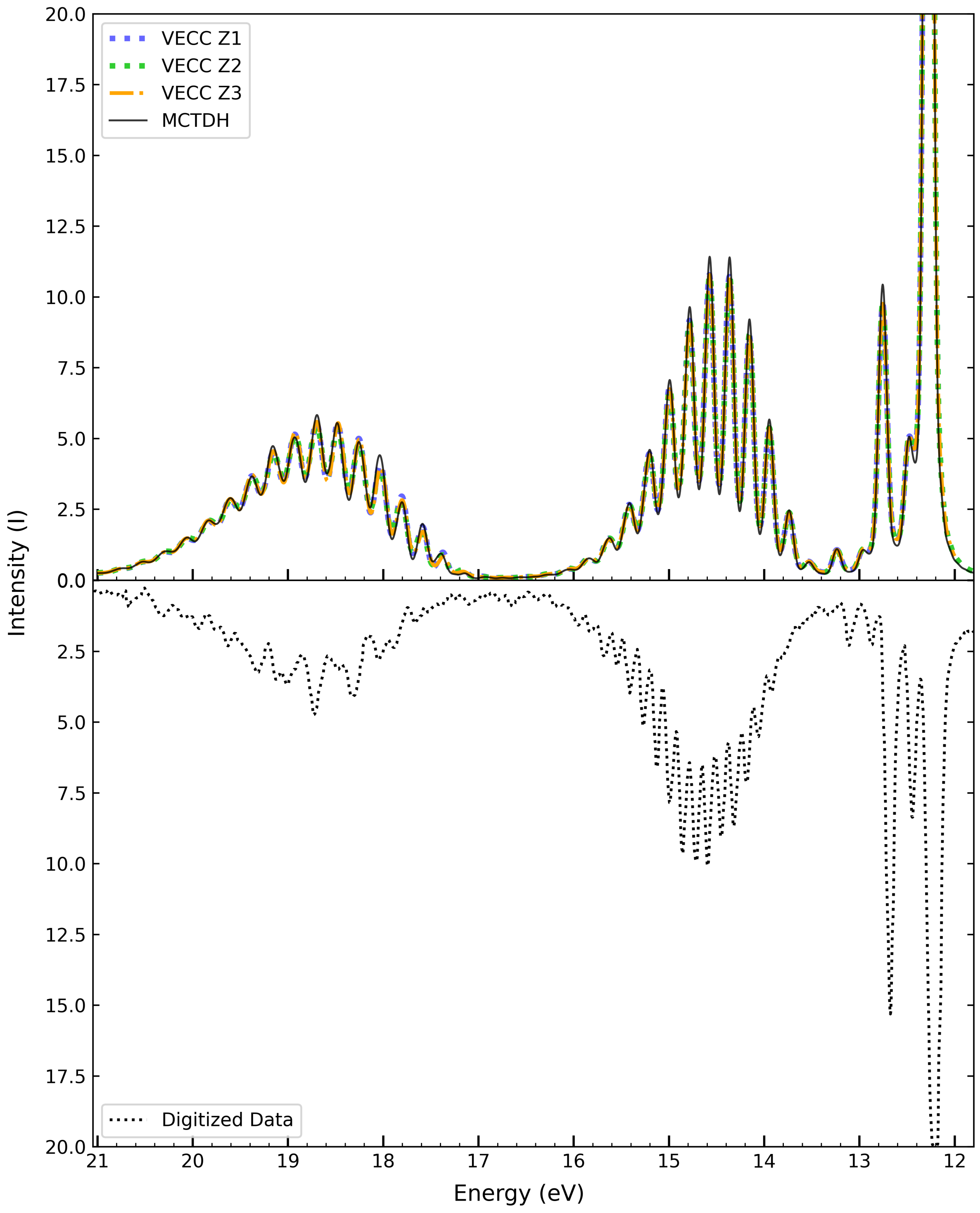

Figure 6.1: Waterfall style plot. Theoretical vibrationally-resolved photo-electron spectra of $H_2O$ is presented in the upper subplot. Digitized data of experimental photo-electron spectrum of $H_2O$ using the He 584(Å) line is presented in the lower subplot. Experimental data reproduced from [75, Fig. 1].

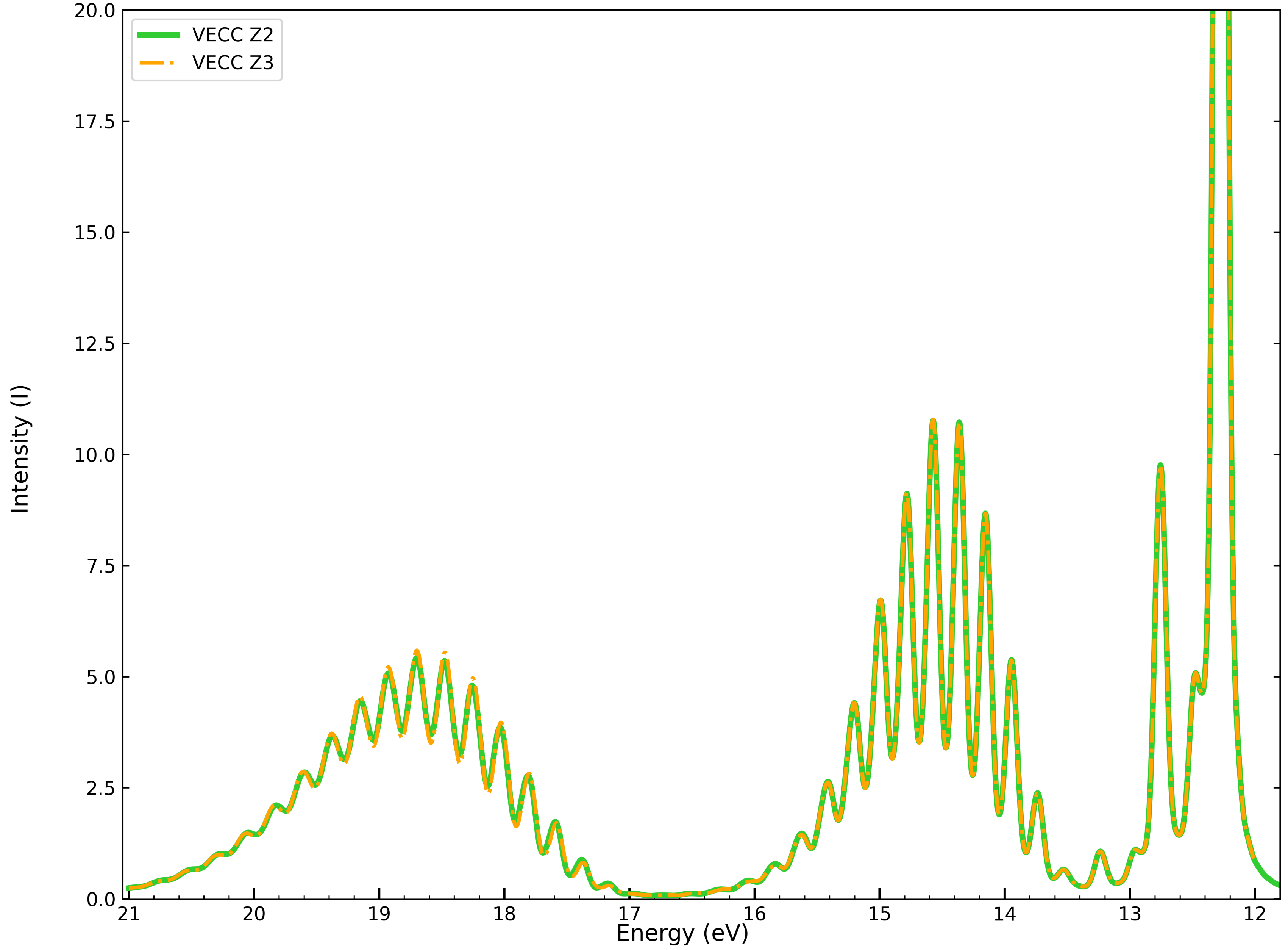

Figure 6.2: Direct comparison of Z2 and Z3 truncation for $H_2O$.

Table 6.6: $H_2O$ model parameters (eV).

| Parameter | | Parameter | | Parameter | |
|---|---|---|---|---|---|
| $E^{11}$ | 14.051 208 | $g_1^{11}$ | 0.650 150 | $g_3^{13}$ | −0.213 821 |
| $E^{22}$ | 11.785 450 | $g_2^{11}$ | 0.302 387 | | |
| $E^{33}$ | 18.280 470 | $g_1^{22}$ | 0.076 394 | | |
| Z.P.E | −0.591 746 | $g_2^{22}$ | 0.291 455 | | |
| $\omega_1$ | 0.208 018 | $g_1^{33}$ | −0.692 915 | | |
| $\omega_2$ | 0.481 209 | $g_2^{33}$ | 0.857 173 | | |
| $\omega_3$ | 0.494 264 | | | | |

### 6.3.5 Carbon Dioxide $CO_2$

The parameters for this vibronic model of $CO_2$ are presented in Table 6.7. This model has six electronic surfaces $A = 6$ and four normal modes $N = 4$. There are two degeneracies in the electron states: $E^{22}$, $E^{33}$ are degenerate, and $E^{55}$, $E^{66}$ are degenerate. We expect this model's spectra to have three bands, around 19eV, 17eV and 13eV. The second and third modes $\omega_2, \omega_3$ are degenerate, and the largest coupling coefficients are $g_4^{25}$, and $g_4^{36}$. There are to coupling coefficients that are exactly degeneracies, and numerous ones which are almost degenerate. We expect the multiple degeneracies around 17eV and the associated coupling strengths to create a complicated band structure in that region.

In Figure 6.3, we see the VECC and MCTDH spectra almost exactly agree. There are two discrepancies around 14eV, and a small difference around 18.5eV. Results take less than 10 minutes to run: 135 seconds for Z1 truncation, 352 seconds for Z2 truncation, and 462 seconds for Z3 truncation. In this case, Z2 truncation is needed to correctly capture the majority of the MCTDH spectra. The error in the Z1 truncation causes most of the spectra to have incorrect energetic placement, shifted higher or lower in energy compared to the MCTDH spectra. Looking at two small peaks around 14–13.5 eV, we can see that Z3 does a better job than Z2 of capturing these less pronounced peaks. The theoretical spectra captures the rough shape of the experimental spectra in the lower subplot of Figure 6.3. Although the theoretical spectra's second band appears to lose intensity sooner than the experimental spectra, around the 17eV region specifically. Comparing magnitudes is not possible.

Table 6.7: $CO_2$ model parameters (eV).

| Parameter | | Parameter | | Parameter | |
|---|---|---|---|---|---|
| $E^{11}$ | 19.027 256 | $g_1^{11}$ | −0.100 514 | $g_2^{12}$ | 0.052 180 |
| $E^{22}$ | 17.590 002 | $g_1^{22}$ | 0.311 746 | $g_3^{13}$ | 0.051 851 |
| $E^{33}$ | 17.590 002 | $g_1^{33}$ | 0.311 746 | $g_4^{14}$ | −0.341 311 |
| $E^{44}$ | 17.744 698 | $g_1^{44}$ | 0.060 581 | $g_4^{25}$ | −0.668 194 |
| $E^{55}$ | 13.315 085 | $g_1^{55}$ | 0.075 499 | $g_2^{45}$ | 0.324 922 |
| $E^{66}$ | 13.315 085 | $g_1^{66}$ | 0.075 499 | $g_4^{36}$ | −0.666 585 |
| Z.P.E | −0.318 398 | | | $g_3^{46}$ | 0.327 569 |
| $\omega_1$ | 0.170 654 | | | | |
| $\omega_2$ | 0.084 656 | | | | |
| $\omega_3$ | 0.084 656 | | | | |
| $\omega_4$ | 0.296 830 | | | | |

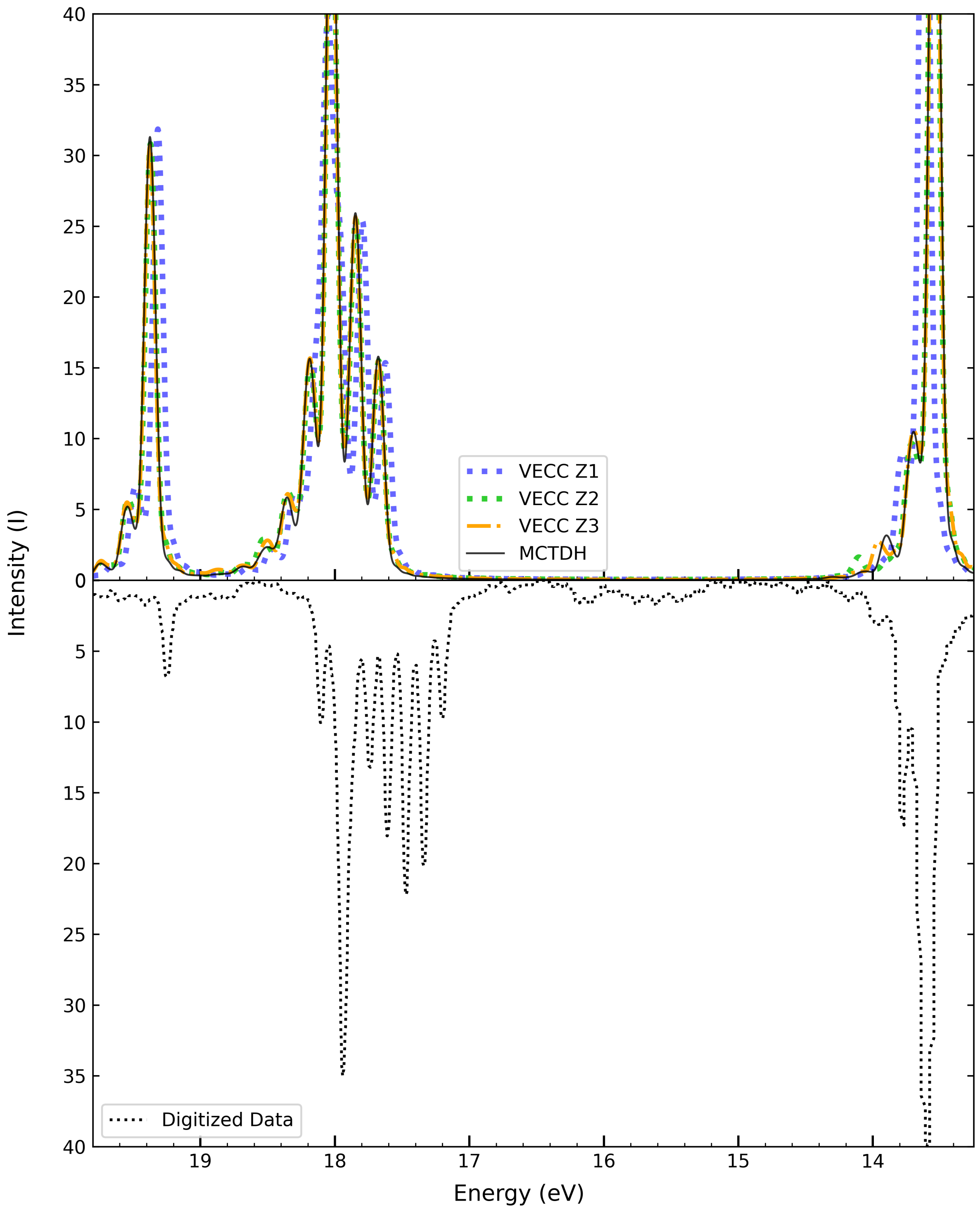

Figure 6.3: Waterfall style plot. Theoretical vibrationally-resolved photo-electron spectra of $CO_2$ is presented in the upper subplot. Digitized data of experimental photo-electron spectrum of $CO_2$ using the He 584(Å) line is presented in the lower subplot. Experimental data reproduced from [76, Fig. 16], individual bands can be found in Figures 17–19.

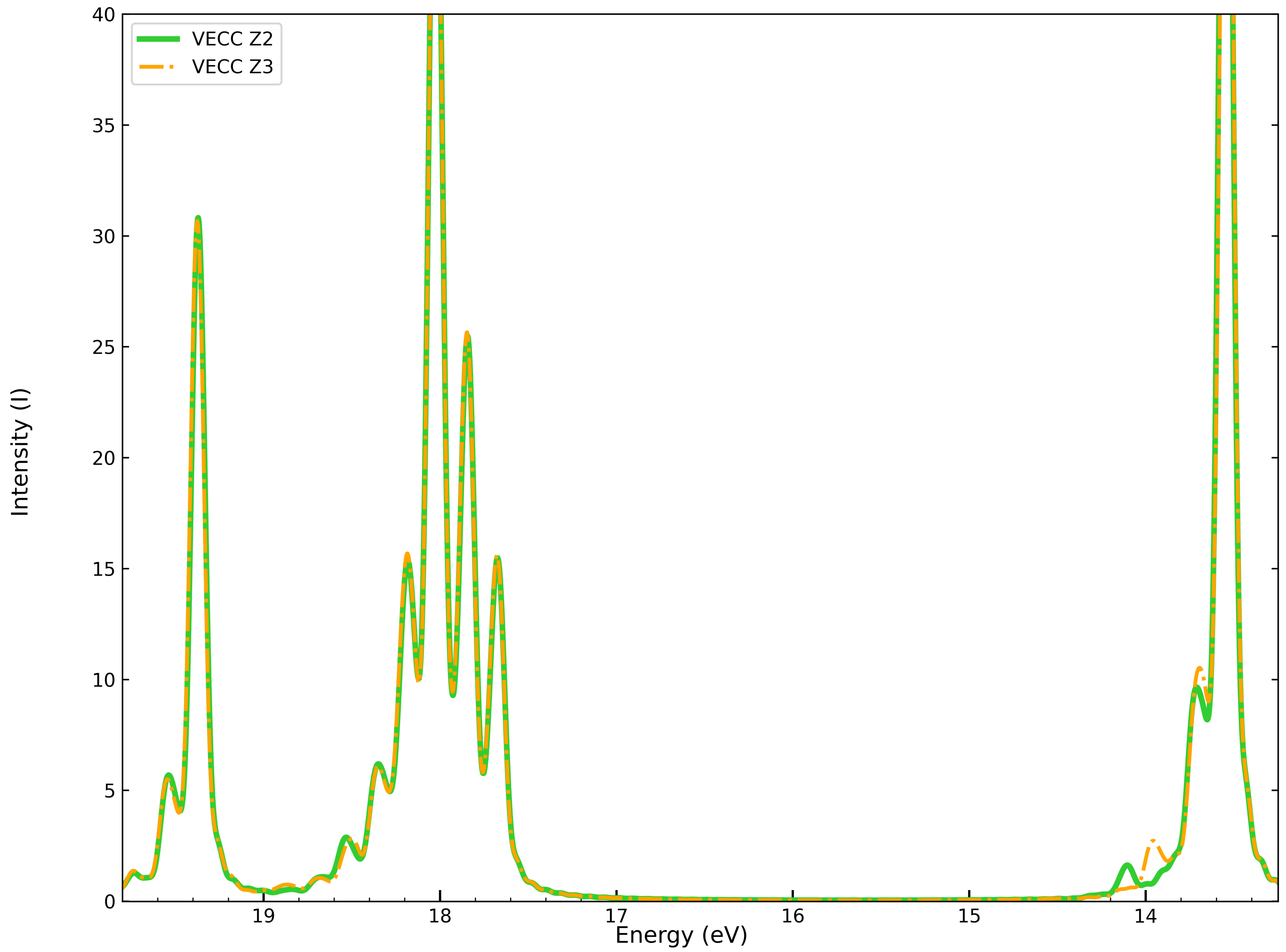

Figure 6.4: Direct comparison of Z2 and Z3 truncation for $CO_2$.

### 6.3.6 Formaldehyde $CH_2O$

The parameters for this vibronic model of $CH_2O$ are presented in Table 6.8. This model has four electronic surfaces $A = 4$ and six normal modes $N = 6$. We expect this model's spectra to have four bands, around 16eV, 15eV, 13eV, and 10eV. No vibrational modes are exactly degenerate, the closest, $\omega_3, \omega_6$, are within 0.01eV. There are no exact degeneracies in the coupling coefficients. The largest coupling term is $g_5^{13}$. We expect a fairly complicated band structure between 17eV and 14eV due to the small spacing between the three highest electronic states.

In Figure 6.5, we see the VECC and MCTDH spectra exactly agree. Results take 1 to 4 minutes: 75 seconds for Z1 truncation, 207 seconds for Z2 truncation, and 250 seconds for Z3 truncation. Both `t-amplitude` results take roughly a minute to run. In this case, Z2 truncation is sufficient to correctly capture the majority MCTDH spectra. The Z1 truncation does a fairly poor job around 18–16 eV, as well as around 11eV.

The theoretical spectra has reasonable agreement with the experimental spectra in Figure 6.5, capturing the general structure of each band. It describes most of the peaks correctly, but fails to reproduce all of the peaks accurately. In the $15 \sim 14$eV band it does reproduce six peaks. But in the 16eV band the theoretical spectra only has four distinct peaks, and there are five distinct peaks present in Figure 6.5.

Table 6.8: $CH_2O$ model parameters (eV).

| Parameter | | Parameter | | Parameter | |
|---|---|---|---|---|---|
| $E^{11}$ | 15.245 549 | $g_1^{11}$ | −0.117 668 | $g_5^{12}$ | 0.144 927 |
| $E^{22}$ | 16.641 110 | $g_2^{11}$ | 0.323 218 | $g_6^{12}$ | 0.194 289 |
| $E^{33}$ | 9.915 591 | $g_3^{11}$ | −0.006 768 | $g_5^{13}$ | −0.509 812 |
| $E^{44}$ | 13.766 519 | $g_1^{22}$ | 0.417 238 | $g_6^{13}$ | −0.150 823 |
| Z.P.E | −0.738 212 | $g_2^{22}$ | 0.022 881 | $g_1^{23}$ | −0.049 255 |
| $\omega_1$ | 0.193 726 | $g_3^{22}$ | 0.461 602 | $g_2^{23}$ | 0.409 333 |
| $\omega_2$ | 0.224 191 | $g_1^{33}$ | −0.072 118 | $g_3^{23}$ | −0.295 391 |
| $\omega_3$ | 0.369 378 | $g_2^{33}$ | −0.006 233 | $g_4^{14}$ | −0.197 230 |
| $\omega_4$ | 0.148 693 | $g_3^{33}$ | 0.051 206 | | |
| $\omega_5$ | 0.161 180 | $g_1^{44}$ | 0.107 009 | | |
| $\omega_6$ | 0.379 255 | $g_2^{44}$ | 0.485 266 | | |
| | | $g_3^{44}$ | −0.064 124 | | |

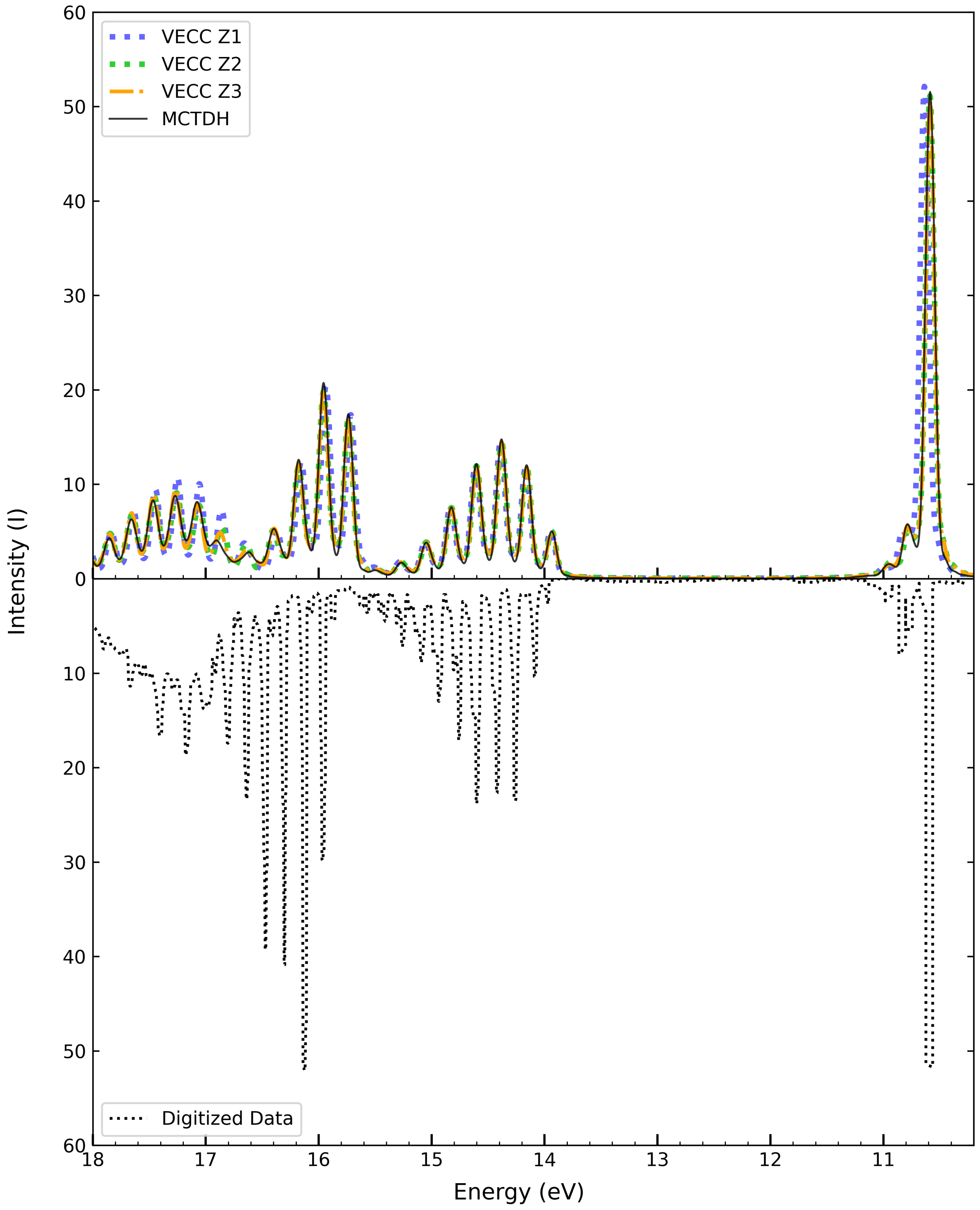

Figure 6.5: Waterfall style plot. Theoretical vibrationally-resolved photo-electron spectra of $CH_2O$ is presented in the upper subplot. Evidence of Z3 contribution making a difference in the spectra. Digitized data of experimental photo-electron spectrum of $CH_2O$ using the He 584(Å) line is presented in the lower subplot. Experimental data reproduced from [77, Fig. 1], individual bands can be found in Figures 2–4.

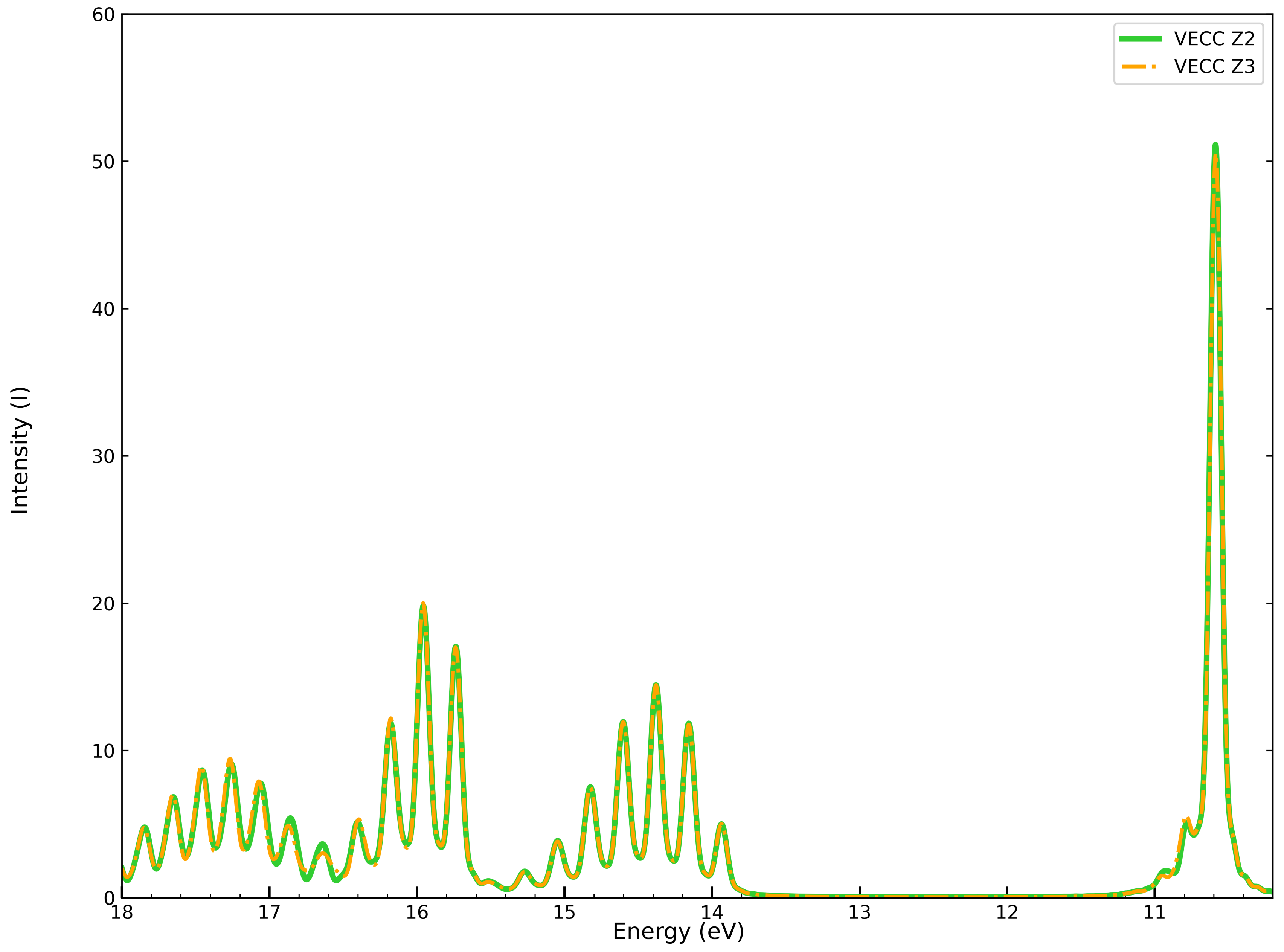

Figure 6.6: Direct comparison of Z2 and Z3 truncation for $CH_2O$.

### 6.3.7 Nitrous Oxide $N_2O$

The parameters for this vibronic model of $N_2O$ are presented in Table 6.9. This model has six electronic surfaces $A = 6$ and four normal modes $N = 4$. There are two degeneracies in the electron states: $E^{33}$, $E^{55}$, are degenerate, and $E^{44}$, $E^{66}$ are degenerate. We expect this model's spectra to have four bands, around 20eV, 18eV, 16eV, and 12eV. The third and fourth modes $\omega_3, \omega_4$ are degenerate, and the strongest coupling coefficients are $g_2^{34}$, and $g_2^{56}$. We expect a fairly complicated band structure between 18eV due to the degenerate electronic states and the fact that the associate coupling coefficients have the greatest magnitude.

The theoretical spectra has reasonable agreement with the experimental spectra in Figure 6.5, capturing the general structure of each band. It describes most of the peaks correctly, but fails to reproduce all of the peaks accurately. In the $15 \sim 14$eV band it does reproduce six peaks. But in the 16eV band the theoretical spectra only has four distinct peaks, and there are 5 peaks present in Figure 6.5.

In Figure 6.7, we see the VECC and MCTDH spectra exactly agree. Results take less than 10 minutes to run: 138 seconds for Z1 truncation, 384 seconds for Z2 truncation, and 461 seconds for Z3 truncation. In this case, Z2 truncation is sufficient to correctly capture the MCTDH spectra. The Z1 truncation again exhibits incorrect energies when compared to the MCTDH spectra. The theoretical spectra has reasonable agreement with the experimental spectra in Figure 6.7, capturing the general spacing of the four bands. However, the 18–19 eV band has a different shape as well as being shift up by $\approx$ 1eV.

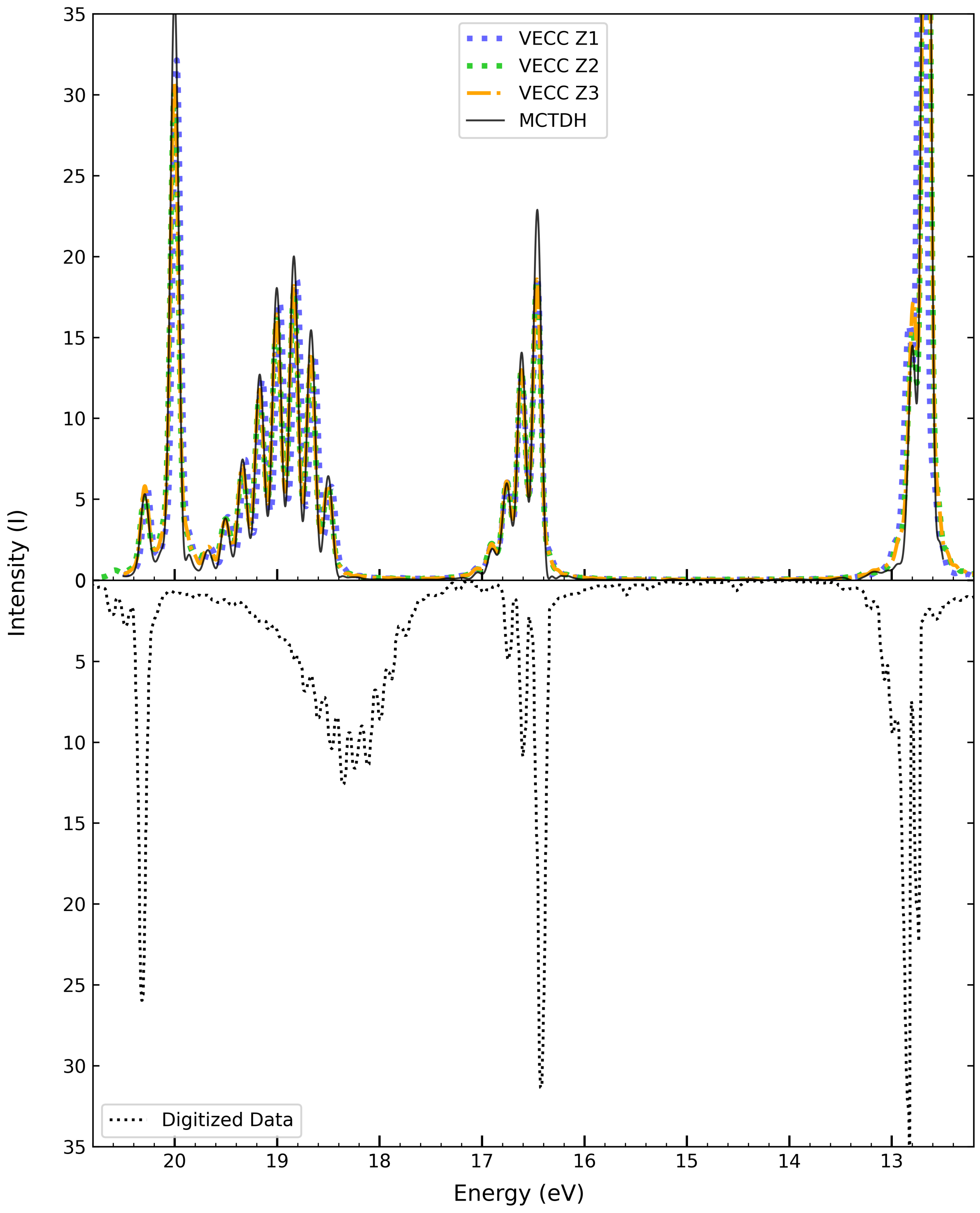

Figure 6.7: Waterfall style plot. Theoretical vibrationally-resolved photo-electron spectra of $N_2O$ is presented in the upper subplot. Digitized data of experimental photo-electron spectrum of $N_2O$ is presented in the lower subplot. Experimental data reproduced from [76, Fig. 1], individual bands can be found in Figures 2–5.

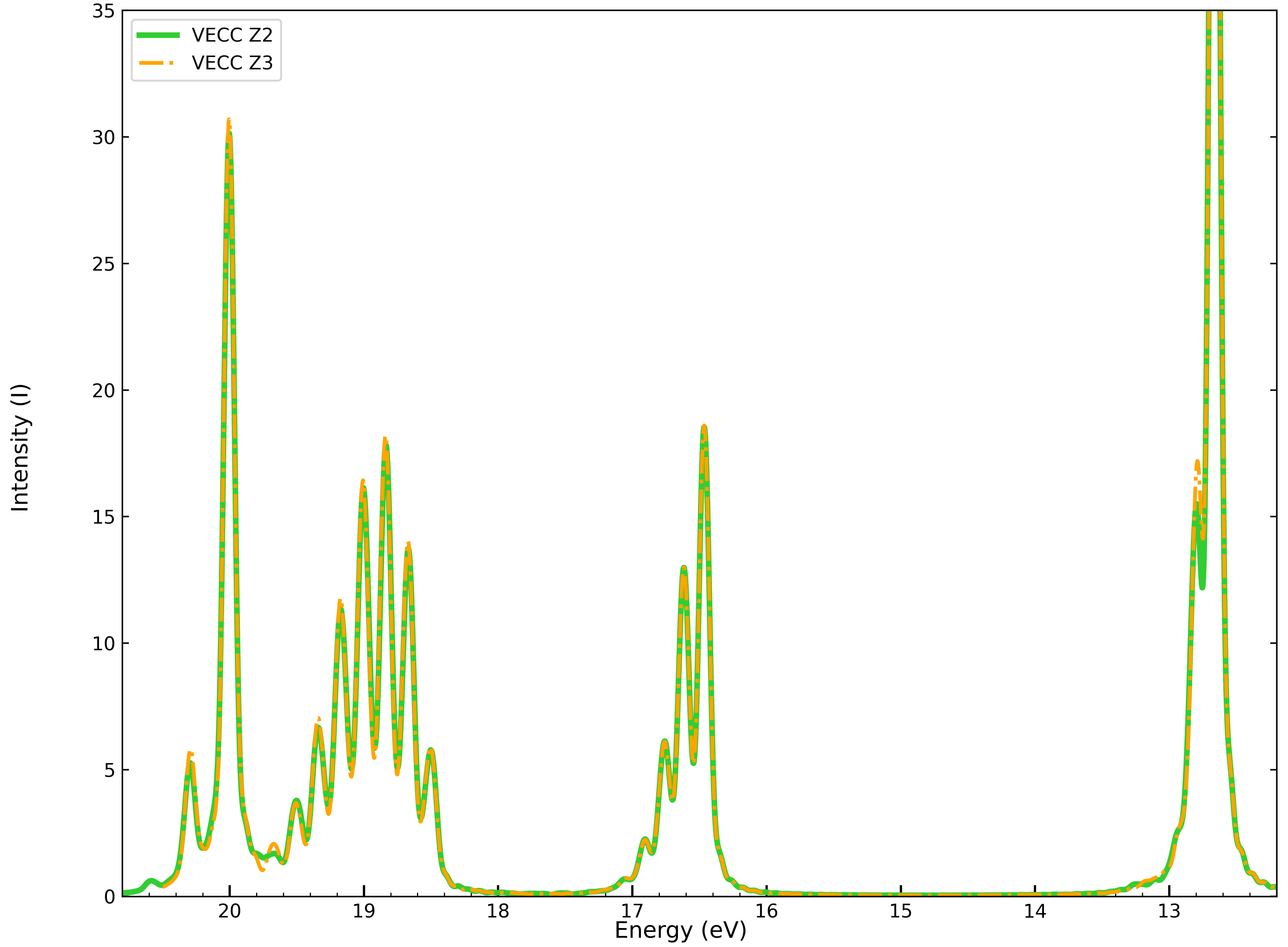

Figure 6.8: Direct comparison of Z2 and Z3 truncation for $N_2O$.

Table 6.9: $N_2O$ model parameters (eV).

| Parameter | | Parameter | | Parameter | |
|---|---|---|---|---|---|
| $E^{11}$ | 19.731 012 | $g_1^{11}$ | 0.010 438 | $g_1^{12}$ | −0.155 567 |
| $E^{22}$ | 16.283 402 | $g_2^{11}$ | 0.170 076 | $g_2^{12}$ | −0.122 708 |
| $E^{33}$ | 18.645 826 | $g_1^{22}$ | −0.162 425 | $g_1^{13}$ | 0.000 088 |
| $E^{44}$ | 12.432 094 | $g_2^{22}$ | −0.123 629 | $g_4^{13}$ | −0.083 328 |
| $E^{55}$ | 18.645 826 | $g_1^{33}$ | 0.354 892 | $g_1^{23}$ | 0.000 388 |
| $E^{66}$ | 12.432 094 | $g_2^{33}$ | −0.221 269 | $g_2^{23}$ | −0.000 085 |
| Z.P.E | −0.298 130 | $g_1^{44}$ | 0.067 501 | $g_4^{23}$ | −0.066 600 |
| $\omega_1$ | 0.159 852 | $g_2^{44}$ | 0.023 286 | $g_1^{14}$ | 0.002 779 |
| $\omega_2$ | 0.285 443 | $g_1^{55}$ | 0.354 892 | $g_2^{14}$ | −0.000 606 |
| $\omega_3$ | 0.075 482 | $g_2^{55}$ | −0.221 269 | $g_4^{14}$ | −0.288 642 |
| $\omega_4$ | 0.075 482 | $g_1^{66}$ | 0.067 501 | $g_1^{24}$ | 0.000 198 |
| | | $g_2^{66}$ | 0.023 286 | $g_4^{24}$ | 0.204 549 |
| | | | | $g_1^{34}$ | 0.267 659 |
| | | | | $g_2^{34}$ | 0.539 477 |
| | | | | $g_3^{15}$ | 0.083 300 |
| | | | | $g_3^{25}$ | 0.066 873 |
| | | | | $g_3^{35}$ | −0.000 003 |
| | | | | $g_3^{16}$ | 0.290 770 |
| | | | | $g_3^{26}$ | −0.204 810 |
| | | | | $g_1^{56}$ | 0.267 225 |
| | | | | $g_2^{56}$ | 0.538 994 |

### 6.3.8 Ammonia $NH_3$

The parameters for this vibronic model of $NH_3$ are presented in Table 6.10. This model has three electronic surfaces $A = 3$ and six normal modes $N = 6$. The electronic states $E^{11}$, $E^{33}$ are degenerate. We expect this model's spectra to have two bands, around 16eV, and 10eV. There are two degeneracies in the modes: $\omega_3, \omega_4$, as well as $\omega_5, \omega_6$. There are many degeneracies among the coupling coefficients: $g_3^{11}$, $g_3^{33}$, $g_4^{13}$, are degenerate, $g_1^{11}$, $g_1^{33}$, are degenerate, and $g_2^{11}$, $g_2^{33}$, are degenerate. Additionally, the coefficients $g_5^{11}$, $g_5^{33}$, $g_6^{13}$, are almost degenerate. The strongest coupling coefficients are the two degenerate groups at 0.573074eV and 0.588249eV.

In Figure 6.9, we see the VECC and MCTDH spectras are slightly different. Results take less than 2 minutes: 37 seconds for Z1 truncation, 55 seconds for Z2 truncation, and 114 seconds for Z3 truncation. All truncation levels can accurately capture the low energy band, however in the high energy band Z1 preforms quite poorly. In this case, Z2 truncation is sufficient to capture the majority of the MCTDH spectra. We do see some deviation between the Z2 and Z3 results. The theoretical spectra has reasonable agreement with the experimental spectra in Figure 6.9, capturing the general structure of both bands. The lower energy band on the other hand is easily described.

Table 6.10: $NH_3$ model parameters (eV).

| Parameter | | Parameter | | Parameter | |
|---|---|---|---|---|---|
| $E^{11}$ | 15.526 843 | $g_1^{11}$ | 0.330 653 | $g_3^{12}$ | −0.252 207 |
| $E^{22}$ | 9.746 612 | $g_2^{11}$ | 0.573 074 | $g_5^{12}$ | 0.108 074 |
| $E^{33}$ | 15.526 843 | $g_3^{11}$ | 0.588 249 | $g_4^{13}$ | −0.588 249 |
| Z.P.E | −0.954 114 | $g_5^{11}$ | 0.332 074 | $g_6^{13}$ | −0.332 078 |
| $\omega_1$ | 0.136 676 | $g_1^{22}$ | −0.501 301 | $g_4^{23}$ | −0.251 602 |
| $\omega_2$ | 0.438 180 | $g_2^{22}$ | 0.154 145 | $g_6^{23}$ | 0.107 915 |
| $\omega_3$ | 0.213 508 | $g_3^{22}$ | 0.000 001 | | |
| $\omega_4$ | 0.213 508 | $g_5^{22}$ | −0.000 004 | | |
| $\omega_5$ | 0.453 178 | $g_1^{33}$ | 0.330 653 | | |
| $\omega_6$ | 0.453 178 | $g_2^{33}$ | 0.573 074 | | |
| | | $g_3^{33}$ | −0.588 248 | | |
| | | $g_5^{33}$ | −0.332 083 | | |

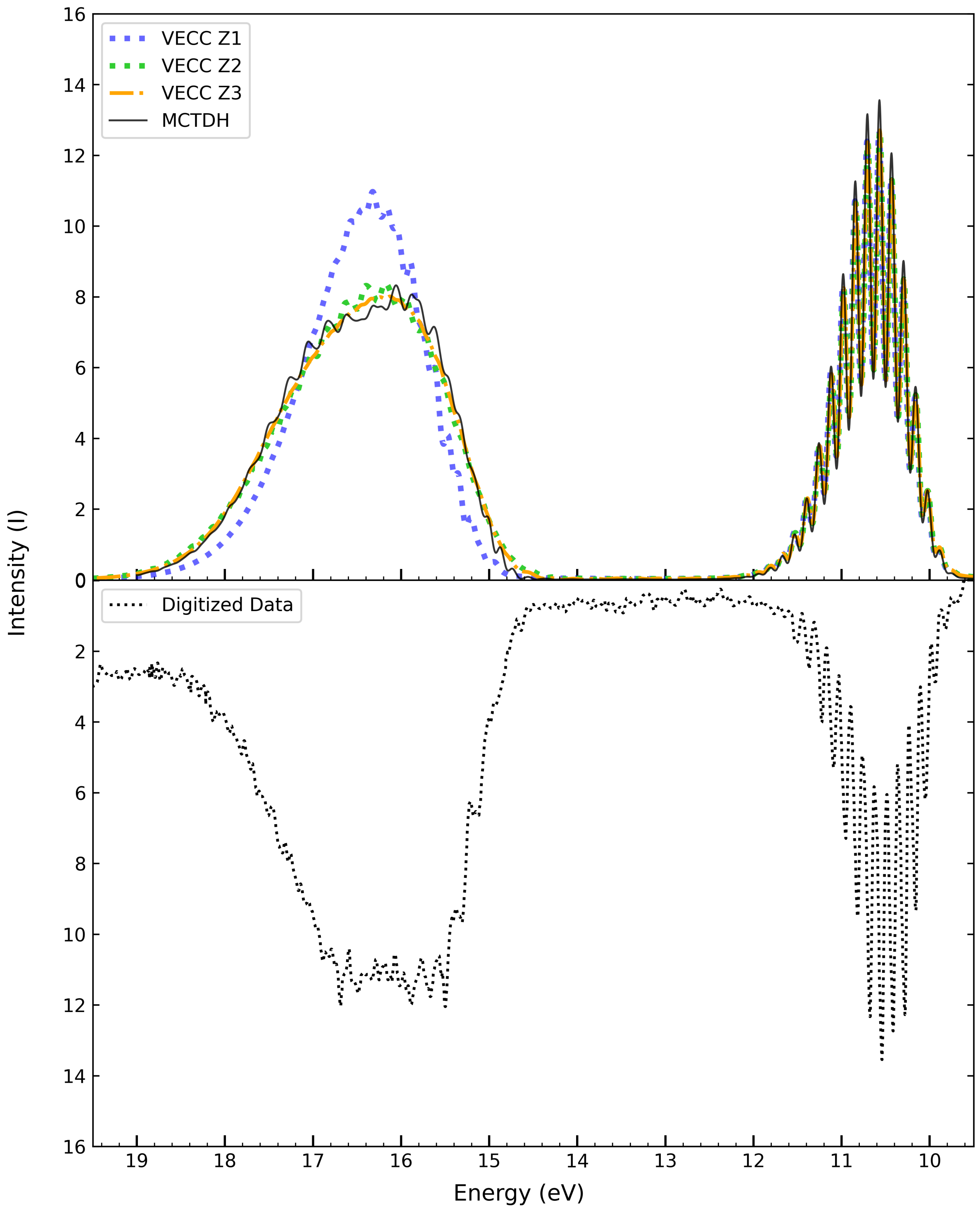

Figure 6.9: Waterfall style plot. Theoretical vibrationally-resolved photo-electron spectra of $NH_3$ is presented in the upper subplot. Digitized data of experimental photo-electron spectrum of $NH_3$, using the He 584(Å) resonance line, is presented in the lower subplot. Experimental data reproduced from [78, Fig. 4].

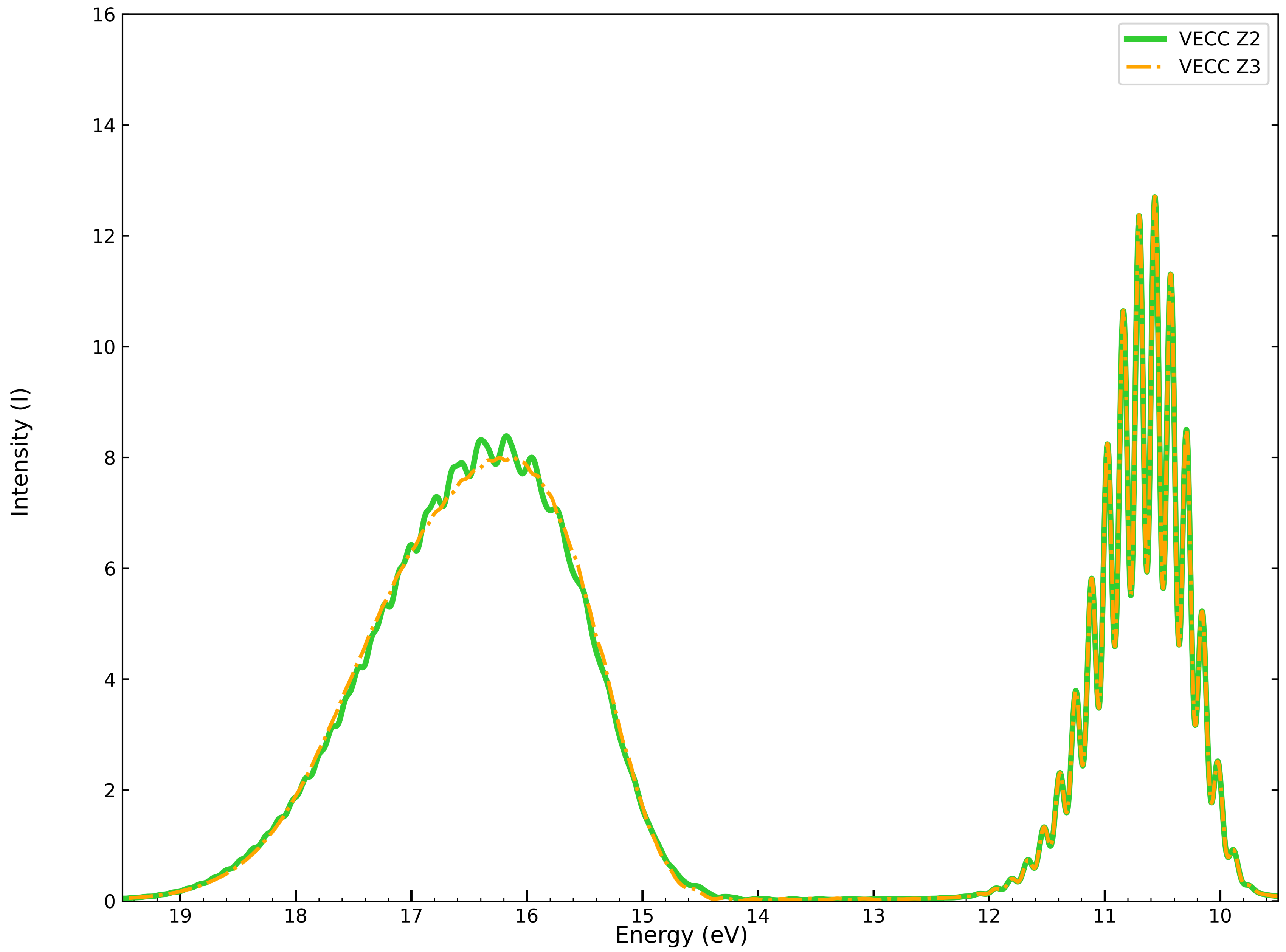

Figure 6.10: Direct comparison of Z2 and Z3 truncation for $NH_3$.

### 6.3.9 Larger Systems

As seen in Sections 6.3.4 and 6.3.8 the VECC method is capable of accurately simulating vibronic spectra. In addition, I have made efforts to test `t-amplitudes` using larger systems. MCTDH results for acroleyin ($C_3H_4O$), cytosine ($C_4H_5N_3O$), furan ($C_4H_4O$), formamide ($CH_3NO$) and vinyl chloride ($C_2H_3Cl$) were calculated. Although at this time VECC results have not been generated for those systems. Instead, in the section below, I will compare results for hexahelicene ($C_{26}H_{16}$) with literature [45].

#### Hexahelicene

The hexahelicene molecule is represented by the linear vibronic model M062X[5], consisting of 63 normal modes and 15 electronic surfaces. This makes it unreasonable to list all the parameters here. The parameters of the M062X model were provided by F. Santoro, as per the relevant paper [45]. Here I compare the electronic absorption spectra generated using VECC to F. Santoro's Multi-layer Multi-configuration time-dependent Hartree (ML-MCTDH) spectra in Figure 6.11. The digitzed experimental spectra is also included.

There is a lower energy band at 3.5–5 eV, and a higher energy band at 5–6 eV. The simulated spectra's vertical energies, intensities, and zero point energy have been shifted in the same manner used in the reference. The peak positions and intensities of the VECC and ML-MCTDH are fairly close in the lower energy band. In this region the VECC method successfully models the vibronic coupling effects. The peak positions of the VECC and ML-MCTDH are consistent in the higher energy band. But there is a discrepancy in the intensities; the VECC's are lower and thus closer to the digitized experimental spectra It appears that the ML-MCTDH overestimates the intensities in the higher energy band

Two levels of truncated VECC results are shown: Singles-Doubles (Z:SD) and Singles-Doubles-Triples (Z:SDT). The SD calculation takes roughly 30 minutes, and the SDT (as discussed in Section 6.2) takes around 6.5 hours. At higher accuracy in the $\hat{Z}$ expansion, we see improvements in overall spectra shape. However, the SD and SDT truncations show only slight differences and so we can conclude that the SD truncation is sufficiently converged.

Compared to the ML-MCTDH approach in [45] the VECC approach has much better agreement with the experimental data in the high energy region. Additionally, the VECC SD calculation is an order of magnitude faster than the ML-MCTDH calculation, which takes roughly 7–8 hours for the M062X model.

[5]The M062X linear vibronic model (LVM) in the reference paper [45] is named after the DFT functional (M062X) used in the diabatization process to generate the model.

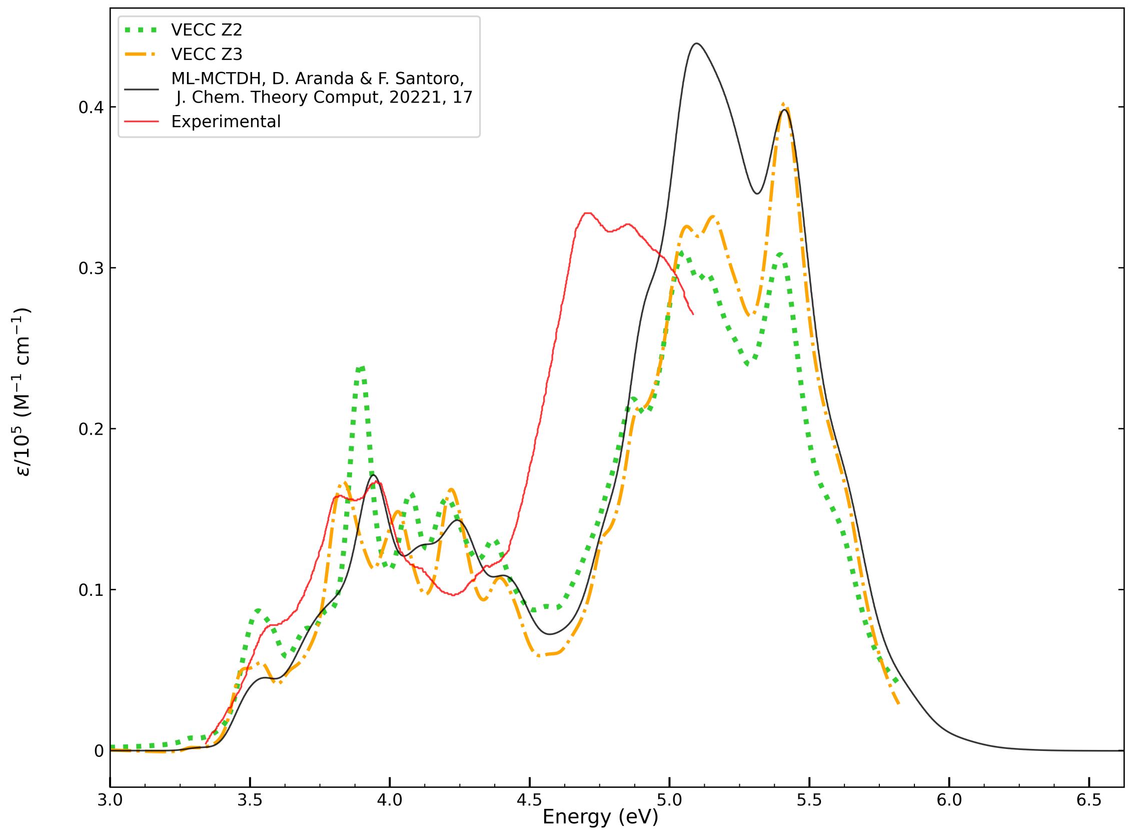

Figure 6.11: Vibrationally-resolved electronic absorption spectra of hexahelicene ($C_{26}H_{16}$).

## 6.4 Conclusion

I showed exact agreement with MCTDH for the small benchmark molecules. For larger molecules like hexahelicene, my results replicate the overall shape of the spectra as presented in the literature. The vibronic models that we are working with may not be of sufficient accuracy and this is motivating current work ongoing in the Nooijen group to collaborate with other researchers to produce better models.

The exponential/linear VECC method with Ehrenfest projection approach has been implemented using `termfactory` and `t-amplitudes`. The approach has been applied to a number of benchmark molecules and shows highly satisfactory results. The favourable scaling properties of the VECC approach results in efficient computations.

In this thesis we limit ourselves to the calculation of spectra, but we note that the code can also be used to calculate time-dependent diabatic state populations. This is of interest to the community and results will be reported elsewhere.

In terms of the software packages as a tool; they have been used successfully to enable research. `t-amplitudes` is well positioned to work with new models when they are produced. There are still further avenues available to improve the runtime.

# Chapter 7

# Conclusions and Final Remarks

Two computational schemes, PIMC and VECC were presented and discussed. in Chapter 4 I showed that by using a PIMC approach, the GMD-reduced scheme, it is possible to circumvent a sign-problem presented by the non-stoquastic nature of the vibronic Hamiltonian. This method was shown to be strongly affected by the proposal distribution. The GMD scheme, which does not trace out all electronic DoF, was shown to be effective for systems with weaker coupling when this non-stoquastic nature was not present. This alternative method appeared to be less dependent on the proposal distribution.

in Chapter 5, I outlined the problem of determining all non-zero contributing pairings for a specific expression (Equation (5.1)) evaluated using Wick's Theorem, and described how the software package `termfactory`, I implemented, solves this problem. I also explored some alternative avenues for increasing the robustness of this package, by shifting away from embedded assumptions, and instead representing constraints as data.

in Chapter 6, we saw the application of the EOM generated by `termfactory` in the calculation of vibrationally-resolved electronic spectra, of vibronic models. We showed exact agreement with MCTDH for small molecules, as well as much improved computational runtimes. For a larger system, hexahelicene, we were able to replicate the overall shape of the spectra as presented in literature. The `t-amplitudes` software package and VECC theory are thus well positioned to investigate other systems of interest.

I described my two computational schemes for investigating vibronic models of nonadiabatic systems. Code for both `termfactory` and `t-amplitudes` are available on GitHub.

## 7.1 Future Work and Ongoing Projects

**Path Integral Monte Carlo (PIMC)** For the PIMC approach I derived an energy estimator in Appendix B. It should be straightforward to implement and obtain results for the model systems. These new results could then be compared to SOS or the previous PIMC results where we employed a re-weighing approach[44].

**Test VECC on additional models** The following additional models were investigated using MCTDH but VECC spectra have not yet been generated.

- Acroleyin ($C_3H_4O$) $A = 7, N = 18$
- Cytosine ($C_4H_5N_3O$) $A = 33, N = 7$
- Furan ($C_4H_4O$) $A = 8, N = 21$
- Formamide ($CH_3NO$) $A = 6, N = 11$
- Vinyl chloride ($C_2H_3Cl$) $A = 6, N = 12$
- Formic acid ($HCO_2H$) $A = 6, N = 9$
- Hydrogen peroxide ($H_2O_2$) $A = 5, N = 6$

This is straightforward in principle and just involves setting up the right scripts on computational servers such as the Advanced Research Computing (ARC) provided by the The Digital Research Alliance of Canada[1]. The Cytosine vibronic model was of particular interest as it was generated using the diabatization procedure described in Section 3.5, as opposed to the benchmark models in Section 6.3.3 which were computed using a different diabatization procedure.

**Vibronic models incorporating Spin-orbit coupling** There has been recent work in the Nooijen group to generate new vibronic models using T. Zeng's spin-orbit coupling diabatization scheme. Ideally this would allow for generating vibronic models for which thermodynamic properties are of interest. These models would provide better foundation for the use of PIMC scheme I described in Chapter 4 than the simple model systems that I have used so far.

**Integer linear programming (ILP)** Personally, I would like to integrate the ILP approach I described into `termfactory`. At this moment I have a proof-of-concept prototype, but it still requires more work before presenting to an end-user. Additionally I would like to further generalize the second stage of the problem, determining pairings.

**Thermal properties using VECC** As previously discussed, with regards to VECC theory, thermal properties can be calculated but require more general and complicated EOM. Extending the `t-amplitudes` software package to support the calculation of thermal properties is of interest. This also requires the extension of `termfactory` to generating more general terms and pairings which include both $f_i$ and $\bar{f}_i$ prefactors, as opposed to the simplified form, as described in Section 5.2.2. I am confident that I know what steps need to be taken to extend `termfactory` in this manner. However, edge cases may prove extremely difficult to handle, as they did with $t^2$ terms. The problem is solvable but requires a careful and methodical approach so as to not get drowned by the complexity.

Below are a few research tangents which were left in an unfinished state or shelved and might be worth future effort.

[1]Formerly known as Compute Canada.

**Iterative Decomposition Scheme** This was an attempt to produce sampling/proposal distributions in a automated manner for a general vibronic model. The general principle was to extract eigenvalues, in manner very similar to singular value decomposition (SVD), and then determine coefficients for new 'fictitious' surfaces using the extracted eigenvalues. An implementation in Julia is available on GitHub. Ultimately we didn't pursue this, as the results were mixed.

**Dense transformation scheme** This was an attempt to make a numerical approach for generating "arbitrary" vibronic models for benchmarking PI approaches. We started with a deliberately crafted Hamiltonian $\hat{H}$, of continuous normal mode co-ordinates $N$ and discrete electronic states $A$, that was diagonal in $A$ and therefore could be evaluated analytically. The approach was to find a transformation to some "dense" representation, meaning that the result was no longer diagonal in $A$ but now had non-zero off-diagonal coupling coefficients. The intent here was that one could produce what appeared to be a challenging vibronic model, which could be freely parameterized, and benchmark the results with ease, due to the nature of already possessing the solution. Ideally the "strength" of the transformation could be controlled by a parameter $A$, such that for $a = 0$ there would be no change, and that for $a = a_{\text{MAX}}$ the matrix would be "maximally" dense. One of the challenges was to design a monotonically increasing transformation. We may have had some minimal success with $2 \times 2$ matrices but for, $3 \times 3$, $4 \times 4$, and larger, things started to break down. Even the notion of $a_{\text{MAX}}$ was not well defined

The Displaced model in Section 4.7.1 is an example of a system which was used to benchmark the PI results and has a parameter which provides control of the vibronic coupling. It was handcrafted to be of use, which was possible due to the small size of the system, but for larger systems this becomes very difficult. With a well-defined transformation, we could investigate the performance for larger systems without the need for expensive SOS calculations.

**PIGS** A significant portion of the initial years of my studies were dedicated to the exploration of Path Integral Ground State (PIGS), specifically to extend the work done in [44] to calculate internal energy. This was waylaid by COVID unfortunately. I think it could still be an interesting avenue to explore, especially if the MH component presented in my thesis could be incorporated into the PIGS approach.

**Distribution Fitting** Another avenue of exploration was the fitting of the PIMC distributions. This was intended to improve the performance of the method by approximating the tails using the slope of the dominant part of the distribution. This was done by evaluating $log\left(\frac{g(R)}{\varrho(R)}\right) = s$ and computing a log binning distribution for $p(s)$, then fitting the binning distribution to a continuous sum of gaussian distributions $\varrho(s)$. We then attempt to numerically integrate $e^s\varrho(s)$, using a scheme such as Gauss-Legendre. Then we calculate

$$Z_f = \int e^s \varrho(s)\,\mathrm{d}s \qquad Z(T_1) = Z_f(T_1)Z_0(T_2) \tag{7.1}$$

Ultimately this approach did not seem promising so it was not pursued further. It is entirely possible that there are techniques I was unaware of, or newly developed techniques which could greatly simplify this problem.

**Application of Machine Learning** This topic is rather large, as one can imagine many ways that machine learning could be used, for example to generate proposal distributions, as well as to do the PI portion, and/or to compute the expectation values. Maybe autoencoders, and/or latent spaces could be a pathway towards a metric of vibronic coupling strength? Training a ML model on vibronic coupling models, and trying to design it to design/predict new ones, to provide a large library of systems to benchmark various tools (VECC, MCTDH, PI) could be useful.

## 7.2 Outstanding questions

Here I would like to briefly outline a collection of thoughts and ideas that could be explored in the future.

**Coupling Metric $\lambda$** To my knowledge there is no formal metric for measuring or assessing how "coupled" a vibronic system is. It is of course clear when a system has no coupling or weak coupling, but the notion of strong coupling seems to be defined as any region where techniques struggle to produce good results. This always bothered me; it felt natural to want to use statements such as

> *We can see that for $\lambda = 0.2$ the system becomes weakly-coupled such that vertical Frank Condon calculations are no longer sufficient but that a CC approach, with a Singles expansion, is sufficient for an accurate description. At $\lambda = 1.2$ the coupling strength increases such that Singles and Doubles are necessary, and for $\lambda \geq 3.5$ the coupling is strong enough that Triples are needed. Finally the strength plateaus around $\lambda \approx 6$.*

A natural comparison would be phase diagrams: plotting (say in the z direction, or as a heatmap); some metric of the coupling strength as a function of certain parameters: geometric configuration, state separation, for example. Consider activation barriers in reaction pathways: there is importance in understanding the different pathways, and the height of activation barriers, and also of the number of steps in a pathway.

Having a standard approach to measure the coupling strength seems intuitively applicable to processing of many different molecules and chemical compounds. Supposing that you have three different methods, each of which performs best within a specific region of coupling strength. Suppose Method 1 is good between $0 \leq \lambda \leq 0.8$, Method 2 between $0.5 \leq \lambda \leq 2.6$ and Method 3 between $4.5 \leq \lambda \leq 7$. You could potentially process a large number of systems, and apply the best Method to each one based on the coupling strength ($\lambda$). This becomes an example of using such a metric in a predicative application for filtering. For systems where

$3 \leq \lambda \leq 4$ you might simply avoid spending time on them, as none of your methods would perform well.

I suspect that realistically one would need a variety of different metrics, that of course would depend on the size and type of the system and therefore become an entire framework requiring maintenance. Metrics suitable for a simple water molecule ($H_2O$) with 3 vibrational modes would very likely not be sufficient to describe a more complex molecule like hexahelicene ($C_{26}H_{16}$) with 64 vibrational modes. There are also different vibronic models and so this would likely lead to certain metrics only being relevant for certain models. For single state systems, you can look at $\frac{\mathrm{d}E}{\mathrm{d}k_{\mathrm{B}}}$, but of course this falls apart for many state systems. Measuring the magnitude of the gradient terms ($\Delta$, $\Delta^2$) in Section 3.2 might be possible, but this seems not worth pursing because it requires computing the integrals that we were trying to avoid all along!

It is possible that such a formal definition/metric does exist and that I simply haven't encountered it; I would be happy to be proven wrong! Overall it just felt like a missing tool in my mathematical toolbox.

**Metric of proposal distribution** It is desirable to have a heuristic to pick a "good" proposal/sampling distribution $\varrho$ for an arbitrary distribution $g$. It seem natural, then, that this would require a metric to measure the "goodness" of the distribution. Of course there is the Kullback–Leibler divergence, which does provide some measure. I feel it would be useful to have a basic protocol of how to approach a generic vibronic model with a small library of "standard" proposal distributions. Being able to heuristically measure their fit and then choose an appropriate distribution would then make for a more complete tool for the PI approach and allow it to be utilized by a broader group of researchers.

**Fourier Transform** I use, the `autospec84` script from the MCTDH software package to produce spectra from an ACF in Chapter 6. However this requires that the ACF has uniform spacing, and therefore that I interpolate the results from the `t-amplitudes` software package. I have some vague idea that there are approaches to transforming variably-spaced functions using the Fourier transform and I was curious if that would have had any significant effect on the spectra that are produced. Additionally my understanding of the various knobs and tuning parameters was only at the basic level and I would have liked to have a better grasp of the underlying nature. From a programmer's perspective, I was annoyed at having the `t-amplitudes` package be tied down to the use of an entirely different software package and would have liked to implement my own Fourier Transform to have a self-contained solution; of course time did not permit this possibility.

# Bibliography

[1] Ken Suzuki et al. “Combinatorial computational chemistry approach to the design of cathode materials for a lithium secondary battery”. In: *Applied surface science* 189.3-4 (2002), pp. 313–318.

[2] Lesley R Rutledge, Seth M McAfee, and Gregory C Welch. “Design and computational characterization of non-fullerene acceptors for use in solution-processable solar cells”. In: *The Journal of Physical Chemistry A* 118.36 (2014), pp. 7939–7951.

[3] M. J. Frisch et al. *Gaussian~16 Revision C.01*. Gaussian Inc. Wallingford CT. 2016.

[4] Malcolm W Chase and National Information Standards Organization (US). *NIST-JANAF thermochemical tables*. Vol. 9. American Chemical Society Washington, DC, 1998, pp. 1–1951. URL: https://webbook.nist.gov/cgi/cbook.cgi?ID=C7727379&Units=CAL&Mask=1#ref-2.

[5] Gilbert. Strang. *Linear algebra and its applications*. 2nd ed. Orlando: Academic, 1980. ISBN: 012673660X.

[6] Marcel Nooijen and Songhao Bao. “Normal ordered exponential approach to thermal properties and time-correlation functions: general theory and simple examples”. In: *Molecular Physics* 119.21-22 (2021), e1980832. DOI: 10.1080/00268976.2021.1980832.

[7] Richard P Feynman. *Statistical mechanics: a set of lectures*. CRC press, 2018.

[8] KE Schmidt and Michael A Lee. “High-accuracy Trotter-formula method for path integrals”. In: *Physical Review E* 51.6 (1995), p. 5495.

[9] Mark Tuckerman. *Statistical mechanics: theory and molecular simulation*. Oxford University Press, 2010, pp. 133–151, 278–281, 284–286.

[10] George Casella Christian P. Robert. *Monte Carlo Statistical Methods*. 2nd ed. 2004. Springer Texts in Statistics. New York, NY: Springer New York, 2004. ISBN: 1-4757-4145-6. DOI: 10.1007/978-1-4757-4145-2. URL: https://doi-org.proxy.lib.uwaterloo.ca/10.1007/978-1-4757-4145-2.

[11] Christian P Robert and George Casella. “The metropolis—Hastings algorithm”. In: *Monte Carlo statistical methods*. Springer, 1999, pp. 231–283. eprint: https://arxiv.org/pdf/1504.01896.pdf.

[12] Jiří Čížek. “On the correlation problem in atomic and molecular systems. Calculation of wavefunction components in Ursell-type expansion using quantum-field theoretical methods”. In: *The Journal of Chemical Physics* 45.11 (1966), pp. 4256–4266.

[13] Fritz Coester and Hermann Kümmel. “Short-range correlations in nuclear wave functions”. In: *Nuclear Physics* 17 (1960), pp. 477–485.

[14] Rodney J Bartlett and George D Purvis. “Many-body perturbation theory, coupled-pair many-electron theory, and the importance of quadruple excitations for the correlation problem”. In: *International Journal of Quantum Chemistry* 14.5 (1978), pp. 561–581.

[15] T Daniel Crawford and Henry F Schaefer III. “An introduction to coupled cluster theory for computational chemists”. In: *Reviews in computational chemistry* 14 (2007), pp. 33–136.

[16] Rodney J Bartlett and Monika Musiał. “Coupled-cluster theory in quantum chemistry”. In: *Reviews of Modern Physics* 79.1 (2007), p. 291.

[17] Frank Neese, Frank Wennmohs, and Andreas Hansen. “Efficient and accurate local approximations to coupled-electron pair approaches: An attempt to revive the pair natural orbital method”. In: *The Journal of chemical physics* 130.11 (2009), p. 114108.

[18] Christoph Riplinger et al. “Natural triple excitations in local coupled cluster calculations with pair natural orbitals”. In: *The Journal of chemical physics* 139.13 (2013), p. 134101.

[19] Martin Schütz and Hans-Joachim Werner. “Low-order scaling local electron correlation methods. IV. Linear scaling local coupled-cluster (LCCSD)”. In: *The Journal of Chemical Physics* 114.2 (2001), pp. 661–681.

[20] Martin Schütz and Hans-Joachim Werner. “Local perturbative triples correction (T) with linear cost scaling”. In: *Chemical Physics Letters* 318.4-5 (2000), pp. 370–378.

[21] Zoltán Rolik et al. “An efficient linear-scaling CCSD (T) method based on local natural orbitals”. In: *The Journal of chemical physics* 139.9 (2013), p. 094105.

[22] Wei Li, Zhigang Ni, and Shuhua Li. “Cluster-in-molecule local correlation method for post-Hartree–Fock calculations of large systems”. In: *Molecular Physics* 114.9 (2016), pp. 1447–1460.

[23] Rodney J Bartlett. “Many-body perturbation theory and coupled cluster theory for electron correlation in molecules”. In: *Annual review of physical chemistry* 32.1 (1981), pp. 359–401.

[24] Marcel Nooijen*, KR Shamasundar, and Debashis Mukherjee. “Reflections on size-extensivity, size-consistency and generalized extensivity in many-body theory”. In: *Molecular Physics* 103.15-16 (2005), pp. 2277–2298.

[25] Dmitry I Lyakh et al. “Multireference nature of chemistry: The coupled-cluster view”. In: *Chemical reviews* 112.1 (2012), pp. 182–243.

[26] Marcel Nooijen. “Many-body similarity transformations generated by normal ordered exponential excitation operators”. In: *The Journal of chemical physics* 104.7 (1996), pp. 2638–2651.

[27] Jacob A Faucheaux and So Hirata. “Higher-order diagrammatic vibrational coupled-cluster theory”. In: *The Journal of chemical physics* 143.13 (2015), p. 134105.

[28] Jacob A Faucheaux, Marcel Nooijen, and So Hirata. "Similarity-transformed equation-of-motion vibrational coupled-cluster theory". In: *The Journal of chemical physics* 148.5 (2018), p. 054104.

[29] Subrata Banik, Sourav Pal, and M Durga Prasad. "Calculation of vibrational energy of molecule using coupled cluster linear response theory in bosonic representation: Convergence studies". In: *The Journal of chemical physics* 129.13 (2008), p. 134111.

[30] Curtis L Janssen and Henry F Schaefer. "The automated solution of second quantization equations with applications to the coupled cluster approach". In: *Theoretica chimica acta* 79.1 (1991), pp. 1–42.

[31] So Hirata et al. "Combined coupled-cluster and many-body perturbation theories". In: *The Journal of chemical physics* 121.24 (2004), pp. 12197–12207.

[32] "An Updated Set of Basic Linear Algebra Subprograms (BLAS)". In: *ACM Trans. Math. Softw.* 28.2 (June 2002), pp. 135–151. ISSN: 0098-3500. DOI: 10.1145/567806.567807. URL: https://doi.org/10.1145/567806.567807.

[33] E. Anderson et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8 (paperback).

[34] *openBLAS*. Version 0.3.20. Feb. 20, 2022. URL: http://www.openblas.net/.

[35] Qian Wang et al. "AUGEM: automatically generate high performance dense linear algebra kernels on x86 CPUs". In: *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE. Denver CO, 2013, pp. 1–12.

[36] Zhang Xianyi, Wang Qian, and Zhang Yunquan. "Model-driven level 3 BLAS performance optimization on Loongson 3A processor". In: *2012 IEEE 18th international Conference on Parallel and Distributed Systems (ICPADS)*. IEEE. Dec. 2012, pp. 684–691.

[37] *IntelMKL*. Version 2019.5.138. 2019. URL: https://www.intel.com/content/www/us/en/develop/documentation/onemkl-linux-developer-guide/top.html.

[38] Endong Wang et al. "Intel math kernel library". In: *High-Performance Computing on the Intel® Xeon Phi™*. Springer, 2014, pp. 167–188.

[39] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[40] Jake Fisher. *CHEM 340 Course Notes and Lectures*. Jan. 2021.

[41] Sebastian Mai and Leticia González. "Molecular Photochemistry: Recent Developments in Theory". In: *Angewandte Chemie International Edition* 59.39 (2020), pp. 16832–16846. DOI: https://doi.org/10.1002/anie.201916381. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.201916381.

[42] David R Yarkony et al. "Diabatic and adiabatic representations: Electronic structure caveats". In: *Computational and Theoretical Chemistry* 1152 (2019), pp. 41–52.

[43] Michael Baer. *Beyond Born-Oppenheimer: Conical intersections and Electronic nonadiabatic coupling terms*. Wiley New York, 2006.

[44] Neil Raymond et al. "A path integral methodology for obtaining thermodynamic properties of nonadiabatic systems using Gaussian mixture distributions". In: *The Journal of Chemical Physics* 148.19 (2018), p. 194110. DOI: 10.1063/1.5025058. URL: https://doi.org/10.1063/1.5025058.

[45] Daniel Aranda and Fabrizio Santoro. "Vibronic spectra of $\pi$-conjugated systems with a multitude of coupled states: A protocol based on linear vibronic coupling models and quantum dynamics tested on hexahelicene". In: *Journal of Chemical Theory and Computation* 17.3 (2021), pp. 1691–1700.

[46] Sergey Bravyi et al. "The complexity of stoquastic local Hamiltonian problems". In: *arXiv preprint quant-ph/0606140* (2006).

[47] Milad Marvian, Daniel A Lidar, and Itay Hen. "On the computational complexity of curing non-stoquastic Hamiltonians". In: *Nature communications* 10.1 (2019), pp. 1–9.

[48] Tao Zeng et al. "MoRiBS-PIMC: A program to simulate molecular rotors in bosonic solvents using path-integral Monte Carlo". In: *Computer Physics Communications* 204 (2016), pp. 170–188.

[49] Neil Raymond. *termfactory*. Version 1.0.0. June 22, 2022. URL: https://github.com/ngraymon/termfactory.

[50] Neil Raymond and Songhao Bao. *t-amplitudes*. Version 1.0.0. July 20, 2022.

[51] Jan Wielemaker et al. "SWI-Prolog". In: *Theory and Practice of Logic Programming* 12.1-2 (2012), pp. 67–96. ISSN: 1471-0684.

[52] Manuel Clavel et al. *All About Maude-A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic*. Vol. 4350. Springer, 2007.

[53] Manuel Clavel et al. *Maude Manual*. Version 3.1. Springer, 2020.

[54] *Maude Overview*. URL: http://maude.cs.illinois.edu/w/index.php?title=Maude_Overview&oldid=14.

[55] Ylloh. *Linear optimization in a 2-dimensional polytope*. 2015. URL: https://en.wikipedia.org/wiki/File:Linear_optimization_in_a_2-dimensional_polytope.svg.

[56] George B. Dantzig. "Origins of the Simplex Method". In: *A History of Scientific Computing*. New York, NY, USA: Association for Computing Machinery, 1990, pp. 141–151. ISBN: 0201508141. URL: https://doi.org/10.1145/87252.88081.

[57] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[58] Daniel G. a. Smith and Johnnie Gray. "opt_einsum - A Python package for optimizing contraction order for einsum-like expressions". In: *Journal of Open Source Software* 3.26 (2018), p. 753. DOI: 10.21105/joss.00753. eprint: https://doi.org/10.21105/joss.00753. URL: https://github.com/dgasmith/opt_einsum.

[59] G. A. Worth et al. The MCTDH Package, Version 8.2, (2000). H.-D. Meyer, Version 8.3 (2002), Version 8.4 (2007). Used version: 8.4.23 (May 2022). See http://mctdh.uni-hd.de. University of Heidelberg, Germany.

[60] Apache Software Foundation. *autospec84*. Version 8.4.21. June 24, 2021. URL: https://www.pci.uni-heidelberg.de/tc/usr/mctdh/doc/analyse/analyse_docu.html#autospec.

[61] G. A. Worth et al. *The Heidelberg MCTDH Package: A set of programs for multi-dimensional quantum dynamics. User's Guide*. English. Version Version 8.5.16. 209 pp. published.

[62] Yannick J Bomble et al. "Equation-of-motion coupled-cluster methods for ionized states with an approximate treatment of triple excitations". In: *The Journal of chemical physics* 122.15 (2005), p. 154107.

[63] Monika Musia and Rodney J Bartlett. "EOM-CCSDT study of the low-lying ionization potentials of ethylene, acetylene and formaldehyde". In: *Chemical physics letters* 384.4-6 (2004), pp. 210–214.

[64] Monika Musial and Rodney J Bartlett. "Equation-of-motion coupled cluster method with full inclusion of connected triple excitations for electron-attached states: EA-EOM-CCSDT". In: *Journal of Chemical Physics* 119.4 (2003), pp. 1901–1908.

[65] John F Stanton and Jürgen Gauss. "Analytic energy derivatives for ionized states described by the equation-of-motion coupled cluster method". In: *The Journal of chemical physics* 101.10 (1994), pp. 8938–8944.

[66] Marcel Nooijen and Jaap G Snijders. "Coupled cluster Green's function method: Working equations and applications". In: *International journal of quantum chemistry* 48.1 (1993), pp. 15–48.

[67] Marcel Nooijen and Jaap G Snijders. "Coupled cluster approach to the single-particle Green's function". In: *International Journal of Quantum Chemistry* 44.S26 (1992), pp. 55–83.

[68] RJ Bartlett and JF Stanton. "Reviews in computational chemistry". In: *Inc. New York* (1994), p. 65.

[69] Anirban Hazra and Marcel Nooijen. "Vibronic coupling in the excited cationic states of ethylene: Simulation of the photoelectron spectrum between 12 and 18 eV". In: *The Journal of chemical physics* 122.20 (2005), p. 204327.

[70] Anirban Hazra and Marcel Nooijen. "Comparison of various Franck–Condon and vibronic coupling approaches for simulating electronic spectra: The case of the lowest photoelectron band of ethylene". In: *Physical Chemistry Chemical Physics* 7.8 (2005), pp. 1759–1771.

[71] Marcel Nooijen. "First-principles simulation of the UV absorption spectrum of ketene". In: *International journal of quantum chemistry* 95.6 (2003), pp. 768–783.

[72] Johannes Neugebauer, Evert Jan Baerends, and Marcel Nooijen. "Vibronic coupling and double excitations in linear response time-dependent density functional calculations: Dipole-allowed states of N 2". In: *The Journal of chemical physics* 121.13 (2004), pp. 6155–6166.

[73] Marcel Nooijen. *Vibronic model parameters, calculated by IP-EOMCC method, for generating photo-electron spectra.* 2011. URL: http://scienide2.uwaterloo.ca/~nooijen/website_new_20_10_2011/vibron/VC/models.html (visited on 08/08/2022).

[74] Julia Endicott. "Generating Vibronic Coupling Models and Simulating Photoelectron Spectra". CHEM 494 project report. Dept. of Chemistry, Faculty of Science, University of Waterloo, 2014. URL: http://scienide2.uwaterloo.ca/~nooijen/website_new_20_10_2011/vibron/VC/julia_thesis.pdf (visited on 08/08/2022).

[75] CR Brundle and David Warren Turner. "High resolution molecular photoelectron spectroscopy II. Water and deuterium oxide". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 307.1488 (1968), pp. 27–36.

[76] CR Brundle and DW Turner. "Studies on the photoionisation of the linear triatomic molecules: N2O, COS, CS2 and CO2 using high-resolution photoelectron spectroscopy". In: *International Journal of Mass Spectrometry and Ion Physics* 2.3 (1969), pp. 195–220.

[77] Baohua Niu et al. "High-resolution He I$\alpha$ photoelectron spectroscopy of H2CO and D2CO using supersonic molecular beams". In: *Chemical physics letters* 201.1-4 (1993), pp. 212–216.

[78] GR Branton et al. "A Discussion on photoelectron spectroscopy-Photoelectron spectra of some polyatomic molecules". In: *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 268.1184 (1970), pp. 77–85.

[79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley, 1979. ISBN: 8178083477.

[80] D. Wood. *Theory of Computation.* John Wiley and Sons., 1987. ISBN: 0471603511.

[81] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations. The IBM Research Symposia Series.* Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9. URL: https://doi.org/10.1007/978-1-4684-2001-2_9.

[82] L.G. Khachiyan. "A Polynomial Algorithm in Linear Programming (english translation)". In: *Doklady Akademiia Nau SSSR.* Vol. 20. 1979, pp. 191–194. DOI: https://doi.org/10.1016/0041-5553(80)90061-0.

[83] V. Klee and G. Minty. "How good is the simplex algorithm?" In: *Inequalities III (Proceedings of the Third Symposium on Inequalities)*. Los Angeles: Academic Press., 1969, pp. 159–175.

[84] Hendrik W Lenstra Jr. “Integer programming with a fixed number of variables”. In: *Mathematics of operations research* 8.4 (1983), pp. 538–548.

[85] Daniel Lokshtanov. “New methods in Paramterized Algorithms and Complexity”. PhD thesis. Bergen, Norway: University of Bergen., 2009.

[86] John Fearnley and Rahul Savani. “The Complexity of the Simplex Method”. In: *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*. STOC ’15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 201–208. ISBN: 9781450335362. URL: https://doi.org/10.1145/2746539.2746558.

[87] Andrew M Childs et al. “Theory of trotter error with commutator scaling”. In: *Physical Review X* 11.1 (2021), p. 011020.

[88] Michel X Goemans and Thomas Rothvoß. “Polynomiality for bin packing with a constant number of item types”. In: *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2014, pp. 830–839. DOI: 10.1145/3421750.

[89] WH Press et al. “Numerical Recipes: The Art of Scientific Computing”. In: *Technometrics* (2007).

[90] *Numerical recipes : the art of scientific computing*. 3rd ed. Cambridge, UK ; Cambridge University Press, 2007. ISBN: 9780521880688.

[91] Gilbert Strang. *Linear algebra and its applications*. 4th ed. Belmont, CA: Thomson, Brooks/Cole, 2006.

[92] Guangyao Zhou. “Mixed Hamiltonian Monte Carlo for mixed discrete and continuous variables”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17094–17104.

[93] Maria I Gorinova et al. “Conditional independence by typing”. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 44.1 (2021), pp. 1–54.

[94] Guangyao Zhou et al. “PGMax: Factor Graphs for Discrete Probabilistic Graphical Models and Loopy Belief Propagation in JAX”. In: *arXiv preprint arXiv:2202.04110* (2022).

[95] Jacob Kelly and Will Sussman Grathwohl. “No Conditional Models for me: Training Joint EBMs on Mixed Continuous and Discrete Data”. In: *Energy Based Models Workshop-ICLR 2021*. 2021.

[96] I. Prigogine and S.A. Rice. *Advances in Chemical Physics*. Advances in Chemical Physics v. 121. Wiley, 2003. ISBN: 9780471619673. URL: https://books.google.ca/books?id=NCOOFAm7IxoC.

[97] Andrew Gelman and Xiao-Li Meng. “Simulating normalizing constants: From importance sampling to bridge sampling to path sampling”. In: *Statistical science* (1998), pp. 169–170.

[98] Nandini Ananth and Thomas F Miller III. “Exact quantum statistics for electronically nonadiabatic systems using continuous path variables”. In: *The Journal of chemical physics* 133.23 (2010), p. 234103.

[99] H Chang. “From electronic structure theory to molecular spectroscopy”. In: *BA degree Thesis, Princeton University* (2003).

[100] Prateek Goel. “First Principles Simulations of Vibrationally Resolved Photodetachment Spectra of Select Biradicals”. MA thesis. University of Waterloo, 2012. URL: https://uwspace.uwaterloo.ca/handle/10012/6985.

[101] Julia Endicott. “Generating Vibronic Coupling Models and Simulating Photoelectron Spectra”. 2012. URL: http://scienide2.uwaterloo.ca/~nooijen/website_new_20_10_2011/vibron/VC/julia_thesis.pdf.

[102] Wolfgang Domcke and David R Yarkony. “Role of conical intersections in molecular spectroscopy and photoinduced chemical dynamics”. In: *Annual review of physical chemistry* 63 (2012), pp. 325–352.

[103] Michael Baer. “Introduction to the theory of electronic non-adiabatic coupling terms in molecular systems”. In: *Physics reports* 358.2 (2002), pp. 75–142.

[104] David R Yarkony. “Nonadiabatic Quantum Chemistry Past, Present, and Future”. In: *Chemical reviews* 112.1 (2011), pp. 481–498.

[105] Shaohong L Li et al. “Nonintuitive Diabatic Potential Energy Surfaces for Thioanisole”. In: *The journal of physical chemistry letters* 6.17 (2015), pp. 3352–3359.

[106] A Devaquet, A Sevin, and B Bigot. “Avoided crossings in excited states potential energy surfaces”. In: *Journal of the American Chemical Society* 100.7 (1978), pp. 2009–2011.

[107] Alexei Stuchebrukhov. “Tunneling Time and the Breakdown of Born–Oppenheimer Approximation”. In: *The Journal of Physical Chemistry B* (2015).

[108] John C Tully. “Perspective: Nonadiabatic dynamics theory”. In: *The Journal of chemical physics* 137.22 (2012), 22A301.

[109] Charles P Enz et al. *[Lectures on physics]; Pauli lectures on physics. 5. Wave mechanics.* Vol. 1. Courier Corporation, 2000, pp. 180–182.

[110] Takeshi M Yamamoto. “Path-integral virial estimator based on the scaling of fluctuation coordinates: Application to quantum clusters with fourth-order propagators”. In: *The Journal of chemical physics* 123.10 (2005), p. 104101.

[111] Troy Van Voorhis et al. “The diabatic picture of electron transfer, reaction barriers, and molecular dynamics”. In: *Annual review of physical chemistry* 61 (2010), pp. 149–170.

[112] Maria Fumanal, Etienne Gindensperger, and Chantal Daniel. “Ultrafast Intersystem Crossing vs Internal Conversion in alpha-Diimine Transition Metal Complexes: Quantum Evidence”. In: *The Journal of Physical Chemistry Letters* 9.17 (2018). PMID: 30145893, pp. 5189–5195. DOI: 10.1021/acs.jpclett.8b02319. URL: https://doi.org/10.1021/acs.jpclett.8b02319.

[113] Konstantin Karandashev and Jiří Vaníček. “Accelerating quantum instanton calculations of the kinetic isotope effects”. In: *The Journal of chemical physics* 143.19 (2015), p. 194104.

[114] Michele Ceriotti et al. "The inefficiency of re-weighted sampling and the curse of system size in high-order path integration". In: *Proc. R. Soc. A*. The Royal Society. 2011, rspa20110413.

[115] Amelia Zutz and David J Nesbitt. "Nonadiabatic Spin–Orbit Excitation Dynamics in Quantum-State-Resolved NO($^2\Pi_{1/2}$) Scattering at the Gas–Room Temperature Ionic Liquid Interface". In: *The Journal of Physical Chemistry C* 119.16 (2015), pp. 8596–8607.

[116] K Mohamed Ali and BWC Sathiyasekaran. "Computer professionals and carpal tunnel syndrome (CTS)". In: *International Journal of Occupational Safety and Ergonomics* 12.3 (2006), pp. 319–325.

[117] Johan Hviid Andersen et al. "Computer mouse use predicts acute pain but not prolonged or chronic pain in the neck and shoulder". In: *Occupational and environmental medicine* 65.2 (2008), pp. 126–131.

[118] Clayton Blehm et al. "Computer vision syndrome: a review". In: *Survey of ophthalmology* 50.3 (2005), pp. 253–262.

[119] R.E. Bellman. *Dynamic Programming*. Dover Books on Computer Science Series. Dover Publications, 2003. ISBN: 9780486428093. URL: https://books.google.ca/books?id=fyVtp3EMxasC.

[120] S Saddique and GA Worth. "Applying the vibronic coupling model Hamiltonian to the photoelectron spectrum of cyclobutadiene". In: *Chemical physics* 329.1 (2006), pp. 99–108.

[121] EN Miranda. "A paradox in the electronic partition function or how to be cautious with mathematics". In: *European Journal of Physics* 22.5 (2001), p. 483.

[122] Donald G Truhlar. "Potential energy surfaces". In: *coordinates* 1 (2001), p. M1.

[123] João Pedro Malhado, Michael J Bearpark, and James T Hynes. "Non-adiabatic dynamics close to conical intersections and the surface hopping perspective". In: *Frontiers in chemistry* 2 (2014).

[124] Juha Tiihonen, Ilkka Kylänpää, and Tapio T Rantala. "Adiabatic and nonadiabatic static polarizabilities of H and H 2". In: *Physical Review A* 91.6 (2015), p. 062503.

[125] Christopher J Cramer. *Essentials of computational chemistry: theories and models*. John Wiley & Sons, 2013, pp. 355–358.

[126] Guiying Liang et al. "Accurate potential energy functions, non-adiabatic and spin–orbit couplings in the ZnH+ system". In: *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* 156 (2016), pp. 9–14.

[127] Thomas J. Penfold et al. "Spin-Vibronic Mechanism for Intersystem Crossing". In: *Chemical Reviews* 118.15 (2018). PMID: 29558159, pp. 6975–7025. DOI: 10.1021/acs.chemrev.7b00617. URL: https://doi.org/10.1021/acs.chemrev.7b00617.

[128] Felix Plasser et al. “Strong Influence of Decoherence Corrections and Momentum Rescaling in Surface Hopping Dynamics of Transition Metal Complexes”. In: *Journal of Chemical Theory and Computation* 15.9 (2019). PMID: 31339716, pp. 5031–5045. DOI: 10.1021/acs.jctc.9b00525. URL: https://doi.org/10.1021/acs.jctc.9b00525.

[129] Sebastian Mai and Leticia González. “Identification of important normal modes in nonadiabatic dynamics simulations by coherence, correlation, and frequency analyses”. In: *The Journal of Chemical Physics* 151.24 (2019), p. 244115. DOI: 10.1063/1.5129335. URL: https://doi.org/10.1063/1.5129335.

# Appendices

# Appendix A

# Statistical distributions

As distributions are an important component of the PIMC approach in Chapter 4, a small primer is presented in Appendix A.1. Analytical expressions for the GMD are also derived in Appendix A.1. A reminder on how to analytically calculate a multivariate Gaussian is shown in **??**. The simplification of Hyperbolic terms is shown in **??**.

## A.1 Notation and form of specific Distributions

Distributions are very important as the performance of the PI, MC, and MCMH methods are determined by the statistics. Evaluating the integral by MC with a sampling distribution $d^*$ which is infinitesimally close to $d$ will have rejection-less sampling. The choice of the sampling distribution is a major contribution to the computational efficiency of MC methods.

I have grouped together various definitions and derivations related to the distributions used in this chapter. I mainly make use of **uniform**, **normal**, and **mixture** distributions.

### Uniform distribution

The discrete uniform distribution $\mathcal{U}\{a, b\}$ or unif$\{a, b\}$ is a very simple way to represent the discrete electronic states. The most basic way of sampling surfaces is to draw them

$$a \in \mathcal{U}\{0, A-1\} \qquad \text{or} \qquad a \in \mathcal{U}\{1, A\}.$$

It can also be said that I draw samples with weights $w^a$.

### Normal/Gaussian distribution

The normal distribution $\mathcal{N}\{\mu, \sigma^2\}$ is used extensively in these notes, very often to describe the continuous normal mode co-ordinates. This distribution is defined by the mean $\mu$ and the variance $\sigma^2$.

Its probability density function (PDF) is:

$$\left(\sigma\sqrt{2\pi}\right)^{-1}\exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^{2}\right\}. \tag{A.1}$$

For my specific case, I choose Gaussians of the form

$$f^{a}(\underline{\boldsymbol{x}}) = \frac{1}{Z_{f^{a}}}\prod_{j=1}^{N}\left(\mathcal{F}_{j}\right)^{P}\pi_{j}(\boldsymbol{x}_{j}^{a}), \tag{A.2}$$

where

$$Z_{f^{a}} = \prod_{j=1}^{N}\frac{1}{2}\operatorname{csch}\left(\frac{\beta\omega_{j}}{2}\right), \tag{A.3}$$

$$\mathcal{F}_{j} = \left(\sqrt{\sinh(\tau\omega_{j})}\sqrt{2\pi}\right)^{-1}, \tag{A.4}$$

and

$$\pi_{j}(\boldsymbol{x}_{j}^{a}) = \exp\left[\operatorname{csch}(\tau\omega_{j})\sum_{i=1}^{P}x_{j,i}^{a}x_{j,i+1}^{a} - \coth(\tau\omega_{j})\sum_{i=1}^{P}\left(x_{j,i}^{a}\right)^{2}\right]. \tag{A.5}$$

These Gaussians are fully defined by the parameters: $\tau = \frac{beta}{P}$, continuous co-ordinates $x_{j,i}^{a}$, and $N$ frequencies $\omega_{j}$

I know the closed form expressions for all my normal distributions, and can sample from them without issue[1].

## Gaussian Mixture distribution (GMD)

The GMD[2] from my prior work [44]

$$\pi_{j}(\boldsymbol{x}_{j}^{a}) = \exp\left[-\frac{1}{2}(\boldsymbol{x}_{j}^{a})^{\mathcal{T}}\left(2\mathcal{C}_{j}\mathbb{1} - \mathcal{S}_{j}\underline{\boldsymbol{B}}\right)\boldsymbol{x}_{j}^{a}\right], \tag{A.6}$$

with $\underline{\boldsymbol{B}}$ a circulant matrix of dimension $P \times P$ defined by the row vector

$$[0100\cdots001]\,. \tag{A.7}$$

[1]For those concerned about the exact details see Numpy documentation where they describe the use of "256-step Ziggurat methods" to sample the normal distribution

[2]A GMD is another label for a mixture distribution whose mixture components are normal/gaussian distributions; it is also important to highlight that this is a special case where I can use either the GMD or a multivariate normal/gaussian distribution

Samples drawn from this distribution are termed *collective bead co-ordinates* as they are independent of the number of beads in a PI

$$y_{j\lambda}^{\tilde{a}} = \sum_{i=1}^{P} x_{ji}^{a} \mathcal{V}_{i\lambda}, \tag{A.8}$$

where

$$B_{ii'} = \sum_{\lambda=1}^{P} \mathcal{V}_{i\lambda} b_{\lambda\lambda} \mathcal{V}_{i'\lambda}^{*}. \tag{A.9}$$

This means that samples are drawn from the distribution $\pi_j$ We can of course extend this to include multiple surfaces by redefining

$$\pi_j(\boldsymbol{x}_j^a) = \exp\left[-\frac{1}{2}(\boldsymbol{x}_j^a)^{\mathcal{T}} \left(2\mathcal{C}_j \mathbb{1} - \mathcal{S}_j \underline{\boldsymbol{B}}\right) \boldsymbol{x}_j^a\right]. \tag{A.10}$$

## Analytical equations for GMDs

Here we work out the analytical expressions for our sampling distribution. Our goal is to have an exact expression for the histogram defined by points $x_j^\alpha$ on a grid $\alpha \in [0, 1, \cdots, N_{\text{grid}} - 1]$ where $x_j^\alpha = x_{\min} + \alpha \Delta x$ and $\Delta x = \frac{x_{\max} - x_{\min}}{N_{\text{grid}}}$ for any mode $j$. Beginning with $j = 1$: The continuous marginal distribution is given by Eq. (A.11), whereas Eq. (A.12) gives the discrete distribution

$$h(x_{1P}) = \int \prod_{i=1}^{P-1} \mathrm{d}x_{1i} \int \mathrm{d}\boldsymbol{x}_2 \cdots \int \mathrm{d}\boldsymbol{x}_N \, \varrho(\underline{\boldsymbol{q}}) \tag{A.11}$$

$$Hist(x_1^\alpha) = \int \mathrm{d}\boldsymbol{x}_1 \cdots \int \mathrm{d}\boldsymbol{x}_N \, \varrho(\underline{\boldsymbol{q}}) \delta(x_1^\alpha - x_{1P}). \tag{A.12}$$

which we can separate by electronic surface, $A$, using Eq. (46) from (prior paper)

$$h(x_{1P}) = \sum_{a=1}^{A} w^a \int \prod_{i=1}^{P-1} \mathrm{d}x_{1i} \int \mathrm{d}\boldsymbol{x}_2 \cdots \int \mathrm{d}\boldsymbol{x}_N \, \varrho_a(\underline{\boldsymbol{q}}). \tag{A.13}$$

We will explore by first focusing on $\varrho_a(\underline{\boldsymbol{q}})$ Eq. (A.2)

$$h(x_{1P}) = \sum_{a=1}^{A} w^a \int \prod_{i=1}^{P-1} \mathrm{d}x_{1i} \int \mathrm{d}\boldsymbol{x}_2 \cdots \int \mathrm{d}\boldsymbol{x}_N \, \varrho_a(\underline{\boldsymbol{q}}) \tag{A.14}$$

$$= \sum_{a=1}^{A} w^a \frac{\left(\mathcal{F}_1 \mathcal{F}_2 \cdots \mathcal{F}_N\right)^P}{Z_{\varrho_a}} \int \prod_{i=1}^{P-1} \mathrm{d}x_{1i} \int \mathrm{d}\boldsymbol{x}_2 \cdots \int \mathrm{d}\boldsymbol{x}_N \, \pi_1(\boldsymbol{x}_1^a) \pi_2(\boldsymbol{x}_2^a) \cdots \pi_N(\boldsymbol{x}_N^a). \tag{A.15}$$

By definition

$$\int \mathrm{d}\boldsymbol{x}_j \, \pi_j(\boldsymbol{x}_j^a) = \frac{\mathcal{C}_j}{2(\mathcal{F}_j)^P}, \tag{A.16}$$

and[3]

$$Z_{\varrho_a} = \prod_{i=1}^{N} \frac{1}{2} \operatorname{csch}\left(\tfrac{\beta w_j}{2}\right) = \prod_{i=1}^{N} \frac{2^{-1/2}\sqrt{(\cosh(\beta w_j)+1)}}{\sinh(\beta w_j)}, \tag{A.17}$$

therefore Equation (A.15) becomes

$$= \frac{(\mathcal{F}_1)^P}{\mathcal{C}_1} \int \prod_{i=1}^{P-1} \mathrm{d}x_{1i} \, \pi_1(\boldsymbol{x}_1^a) \tag{A.18}$$

$$= \frac{(\mathcal{F}_1)^P}{\mathcal{C}_1} \int \mathrm{d}x_{11}\, , \mathrm{d}x_{12}\, , \cdots \mathrm{d}x_{1P-1} \, \pi_1(\boldsymbol{x}_1^a). \tag{A.19}$$

After integrating out $N-1$ modes we are left with mode $j = 1$

$$h(x_{1P}) = \left(\frac{(\mathcal{F}_1)^P}{\mathcal{C}_1}\right) \sum_{a=1}^{A} w^a \int \mathrm{d}x_{11}\, , \mathrm{d}x_{12}\, , \cdots \mathrm{d}x_{1P-1} \, \pi_1(\boldsymbol{x}_1^a). \tag{A.20}$$

Here we can use a simple trick where we pretend that there is only one bead[4].

If $P = 1$ then it should be simple

$$h(x_{1P}) = \left(\frac{\mathcal{F}_1}{\mathcal{C}_1}\right) \sum_{a=1}^{A} w^a \pi_1(\boldsymbol{x}_1^a) \tag{A.21}$$

$$= \frac{\tanh(\beta w_j)}{\sqrt{2\pi \sinh(\beta w_j)}} \sum_{a=1}^{A} w^a \exp\left[\mathcal{S}_j (x_{11}^a)^2 - \mathcal{C}_j (x_{11}^a)^2\right] \tag{A.22}$$

$$= \frac{1}{\sqrt{2\pi}} \frac{\sqrt{\sinh(\beta w_j)}}{\cosh(\beta w_j)} \sum_{a=1}^{A} w^a \exp\left[-(\mathcal{C}_j - \mathcal{S}_j)(x_{11}^a)^2\right]. \tag{A.23}$$

Comparing to the generic form Eq. (A.1) we can see that

$$\sigma^2 = \frac{1}{2(C_j - S_j)} = \frac{1}{2}\left(\frac{\cosh(x)}{\sinh(x)} - \frac{1}{\sinh(x)}\right)^{-1} = \frac{1}{2}\frac{\sinh(x)}{\cosh(x) - 1}. \tag{A.24}$$

$$\coth(\beta w_j) - \operatorname{csch}(\beta w_j) = \frac{-1}{\sinh(\beta w_j)} + \frac{\cosh(\beta w_j)}{\sinh(\beta w_j)}. \tag{A.25}$$

[3]See **??**.

[4]For a more explicit derivation with $P$ beads see Appendix **??**.

# Appendix B

# Deriving energy estimator for PIMC method

This appendix details the derivation of an energy estimator for the PIMC method. A very concise derivation is shown in Appendix B.1. In Appendix B.2, it is shown that $\frac{\mathrm{d}}{\mathrm{d}\tau}W(\boldsymbol{Q},\boldsymbol{a}) = W(\boldsymbol{Q},\boldsymbol{a})\mathcal{F}(\boldsymbol{Q},\boldsymbol{a})$, using the explicit forms for $\frac{\mathrm{d}\mathbb{O}}{\mathrm{d}\tau}$, derived in Appendix B.3, and $\frac{\mathrm{d}\mathbb{M}}{\mathrm{d}\tau}$, derived in Appendix B.4.

## B.1 Deriving the Energy estimator

We calculate $\langle E\rangle$ as[1]

$$\langle E\rangle = -\frac{\mathrm{d}}{\mathrm{d}\tau}\ln(Z) \tag{B.1}$$

where Z can be expressed in terms of $g(\boldsymbol{Q})$ from Eq. (4.3), and $W(\boldsymbol{Q},\boldsymbol{a})$ from in Eq. (4.11):

$$Z = \int \mathrm{d}\boldsymbol{Q}\, g(\boldsymbol{Q}) = \int \mathrm{d}\boldsymbol{Q} \sum_{\boldsymbol{a}} W(\boldsymbol{Q},\boldsymbol{a}). \tag{B.2}$$

Using logarithmic rules $\langle E\rangle$ becomes[2]

$$\langle E\rangle = \frac{\int \mathrm{d}\boldsymbol{Q} \sum_{\boldsymbol{a}} \frac{\mathrm{d}}{\mathrm{d}\tau} W(\boldsymbol{Q},\boldsymbol{a})}{\int \mathrm{d}\boldsymbol{Q}\, g(\boldsymbol{Q})} = \frac{\int \mathrm{d}\boldsymbol{Q} \sum_{\boldsymbol{a}} W(\boldsymbol{Q},\boldsymbol{a})\mathcal{F}(\boldsymbol{Q},\boldsymbol{a})}{\int \mathrm{d}\boldsymbol{Q}\, g(\boldsymbol{Q})} = \frac{\int \mathrm{d}\boldsymbol{Q}\, g(\boldsymbol{Q})\mathcal{K}(\boldsymbol{Q})}{\int \mathrm{d}\boldsymbol{Q}\, g(\boldsymbol{Q})} = \Big\langle \mathcal{K}(\boldsymbol{Q}) \Big\rangle_{g(\boldsymbol{Q})} \tag{B.3}$$

where

$$\mathcal{K}(\boldsymbol{Q}) = \sum_{\boldsymbol{a}} \mathcal{F}(\boldsymbol{Q},\boldsymbol{a}) = \sum_{\boldsymbol{a}} \sum_{i=1}^{P} V(\boldsymbol{q}_i)_{a_i a_{i+1}} \left( \tilde{E}^{a_i} + \sum_{j=1}^{N} f(x_{j,i}^{a_i},\, x_{j,i+1}^{a_i}, \omega_j) \right) \tag{B.4}$$

[1]The negative sign from $\langle E\rangle$ is cancelled by $\tau$ derivative of $e^{-\tau\hat{H}}$

[2]For a more detailed derivation see appendix 1

So we have two estimators:

1. $\mathcal{K}(\boldsymbol{Q}) = \sum_{\boldsymbol{a}} \mathcal{F}(\boldsymbol{Q}, \boldsymbol{a})$
2. $\mathcal{F}(\boldsymbol{Q}, a)$,

which are evaluated over points $\boldsymbol{Q}$ and $\boldsymbol{a}$ obtained using Metropolis sampling (min(1, (g/r)*(r/g))

## B.2 Derivation of explicit form of F(Q, a)

For compactness we re-define $W(\boldsymbol{Q}, \boldsymbol{a})$ like so

$$W(\boldsymbol{Q}, \boldsymbol{a}) = \prod_{i=1}^{P} \langle \boldsymbol{q}_i, a_i | e^{-\tau \hat{h}} | \boldsymbol{q}_{i+1}, a_i \rangle \langle \boldsymbol{q}_{i+1}, a_i | e^{-\tau \hat{V}} | \boldsymbol{q}_{i+1}, a_{i+1} \rangle = \prod_{i=1}^{P} A_i B_i. \tag{B.5}$$

A more detailed explanation of eq. (B.3) will follow:

$$\langle E \rangle = \frac{\int \mathrm{d}\boldsymbol{Q} \sum_{\boldsymbol{a}} \frac{\mathrm{d}}{\mathrm{d}\tau} W(\boldsymbol{Q}, \boldsymbol{a})}{\int \mathrm{d}\boldsymbol{Q} \sum_{\boldsymbol{a}} W(\boldsymbol{Q}, \boldsymbol{a})}, \tag{B.6}$$

where

$$\frac{\mathrm{d}}{\mathrm{d}\tau} W(\boldsymbol{Q}, \boldsymbol{a}) = \frac{\mathrm{d}}{\mathrm{d}\tau} \prod_{i=1}^{P} A_i B_i = \sum_{i=1}^{P} \left[ \left( \frac{\mathrm{d}A_i}{\mathrm{d}\tau} B_i + A_i \frac{\mathrm{d}B_i}{\mathrm{d}\tau} \right) \times \prod_{j \neq i}^{P} A_j B_j \right]. \tag{B.7}$$

First we consider $\frac{\mathrm{d}A_j}{\mathrm{d}\tau}$

$$\frac{\mathrm{d}A_j}{\mathrm{d}\tau} = \frac{\mathrm{d}}{\mathrm{d}\tau} \langle \boldsymbol{q}_j, a_j | e^{-\tau \hat{h}} | \boldsymbol{q}_{j+1}, a_j \rangle = \frac{\mathrm{d}}{\mathrm{d}\tau} \mathbb{O} \left( \boldsymbol{q}_j, \boldsymbol{q}_{j+1} \right)_{a_j a_j}, \tag{B.8}$$

which we know from eq. (B.35) is equal to a prefactor $p(A_j)$ times $\mathbb{O} \left( \boldsymbol{q}_i, \boldsymbol{q}_{i+1} \right)_{a_j a_j}$:

$$\frac{\mathrm{d}A_j}{\mathrm{d}\tau} = p(A_j) \times A_j. \tag{B.9}$$

Next we consider $\frac{\mathrm{d}B_j}{\mathrm{d}\tau}$

$$\frac{\mathrm{d}B_j}{\mathrm{d}\tau} = \frac{\mathrm{d}}{\mathrm{d}\tau} \langle \boldsymbol{q}_j, a_j | e^{-\tau \hat{V}} | \boldsymbol{q}_j, a_{j+1} \rangle = \frac{\mathrm{d}}{\mathrm{d}\tau} \mathbb{M} \left( \boldsymbol{q}_j \right)_{a_j a_{j+1}}, \tag{B.10}$$

which we know from eq. (B.45) is equal to a prefactor $p(B_j)$ times $\mathbb{M} \left( \boldsymbol{q}_j \right)_{a_j a_{j+1}}$:

$$\frac{\mathrm{d}B_j}{\mathrm{d}\tau} = p(B_j) \times B_j. \tag{B.11}$$

Proceeding from eq. (B.7)

$$\sum_{i=1}^{P}\left[\Big(\frac{\mathrm{d}A_i}{\mathrm{d}\tau}B_i + A_i\frac{\mathrm{d}B_i}{\mathrm{d}\tau}\Big)\times\prod_{j\neq i}^{P}A_jB_j\right] \tag{B.12}$$

$$= \sum_{i=1}^{P}\left[\Big(p(A_i)A_iB_i + p(B_i)A_iB_i\Big)\times\prod_{j\neq i}^{P}A_jB_j\right] \tag{B.13}$$

$$= \sum_{i=1}^{P}\left[\Big(p(A_i) + p(B_i)\Big)\Big(A_iB_i\Big)\times\prod_{j\neq i}^{P}A_jB_j\right] \tag{B.14}$$

$$= \sum_{i=1}^{P}\left[\Big(p(A_i) + p(B_i)\Big)\times\prod_{i=1}^{P}A_iB_i\right], \tag{B.15}$$

$$= \mathcal{F}(\boldsymbol{Q},\boldsymbol{a})\times\prod_{i=1}^{P}A_iB_i, \tag{B.16}$$

$$= \mathcal{F}(\boldsymbol{Q},\boldsymbol{a})\times W(\boldsymbol{Q},\boldsymbol{a}), \tag{B.17}$$

## Result

So in summary we have

$$\frac{\mathrm{d}}{\mathrm{d}\tau}W(\boldsymbol{Q},\boldsymbol{a}) = W(\boldsymbol{Q},\boldsymbol{a})\mathcal{F}(\boldsymbol{Q},\boldsymbol{a}), \tag{B.18}$$

where

$$\mathcal{F}(\boldsymbol{Q},\boldsymbol{a}) = \sum_{i=1}^{P}\left[V(\boldsymbol{q}_i)_{a_ia_{i+1}} + \left(\tilde{E}^{a_i} + \sum_{j=1}^{N}f(x_{j,i}^{a_i},\, x_{j,i+1}^{a_i},\omega_j)\right)\right], \tag{B.19}$$

for an energy estimator

$$\langle E\rangle = \frac{\int \mathrm{d}\boldsymbol{Q}\sum_{\boldsymbol{a}}\frac{\mathrm{d}}{\mathrm{d}\tau}W(\boldsymbol{Q},\boldsymbol{a})}{\int \mathrm{d}\boldsymbol{Q}\sum_{\boldsymbol{a}}W(\boldsymbol{Q},\boldsymbol{a})} = \frac{\int \mathrm{d}\boldsymbol{Q}\sum_{\boldsymbol{a}}W(\boldsymbol{Q},\boldsymbol{a})\frac{\mathrm{d}}{\mathrm{d}\tau}\tilde{W}(\boldsymbol{Q},\boldsymbol{a})}{\int \mathrm{d}\boldsymbol{Q}\sum_{\boldsymbol{a}}W(\boldsymbol{Q},\boldsymbol{a})} = \left\langle\frac{\mathrm{d}}{\mathrm{d}\tau}\tilde{W}(\boldsymbol{Q},\boldsymbol{a})\right\rangle_{W(\boldsymbol{Q},\boldsymbol{a})}. \tag{B.20}$$

## B.3 $\tau$ derivative of individual O matrix element

An individual matrix element

$$\mathbb{O}\left(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}\right)_{aa} = \langle \boldsymbol{q}_i | \, e^{-\tau \hat{h}^a} \, | \boldsymbol{q}_{i+1} \rangle \tag{B.21a}$$

$$= \left\langle q_{1,i}\, q_{2,i} \, \cdots \, q_{N,i} \right| e^{-\tau \hat{h}^a} \left| q_{1,i+1}\, q_{2,i+1} \, \cdots \, q_{N,i+1} \right\rangle \tag{B.21b}$$

$$= \left(e^{-\tau \tilde{E}^a}\right) \left\langle x^a_{1,i}\, x^a_{2,i} \, \cdots \, x^a_{N,i} \right| e^{-\tau \hat{h}^a_{\mathrm{o}}} \left| x^a_{1,i+1}\, x^a_{2,i+1} \, \cdots \, x^a_{N,i+1} \right\rangle \tag{B.21c}$$

$$= \left(e^{-\tau \tilde{E}^a}\right) \prod_{j=1}^{N} K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right), \tag{B.21d}$$

where the kernel $K$ is:

$$K(x, x'; \tau\omega_j) = \sqrt{\frac{\operatorname{csch}(\tau\omega_j)}{2\pi}} \exp\left( \operatorname{csch}(\tau\omega_j) x x' - \coth(\tau\omega_j) \frac{1}{2}\Big(x^2 + (x')^2\Big) \right), \tag{B.22}$$

and $\tilde{E}$ comes from

$$\hat{h}^a = \Big[E^{aa} + \Delta^a\Big] + \left[\frac{1}{2}\sum_{j=1}^{N} \omega_j \Big(\hat{p}_j^2 + (\hat{x}^a_j)^2\Big)\right] \tag{B.23a}$$

$$= \tilde{E}^a + \hat{h}^a_{\mathrm{o}}, \tag{B.23b}$$

remembering that

$$x^a_j = q_j - d^a_j, \tag{B.24}$$

$$d^a_j = \frac{-g^{aa}_j}{\omega_j}, \tag{B.25}$$

$$\Delta^a = -\frac{1}{2}\sum_{j=1}^{N} \frac{(g^{aa}_j)^2}{\omega_j}. \tag{B.26}$$

To find $\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbb{O}\left(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}\right)_{aa}$

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbb{O}\left(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}\right)_{aa} = \frac{\mathrm{d}}{\mathrm{d}\tau}\left(e^{-\tau \tilde{E}^a}\right) \prod_{j=1}^{N} K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right) \tag{B.27a}$$

$$= \left(e^{-\tau \tilde{E}^a}\right) \left[ \tilde{E}^a \prod_{j=1}^{N} K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right) + \frac{\mathrm{d}}{\mathrm{d}\tau} \prod_{j=1}^{N} K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right) \right]. \tag{B.27b}$$

For each mode $j$ we need to calculate

$$\frac{\mathrm{d}}{\mathrm{d}\tau}K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right) = \frac{1}{\sqrt{2\pi}}\frac{\mathrm{d}}{\mathrm{d}\tau}\big(Ae^B\big) = \frac{1}{\sqrt{2\pi}}\left[A\frac{\mathrm{d}}{\mathrm{d}\tau}e^B + e^B\frac{\mathrm{d}}{\mathrm{d}\tau}A\right], \tag{B.28a}$$

where

$$e^B\frac{\mathrm{d}}{\mathrm{d}\tau}A = e^B\frac{\mathrm{d}}{\mathrm{d}\tau}\sqrt{\operatorname{csch}(\tau\omega_j)} = e^B\frac{-1}{2}\omega_j\cosh(\tau\omega_j)\operatorname{csch}^{3/2}(\tau\omega_j), \tag{B.29}$$

and

$$A\frac{\mathrm{d}}{\mathrm{d}\tau}e^B = Ae^B\frac{\mathrm{d}}{\mathrm{d}\tau}B = Ae^B\frac{\mathrm{d}}{\mathrm{d}\tau}\Big[\operatorname{csch}(\tau\omega_j)xx' - \coth(\tau\omega_j)\frac{1}{2}\Big(x^2+(x')^2\Big)\Big] \tag{B.30a}$$

$$= Ae^B\frac{1}{2}\omega_j\operatorname{csch}(\tau\omega_j)\left[\frac{x^2+(x')^2}{\sinh(\tau\omega_j)} - 2xx'\coth(\tau\omega_j)\right]. \tag{B.30b}$$

Then

$$\frac{1}{\sqrt{2\pi}}\left[A\frac{\mathrm{d}}{\mathrm{d}\tau}e^B + e^B\frac{\mathrm{d}}{\mathrm{d}\tau}A\right] \tag{B.31a}$$

$$= \frac{1}{\sqrt{2\pi}}Ae^B\frac{1}{2}\omega_j\operatorname{csch}(\tau\omega_j)\left[\frac{x^2+(x')^2}{\sinh(\tau\omega_j)} - 2xx'\coth(\tau\omega_j)\right] - \frac{1}{\sqrt{2\pi}}e^B\frac{1}{2}\omega_j\cosh(\tau\omega_j)\operatorname{csch}^{3/2}(\tau\omega_j) \tag{B.31b}$$

$$= \frac{\omega_j\operatorname{csch}(\tau\omega_j)}{2}\left(\left[\frac{x^2+(x')^2}{\sinh(\tau\omega_j)} - 2xx'\coth(\tau\omega_j)\right] - \cosh(\tau\omega_j)\right)\frac{Ae^B}{\sqrt{2\pi}} \tag{B.31c}$$

$$= f(x,x',\omega_j)\frac{Ae^B}{\sqrt{2\pi}} \tag{B.31d}$$

To find $\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbb{O}\left(\boldsymbol{q}_i,\boldsymbol{q}_{i+1}\right)_{aa}$

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbb{O}\left(\boldsymbol{q}_i,\boldsymbol{q}_{i+1}\right)_{aa} = \frac{\mathrm{d}}{\mathrm{d}\tau}\left(e^{-\tau\tilde{E}^a}\right)\prod_{j=1}^{N}K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right) \tag{B.32a}$$

$$= \left(e^{-\tau\tilde{E}^a}\right)\left[\tilde{E}^a\prod_{j=1}^{N}K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right) + \sum_{j=1}^{N}f(x,x',\omega_j)\prod_{j=1}^{N}K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right)\right] \tag{B.32b}$$

$$= \left(e^{-\tau\tilde{E}^a}\right)\left(\tilde{E}^a + \sum_{j=1}^{N}f(x,x',\omega_j)\right)\left[\prod_{j=1}^{N}K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right)\right]. \tag{B.32c}$$

Where

$$f(x,x',\omega_j) = \frac{\omega_j}{2\sinh(\tau\omega_j)}\left(\left[\frac{x^2+(x')^2}{\sinh(\tau\omega_j)} - 2xx'\coth(\tau\omega_j)\right] - \cosh(\tau\omega_j)\right) \tag{B.33}$$

### Result

So in summary we have

$$\mathbb{O}\left(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}\right)_{aa} = \left(e^{-\tau \tilde{E}^a}\right) \prod_{j=1}^{N} K\left(x^a_{j,i},\, x^a_{j,i+1};\, \tau\omega_j\right), \tag{B.34}$$

and (where $f$ is defined in Eq. (B.33))

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbb{O}\left(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}\right)_{aa} = \left(\tilde{E}^a + \sum_{j=1}^{N} f(x, x', \omega_j)\right) \times \mathbb{O}\left(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}\right)_{aa}. \tag{B.35}$$

## B.4 $\tau$ derivative of individual M matrix element

An individual matrix element

$$\begin{aligned} \mathbb{M}\left(\boldsymbol{q}_i\right)_{aa'} &= \langle a | e^{-\tau \hat{V}(\boldsymbol{q}_i)} | a' \rangle & \text{(B.36a)}\\ &= \exp[-\tau V(\boldsymbol{q}_i)_{aa'}] & \text{(B.36b)}\\ &= \boldsymbol{L}_a \exp[-\tau \boldsymbol{\varepsilon}] \boldsymbol{L}^\dagger_{a'}, & \text{(B.36c)} \end{aligned}$$

for some $\underline{\boldsymbol{L}}, \boldsymbol{\varepsilon}$ obtained by diagonalizing the matrix $\boldsymbol{V}(\boldsymbol{q}_i)$

$$\begin{aligned} V(\boldsymbol{q}_i)_{aa'} &= \boldsymbol{g}^{aa'}\boldsymbol{q}_i(1-\delta_{aa'}) + \frac{1}{2}\boldsymbol{q}_i \underline{\boldsymbol{G}}^{aa'} \boldsymbol{q}_i^\dagger & \text{(B.37a)}\\ &= \sum_{\lambda=1}^{A} L_{a\lambda}\varepsilon_\lambda L_{a'\lambda}, & \text{(B.37b)} \end{aligned}$$

$$\hat{V}^{aa'} = \sum_{j}^{N} g_j^{aa'} \hat{q}_j (1-\delta_{aa'}) + \frac{1}{2}\sum_{jj'}^{N} g_{jj'}^{aa'} \hat{q}_j \hat{q}_{j'}. \tag{B.38}$$

To find $\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbb{M}\left(\boldsymbol{q}_i\right)_{aa'}$

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau}\mathbb{M}\left(\boldsymbol{q}_i\right)_{aa'} &= \frac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{L}_a \exp[-\tau\boldsymbol{\varepsilon}]\boldsymbol{L}^\dagger_{a'} & \text{(B.39)}\\ &= \boldsymbol{L}_a \frac{\mathrm{d}}{\mathrm{d}\tau}\Big(\exp[-\tau\boldsymbol{\varepsilon}]\Big)\boldsymbol{L}^\dagger_{a'} & \text{(B.40)}\\ &= \boldsymbol{L}_a \Big(-\boldsymbol{\varepsilon} \times \exp[-\tau\boldsymbol{\varepsilon}]\Big)\boldsymbol{L}^\dagger_{a'} & \text{(B.41)}\\ &= V(\boldsymbol{q}_i)_{aa'} \times \exp[-\tau V(\boldsymbol{q}_i)_{aa'}] & \text{(B.42)}\\ &= V(\boldsymbol{q}_i)_{aa'} \times \mathbb{M}\left(\boldsymbol{q}_i\right)_{aa'}. & \text{(B.43)} \end{aligned}$$

## Result

So in summary we have

$$\mathbb{M}\left(\boldsymbol{q}_i\right)_{aa'} = \boldsymbol{L}_a \exp[-\tau\boldsymbol{\varepsilon}]\boldsymbol{L}_{a'}^{\dagger}, \tag{B.44}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbb{M}\left(\boldsymbol{q}_i\right)_{aa'} = V(\boldsymbol{q}_i)_{aa'} \times \mathbb{M}\left(\boldsymbol{q}_i\right)_{aa'}\,. \tag{B.45}$$

# Appendix C

# Assorted Notes

Appendix C.1 provides a more in-depth explanation of the Normal Ordering inequalities (Equations (5.25) and (5.26)) that were defined in Section 5.3.

In Section 5.5, the specific method that `termfactory` uses to solve the problem of determining pairings is discussed. A detailed summary of the general contraction scheme is shown in Appendix C.2.

I initially wanted to express the pairing of terms as a language, such as would be defined in Formal Language Theory, and so a short discussion of Formal Language Theory is provided in Appendix C.3. Additionally, a short discussion of the complexity of linear programming is provided in Appendix C.4.

## C.1 Normal Ordering inequality explanation

In Section 5.3 we define the Normal Ordering (NO) constraint as the requirement (for the simplified case where $\bar{f} = 0$) that each *valid* term $\hat{\mathbf{A}}_N \hat{\mathbf{B}}_N^M \hat{\mathbf{C}}_N^M \cdots \hat{\mathbf{Z}}^M$ has at least one pairing where no operators pair with themselves. This constraint can be expressed be applying two conditions to each operator in the term:

1. If either $M = 0$ or $N = 0$ an operator cannot pair with itself and therefore meets the constraint.

2. Otherwise, if both inequalities

$$M \leq N_{\text{tot}} - N, \tag{C.1}$$
$$N \leq M_{\text{tot}} - M, \tag{C.2}$$

   are satisfied, then the operator satisfies the NO constraint.

   If each operator in the term satisfies either 1 or 2 then the term is Normal Ordered and meets the constraint.

However, it may not be clear how the inequalities Equations (C.1) and (C.2) satisfy the NO constraint, or how they were obtained. I will show that these inequalities satisfy the NO constraint in the general case (where $\bar{f} \in \mathbb{R}$) and by extension also the simplified case (where $\bar{f} = 0$).

## Setup

First, we need to quantify what it means for an operator to pair with itself, and therefore break the Normal Ordering constraint. Consider the general term

$$\hat{\mathbf{A}}_{N_a}\hat{\mathbf{B}}^{M_b}_{N_b}\hat{\mathbf{C}}^{M_c}, \tag{C.3}$$

whose operators are balanced[1]:

$$M_{\text{tot}} = N_{\text{tot}},$$
$$M_b + M_c = N_a + N_b.$$

By definition $\hat{\mathbf{A}}$ and $\hat{\mathbf{C}}$ cannot pair with themselves and so we need only consider checking if the $\hat{\mathbf{B}}$ operator can pair with itself. Starting with the simplest case: $\hat{\mathbf{A}}_1\hat{\mathbf{B}}^1_1\hat{\mathbf{C}}^1$, where $M_{\text{tot}} = 2$ and $N_{\text{tot}} = 2$, two possible unique pairings exist:

$$\hat{\mathbf{A}}_i\hat{\mathbf{B}}^i_j\hat{\mathbf{C}}^j, \tag{C.4}$$
$$\hat{\mathbf{A}}_i\hat{\mathbf{B}}^j_j\hat{\mathbf{C}}^i. \tag{C.5}$$

The first pairing obeys the Normal Ordering constraint and the second pairing does not. We can distinguish between these two pairings by splitting each of the $M_b$ and $N_b$ labels into two parts: one which counts the number of pairings that obey the Normal Ordering constraint (i.e. with other operators $\hat{\mathbf{A}}$ or $\hat{\mathbf{C}}$) and one which counts the number of pairings $\hat{\mathbf{B}}$ has with itself .

$$M_b = M_b^{\text{NO}} + M_b^{\text{S}} \tag{C.6}$$
$$N_b = N_b^{\text{NO}} + N_b^{\text{S}} \tag{C.7}$$

The term which obeys the Normal Ordering constraint $\hat{\mathbf{A}}_i\hat{\mathbf{B}}^i_j\hat{\mathbf{C}}^j$, has values

$$M_b = M_b^{\text{NO}} + M_b^{\text{S}} = 1 + 0 = 1,$$
$$N_b = N_b^{\text{NO}} + N_b^{\text{S}} = 1 + 0 = 1.$$

[1]Any term whose operators are not balanced is *invalid* by definition, and therefore we can ignore such terms when checking if a term meets the Normal Ordering constraint.

The term which does not satisfy the Normal Ordering constraint $\hat{\mathbf{A}}_i \hat{\mathbf{B}}_j^j \hat{\mathbf{C}}^i$, has values

$$M_b = M_b^{\mathrm{NO}} + M_b^{\mathrm{S}} = 0 + 1 = 1,$$
$$N_b = N_b^{\mathrm{NO}} + N_b^{\mathrm{S}} = 0 + 1 = 1.$$

$M^{\mathrm{S}} = N^{\mathrm{S}}$ must always be true for any operator, by definition, because a pairing is between an upper $\hat{i}^\dagger$ operator and a lower $\hat{j}$ operator, a term cannot be *valid* and have unpaired operators. However, $M^{\mathrm{NO}}$ and $N^{\mathrm{NO}}$ are not required to be equal.

We can now define, for a generic operator $\hat{\mathbf{B}}_N^M$, the condition for satisfying the Normal Ordering constraint as

$$M^{\mathrm{S}} = N^{\mathrm{S}} = 0, \tag{C.8}$$

and consequently when

$$M^{\mathrm{S}} = N^{\mathrm{S}} > 0, \tag{C.9}$$

it fails to satisfy the constraint. We can now show that Equations (C.1) and (C.2) satisfy the NO constraint.

## Contradiction example

Since the NO constraint only requires:

> "*at least one* pairing exists, where no operators pair with themselves"

then we want to show that if both inequalities are satisfied, at least one such pairing must exist. Consider the generic term

$$\hat{\mathbf{A}}_{N_a}^{M_a} \hat{\mathbf{B}}_{N_b}^{M_b} \cdots \hat{\mathbf{G}}_{N_g}^{M_g} \cdots \hat{\mathbf{Z}}_{N_z}^{M_z} \tag{C.10}$$

where $\hat{\mathbf{G}}$ is some arbitrary operator in the term with both upper and lower labels. We know by definition:

$$N_{\mathrm{tot}} = N_a + \cdots + N_g + \cdots + N_z, \tag{C.11}$$
$$M_{\mathrm{tot}} = M_a + \cdots + M_g + \cdots + M_z, \tag{C.12}$$

where

$$N_g = N_g^{NO} + N_g^S, \tag{C.13}$$
$$M_g = M_g^{NO} + M_g^S, \tag{C.14}$$

and

$$M_g^S = N_g^S. \tag{C.15}$$

Assume that both inequalities are satisfied:

$$M_g \leq N_{\text{tot}} - N_g, \tag{C.16}$$
$$N_g \leq M_{\text{tot}} - M_g, \tag{C.17}$$

and that for all possible pairings $\hat{\mathbf{G}}$ <u>must</u> pair with itself, which can be represented by the following:

$$N_g^S = M_g^S > 0, \tag{C.18}$$
$$N_{\text{tot}} - N_g = M_g^{NO}, \tag{C.19}$$
$$M_{\text{tot}} - M_g = N_g^{NO}. \tag{C.20}$$

## Justifying eqs. (C.19) and (C.20)

Equations (C.19) and (C.20) follow from the requirement that $\hat{\mathbf{G}}$ <u>must</u> pair with itself. $M_g^{NO}$ counts the number of $\hat{\imath}^\dagger$ operators from $\hat{\mathbf{G}}$ that pair with other operators in Equation (C.10).

Conceptually it should be clear that $M_g^{NO} \leq N_{\text{tot}} - N_g$ is true, as there must be at least the same number of $\hat{\jmath}$ operators as $\hat{\imath}^\dagger$ operators for the pairings to exist. Since $N_{\text{tot}} - N_g$ counts all $\hat{\jmath}$ operators that do not belong to $\hat{\mathbf{G}}$, and $M_g^{NO}$ counts all $\hat{\imath}^\dagger$ operators from $\hat{\mathbf{G}}$ that follow the Normal Ordering constraint.

It can also be shown this is true by definition, starting with Equation (C.16):

$$M_g \leq N_{\text{tot}} - N_g \tag{C.21a}$$
$$M_g^{NO} \leq N_{\text{tot}} - N_g - M_g^S, \tag{C.21b}$$

but if $M_g^{NO} > N_{\text{tot}} - N_g$, then

$$M_g^{NO} > N_{\text{tot}} - N_g \tag{C.22a}$$
$$N_{\text{tot}} - N_g - M_g^S > N_{\text{tot}} - N_g \tag{C.22b}$$
$$-M_g^S > 0 \tag{C.22c}$$
$$M_g^S < 0 \tag{C.22d}$$

which contradicts Equation (C.18) thus $M_g^{NO} \leq N_{\text{tot}} - N_g$.

However, $M_g^{NO} < N_{\text{tot}} - N_g$ implies that there is a $\hat{\jmath}$ operator that is not pairing with a $\hat{\imath}^\dagger$ operator from $\hat{\mathbf{G}}$. Since $N_g^S = M_g^S > 0$ we are free to choose a different pairing of Equation (C.10), by pairing one of the $\hat{\imath}^\dagger$ operators that $N_g^S$ counts with the aforementioned $\hat{\jmath}$ operator. The same can be done for one of the $\hat{\jmath}$ operator's counted by $M_g^S$.

As a discrete example consider:

$$\hat{\mathbf{A}}_{ij}\hat{\mathbf{G}}^{im}_{lm}\hat{\mathbf{Z}}^{lj}, \tag{C.23}$$

where $M_g^{NO} = 1$ and $N_{\text{tot}} - N_g = 4 - 2 = 2$ thus $M_g^{NO} < N_{\text{tot}} - N_g$. We can "re-pair" the $j$

and $m$ labels:

$$\hat{\mathbf{A}}_{ij}\hat{\mathbf{G}}^{ij}_{lm}\hat{\mathbf{Z}}^{lm}, \tag{C.24}$$

where the operator $\hat{\mathbf{G}}$ no longer pairs with itself and the term clearly satisfies the NO constraint. After this "re-pairing" $M_g^{NO} = 2 = 4 - 2 = N_{\text{tot}} - N_g$. We can preform this process for terms of any size like so

$$\hat{\mathbf{A}}^{i}_{\cdots}\hat{\mathbf{B}}^{\cdots}_{\cdots}\cdots\hat{\mathbf{G}}^{\cdots j\cdots}_{\cdots j\cdots}\cdots\hat{\mathbf{Z}}^{\cdots}_{i} \quad \rightarrow \quad \hat{\mathbf{A}}^{i}_{\cdots}\hat{\mathbf{B}}^{\cdots}_{\cdots}\cdots\hat{\mathbf{G}}^{\cdots j\cdots}_{\cdots i\cdots}\cdots\hat{\mathbf{Z}}^{\cdots}_{j} \tag{C.25}$$

Thus, whenever $M_g^{NO} < N_{\text{tot}} - N_g$ we can simply move operators from $N_g^S$ to $M_g^{NO}$ until $M_g^{NO} = N_{\text{tot}} - N_g$. If there were not enough operators to move over (e.g. $M_g^{NO} + N_g^S < N_{\text{tot}} - N_g$) then after moving all the operators from $N_g^S$ to $M_g^{NO}$, the following would be true $M_g^S = N_g^S = 0$ which contradicts Equation (C.18)! We are left with two possibilities, either a pairing where $\hat{\mathbf{G}}$ does not pair with itself exists, or there is enough operators and so $M_g^{NO} = N_{\text{tot}} - N_g$. A similar argument can be employed for $M_{\text{tot}} - M_g = N_g^{NO}$.

Thus, we can say that if either Equation (C.19) or Equation (C.20) are not true, then Equation (C.18) is not true and therefore $\hat{\mathbf{G}}$ cannot pair with itself.

### Proof

By definition (eq. (C.20)):

$$M_{\text{tot}} - M_g = N_g^{NO} \tag{C.26}$$

and (eq. (C.17)):

$$N_g \leq M_{\text{tot}} - M_g \tag{C.27}$$

but that implies

$$\begin{aligned} N_g &\leq M_{\text{tot}} - M_g && \text{(C.28)}\\ N_g &\leq N_g^{NO} && \text{(C.29)}\\ N_g^{NO} + N_g^S &\leq N_g^{NO} && \text{(C.30)}\\ N_g^S &\leq 0 && \text{(C.31)} \end{aligned}$$

which contradicts Equation (C.18) ($N_g^S > 0$). This holds true for any of the operators in Equation (C.10). Therefore if Equations (C.16) and (C.17), (or equivalently Equations (C.1) and (C.2)) are satisfied then there must be <u>at least one</u> pairing where no operators pair with themselves.

## C.2 More general contraction scheme

*This entire section is taken verbatim (with permission) from M. Nooijen's notes. The language and notation is slightly different than those used in Chapter 5.*

This section uses a more general contraction scheme. It describes equations that we use currently to describe thermal properties. We can also use it for time-correlation functions. The aim would be to formulate a theory that has less (or no) issues with singularities of the Matrix **u**. This will be a first attempt at writing down the general theory. It may need some refinements and better explanations later on. I think it can be used as a blue print for a generalized equation and code generator. As before we will focus on the most complicated term $\left\langle \hat{P}\hat{\mathbf{H}}\hat{\mathbf{W}} \right\rangle$. All operators in the more general theory now carry both lower and upper labels. Let us write a generic (residual) term with explicit indices

$$R^{m_1 m_2 \cdots m_M}_{i_1 i_2 \cdots i_I} = \left\langle P^{m_1 m_2 \cdots m_M}_{i_1 i_2 \cdots i_I} \sum_{\text{all labels } k,l,m,n} \mathbf{H}^{l_1 l_2 \cdots l_L}_{k_1 k_2 \cdots k_K} \mathbf{W}^{j_1 j_2 \cdots j_J}_{n_1 n_2 \cdots n_N} \right\rangle \tag{C.32}$$

In a non-vanishing expression the lower labels all have to be paired with the upper labels. The pairings can only occur between different groups of labels. We can write

$$R^{m_1 m_2 \cdots m_M}_{i_1 i_2 \cdots i_I} = \sum_{\text{all labels } k,l,m,n} \mathbf{H}^{l_1 l_2 \cdots l_L}_{k_1 k_2 \cdots k_K} \mathbf{W}^{j_1 j_2 \cdots j_J}_{n_1 n_2 \cdots n_N} F^{m_1 \cdots m_M \,|\, l_1 \cdots l_L \,|\, j_1 \cdots j_J}_{i_1 \cdots i_I \,|\, k_1 \cdots k_K \,|\, n_1 \cdots n_N} \tag{C.33}$$

The vertical bars in the new "contraction tensor" **F** indicate the various groups of indices. As before, upper indices have to be paired with lower indices, but always pairing operators from different groups. Every pairing of two normal mode labels introduces a pair-like Kronecker delta $\delta^p_q$, and the labels can then simply be equated, while reducing the summation labels.

In the final result one will always keep the projection labels (on $\hat{P}$), which are not summed over, and it is most convenient to use as the other remaining labels, the labels on **H** that are contracted to **W**, and which will be summed over. The labels on **W** will all be replaced.

In this more general theory there is one additional rule. If external labels from $\hat{P}$ are contracted to **H**, there is an additional factor. If an upper label $m$ from $P$ is contracted to $k$ on **H**, corresponding to the contraction $\left\langle \hat{m}^\dagger \hat{k} \right\rangle$ we have the factor $f_m \delta_{mk}$. If a lower label $i$ on $P$ is contracted with $l$ on the hamiltonian we get a factor $\bar{f}_i \delta_{il}$. It follows that one can still identify paired indices (from Kronecker $\delta$), but there are additional factors $f_m, \bar{f}_i$ associated with *external labels* on **H**, that are not summed over.

It is part of the theory that $\bar{f}_i = 1 + f_i$. The factors $f_i$ can be arbitrary positive numbers, and final results are in principle independent of these numbers (i.e. for an exact version of the theory). The numbers can depend on the normal mode of interest, and this is indicated by the subscript. A simpler theory can arise if $f_i = f, \bar{f}_i = (1 + f)$. Let me note that the previous simpler theory follows from the more general theory by setting $f_m = 0$. We anticipate that results for the auto-correlation function will be (slightly) different if equations are truncated. The most important aspect is that the $\hat{W}$ and in particular **u** amplitudes will be different, and this we hope results in improved robustness during time-propagation. We should implement the most general theory first, i.e. using explicitly $f_m, \bar{f}_i$ factors that depend on the normal mode. As before, In the sequel we will be completely systematic about the naming convention.

## Derivation of generic prefactors in general term

Let us consider a generic term in the residual equation

$$R^{m_1\cdots m_M}_{i_1\cdots i_I} = \frac{1}{I!}\hat{S}(i_1\cdots i_I)\frac{1}{M!}\hat{S}(m_1\cdots m_M)\left\langle \hat{\Omega}^{m_1\cdots m_M}_{i_1\cdots i_I} \sum_{\text{all labels}} \mathbf{H}^{l_1\cdots l_L}_{k_1\cdots k_K}\mathbf{W}^{j_1\cdots j_J}_{n_1\cdots n_M}\right\rangle \tag{C.34}$$

The (normalized) symmetrization operators like $\frac{1}{I!}\hat{S}(i_1\cdots i_I)$ leave a symmetric expression unchanged. We can abbreviate such a term by listing only the number of operators of a given type

$$R^M_I = \frac{1}{I!}S(I)\frac{1}{M!}S(M)\frac{1}{L!K!J!N!}\left\langle \hat{\Omega}^M_I \mathbf{H}^L_K W^J_N\right\rangle \tag{C.35}$$

In the term in expectation value, we have to consider all possible allowed pairings (permutations), that all give rise to the same canonically ordered result. Let us assume that $L_I$ operators are paired with operators in $I$, while $L_N = L - L_I$ are paired with $N$. Likewise we can partition $K = K_M + K_J$. The canonical form of the equation including all factors takes the form

$$\begin{aligned} R^M_I = {} & \hat{S}(I)\hat{S}(M)\frac{1}{I!M!}\frac{1}{L!K!J!N!}f_{M_K}\bar{f}_{I_L}\mathbf{H}^{I_L,L_N}_{M_K,K_J}W^{I_J,K_J}_{M_N,L_N} \\ & \times\left(\binom{I}{I_L}I_L!\right)\left(\binom{N}{L_N}L_N!\right)\left(\binom{M}{M_K}M_K!\right)\left(\binom{J}{K_J}K_J!I_J!M_N!\right) \end{aligned} \tag{C.36}$$

The factorial expressions on the second line count how many different pairings can be made including all permutations. The expression can be simplified, using $I_J = I - I_L$ and $M_N = M - M_K$, to

$$R^M_I = \hat{S}(I)\hat{S}(M)\frac{f_{M_K}\bar{f}_{I_L}}{L!K!J!N!}\mathbf{H}^{I_L,L_N}_{M_K,K_J}W^{I_J,K_J}_{M_N,L_N}\left(\binom{N}{L_N}L_N!\right)\left(\binom{J}{K_J}K_J!\right) \tag{C.37}$$

$$= \hat{S}(I)\hat{S}(M)\frac{f_{M_K}\bar{f}_{I_L}}{L!K!(J-K_J)!(N-L_N!)}\mathbf{H}^{I_L,L_N}_{M_K,K_J}W^{I_J,K_J}_{M_N,L_N} \tag{C.38}$$

The above formula agrees with the previously derived general formula if we choose $M = N = 0$.

## C.3 Formal language theory

In our work building software to automatically generate expressions, we have had occasion to consider our series expansions from the perspective of formal language theory.

Formal language theory is a branch of computational science that explores the properties of languages. A language is simply a set of finite-length words over a finite alphabet of symbols, where a word is a specific string of symbols. For example

**Definition 1** *let $\Sigma$ be an alphabet containing the two symbols* 1 *and* 0

$$\Sigma = \{0, 1\}. \tag{C.39}$$

**Definition 2** *Let each word $w$ in our language $L$ be such that every $w \in L$ contains as exactly as many* 1*'s as it contains* 0*'s.*

**Definition 3** *Then $L$ is the infinite set* $\{\epsilon, 10, 01, 1100, 1010, 1001, 0011, 0101, 0110, 111000, \cdots\}$

A language has a finite alphabet, and each word is finite, but the language may be infinite, as in our example above.

Formal language theory arose from work in linguistics and computer science. Linguists were interested in describing derivation grammars that could be used to understand the meaning of human languages. Computer scientists were initially interested in describing switching circuits, and later in developing parsers for programming languages. As formal language theory matured, it became a more general discipline for describing the computational properties of strings and the automata that can produce or recognize them.

In addition to human languages, there are also programming languages. Each programming language can be understood as a language in formal language theory:

- The alphabet $\Sigma$ of a programming language is the set of letters, numbers, and special symbols available on a keyboard
- The words $w$ of a programming language are strings of the alphabet that satisfy the syntactic and semantic rules of the programming language. Such words are what we know as 'valid' or 'legal' programs

Thus a programming language is a formal language $L = \{w_i\}$ where each $w_i$ is a valid program.

How do we specify the rules of a programming language so that we can recognize a valid program? We do this by constructing a parser that the rules of the language. In the early days of computing, parsers were hand-crafted for each specific language, but this was a tedious and error-prone method.

One of the key results of formal language theory is the Chomsky hierarchy, Figure C.1 which states the four major types of languages (in bold) and the machines that are required to recognize them:

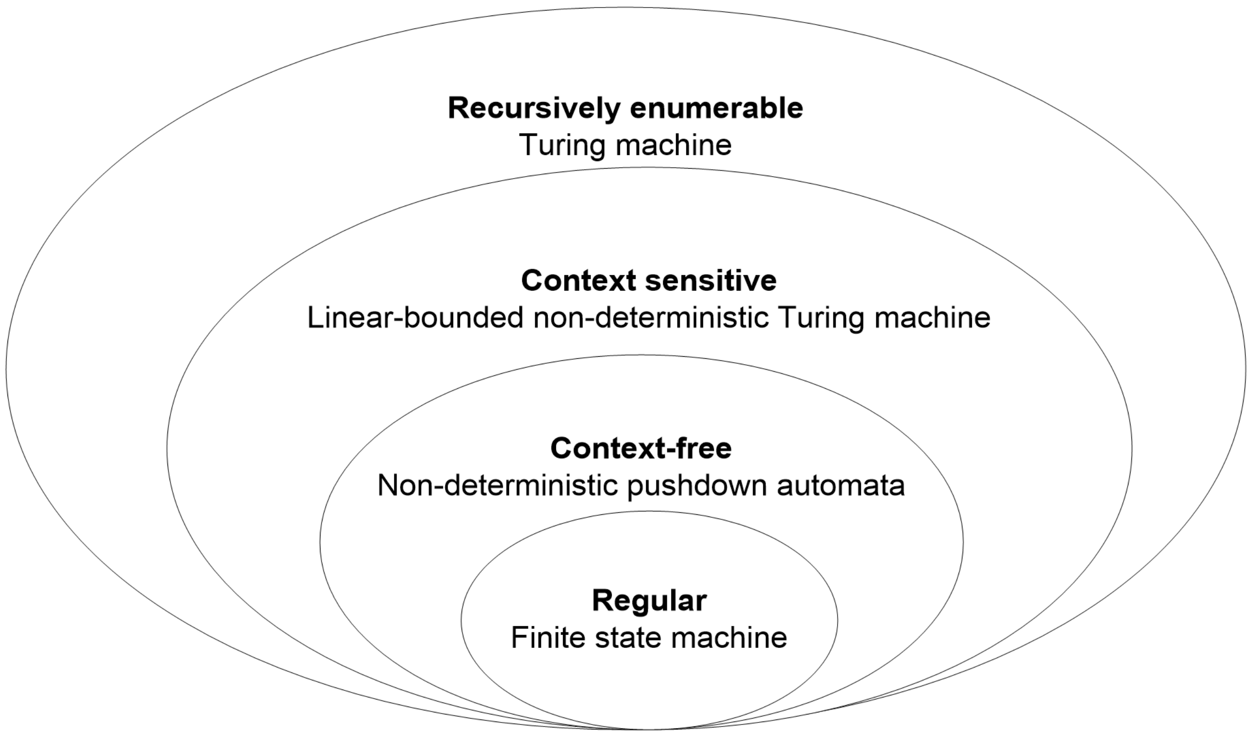
**Recursively enumerable**
Turing machine
**Context sensitive**
Linear-bounded non-deterministic Turing machine
**Context-free**
Non-deterministic pushdown automata
**Regular**
Finite state machine

Figure C.1: Chomsky hierarchy.

Figure C.1 shows set inclusion: regular languages are a subset of context-free languages; context free languages are a subset of context-sensitive languages, and all are a subset of recursively enumerable languages.

Since Turing machines are capable of doing any known computation, any language that is computable is at most recursively enumerable—this is an upper bound, but not a tight one. Often we want to find a more restrictive upper bound by showing that our language is contained in one of the subset languages. For example, experience with programming languages has taught us that it is best if we design programming languages so their parsers are no more complicated than context-free. This is partly because it reduces the complexity of the compiler, but also because humans need to read programs, and context-sensitive grammars can be very difficult for humans to parse [2] More information on formal language theory can be found in standard texts [79, 80].

Now let us turn to consider how language theory may apply to series expansions. First,

[2]Note that when we speak of the complexity of a language parser, we mean only the complexity of the machine that recognizes valid programs in the language.

let's consider a very simple example, the geometric series:

$$\sum_{n=0}^{\infty} x^n = 1 + x + x^2 + x^3 + \cdots \tag{C.40}$$

This expansion can be viewed as a language that consists of an infinite set of words, each word w of the form $x^n$. We can express this language with a regular expression of the following form:

$$1 + x(x)* \tag{C.41}$$

This regular expression is the union (+) of 1 and one or more $x$'s (the $*$ operator means zero or more occurrences of what is in the brackets.

Every regular language is recognized (or can be generated by) a finite state machine, and every finite state machine defines a regular language. For the regular expression Equation (C.41), a finite state machine representation would be this:

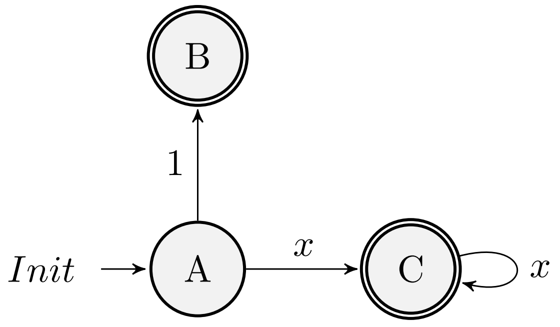

Figure C.2: FSM representation of Equation (C.41)

The finite state machine in Figure C.2 has three states A, B, and C. A is the initial state because it has a transition entering it from no other state. States B and C are both final states [3]: any string that ends in a final state is a word in the language.

We enter the machine from the left, on the transition state A. At this point we can recognize 1 and enter the final state B, which means 1 is a word in our language. From B we cannot go anywhere, so this terminates what can be recognized on this path. Alternatively, we can recognize $x$ and enter state C. Since state C is a final state, then we also recognize $x$ as a word in the language. But we can also continue with the transition labelled $x$ and remain in state C, and this can be repeated as many times as we like. Thus $xx$ (that is, $x^2$) and $xxx$ (that is, $x^3$) and so on will also be words in our language; in fact any $x^n$ is a word in our language.

The geometric series is simple enough that it can be done by hand, and we don't really need the mechanism of regular languages and finite state machines. But consider a highly

[3]A final state is traditionally represented by two concentric circles.

simplified form of one of our expansions:

$$H_b^a Z_b^a$$

This language will consist of words such as the following:

$$H_1^0 Z_0^1$$
$$H_2^0 Z_0^2$$
$$H_0^1 Z_1^0$$
$$H_5^1 Z_1^5$$

and so on.

We can ask: is the language $L = H_b^a Z_a^b$ a regular language? Intuitively, we guess that it is not, because it seems we must "remember" $A$ and $b$ from the $H$ term so that we can "repeat" them for the $Z$ term, and a finite state machine, having only states and transitions, has no apparent "memory" where we could store the values of $A$ and $b$.

We can show that $L = H_b^a Z_a^b$ is not regular by reducing it to a known non-regular language:

1. The language $A^n B^n$ where $n \in \mathbb{N}$ is known not to be regular
2. If we choose $a = b$ in our language, then we have $L = H_a^a Z_a^a$
3. To recognize this we must recognize $H^a Z^a$, but this is isomorphic to $A^n B^n$, which is non-regular
4. Hence $L$ is non-regular

It is worth noting here that L is non-regular where $A$ and $b$ can take any value in $N$. If we limit the choice of $A$ and $b$ to, say, the set $1, 2, 3, 4, 5$, then L will be regular, because it now becomes possible to define a (much larger) finite state machine that captures all the permutations of 1 through 5 in its states. We do not show such a construction here.

In the Chomsky hierarchy (Figure C.1), the next superset of languages above regular languages, are languages which do have a memory: these are the context-free grammars (CFGs). CFGs are recognized by a non-deterministic pushdown automaton (NDPDA). In essence, an NDPDA is a finite state machine that also has a "stack" on which it can "push" and "pop" symbols: the stack is a simple form of "memory". If we can find an NDPDA that recognizes or generates our language $L$, then we will have shown that $L$ is at most context-free (on the other hand, if we cannot find an NDPDA that recognizes our language $L$, then $L$ may be context-sensitive or even recursively enumerable).

To simplify such an NDPDA, let us use a simple encoding for the a and b in our language $L$. Instead of using base 10, we will use base 1: thus instead of $A$ being (for example) "5", it will be "11111"; similarly instead of $b$ being (for example) "7", it will be "0000000". By choosing this encoding, we reduce the complexity of our input alphabet. Also, we will create an NDPDA that generates our language, by writing output symbols instead of reading input symbols.

With these choices, one NDPDA that recognizes L is this:

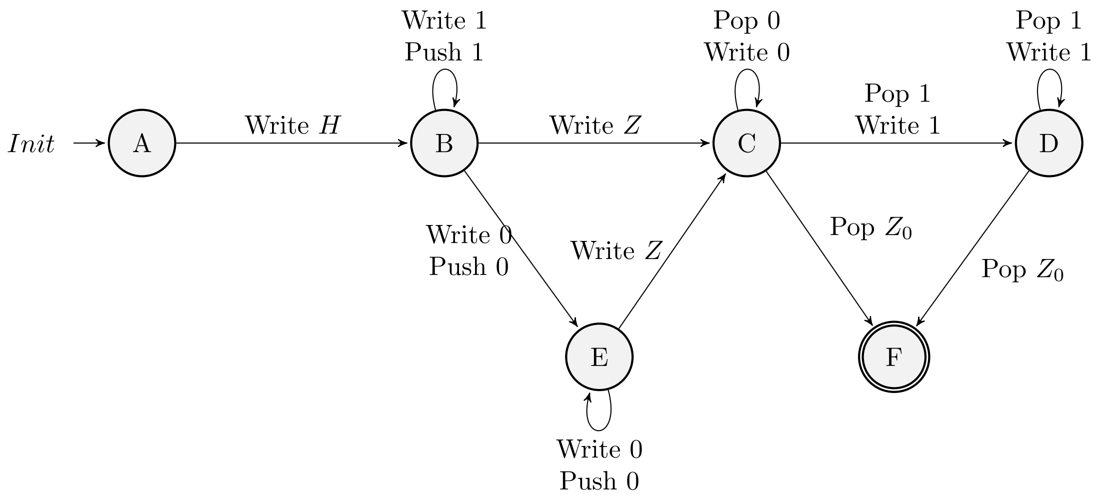

Figure C.3: NDPDA that recognizes $L$.

In this pushdown machine, we do not show the stack explicitly; instead, we have push and pop operators to push a value on the stack, or pop it off. The stack has a start symbol $Z_0$: by popping off this symbol and looking at it, we can tell if the stack is empty. Let us follow some paths through the NDPDA to see how it works:

1. If we go from A to B to C to F we will write $H$, then write $Z$, then pop the stack start symbol. This gives us the output $HZ$, which is the word in our language $L$ corresponding to $a = b = 0$

2. If we go from A to B writing $H$, then write $A$ 1's, then write $Z$, then pop $A$ 1's (writing them), then pop the stack start symbol. This gives us the output $H_0^a Z_a^0$, for any $A$

3. If we go from A to B writing $H$, then go to E writing $b$ 0's, then write $Z$, then pop $b$ 0's (writing them), then pop the stack start symbol, this gives us the output $H_b^0 Z_0^b$, for any $b$.

4. If we go from A to B writing $H$, then write $A$ 1's, then go to E writing $b$ 0's, then write $Z$, then pop $A$ 1's (writing them), pop $b$ 0's (writing them), then pop the stack start symbol, this gives us the output $H_b^a Z_a^b$, for any $A$ and $b$.

Notice that the machine will only generate legal words in our language, because any other word will not lead us to the terminating state, and hence is not recognized by the machine.

We are now ready to write a formal definition for the NDPDA for the language $L$. Our notation is taken from [79].

A pushdown automaton is a system $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

1. $Q$ is a finite set of states
2. $\Sigma$ is an alphabet called the input alphabet[4]
3. $\Gamma$ is an alphabet called the stack alphabet
4. $q_0 \in Q$ is initial state
5. $Z_0 \in \Gamma$ is a stack symbol called the start symbol
6. $F \subseteq Q$ is the set of final states
7. $\delta$ is a mapping from $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ to finite subsets of $Q \times \Gamma*$

Our NDPDA as shown in the figure above is formally defined as follows:

$$\begin{aligned}
&Q = \{A, B, C, D, E, F\}\\
&\Sigma = \{H, Z, 1, 0\}\\
&\Gamma = \{Z_0, 1, 0\}\\
&A = q_0\\
&F = \{F\}
\end{aligned}$$

And the mapping $\delta$ is as follows:

$$\begin{aligned}
&\delta(A, H, \epsilon) = (B, Z_0)\\
&\delta(B, Z, \epsilon) = (C, Z_(0))\\
&\delta(C, \epsilon, Z_0) = (F, \epsilon)\\
&\delta(B, 1, 1) = (B, 11)\\
&\delta(B, 0, 1) = (E, 10)\\
&\delta(E, 0, 0) = (E, 00)\\
&\delta(E, Z, 0) = (C, 0)\\
&\delta(C, 0, 0) = (C, \epsilon)\\
&\delta(C, 1, 1) = (D, \epsilon)\\
&\delta(D, 1, 1) = (D, \epsilon)\\
&\delta(D, Z, Z_0) = (F, \epsilon)\\
&\delta(C, \epsilon, Z_0) = (F, \epsilon)\\
&\delta(D, \epsilon, Z_0) = (F, \epsilon)
\end{aligned}$$

In the definition of $\delta$, a "push" occurs when there are two alphabet characters on the stack (for example, $\{(E, 10)\}$ ) , and a "pop" occurs when there is an empty string $\epsilon$ which "erases" the top of the stack (for example, $\{(C, \epsilon)\}$).

[4]In our case, it will be the output alphabet because we are generating the language instead of recognizing it.

Let us now review. We have shown that we can think of a series expansion as a sequence of words from a language, and that for some simple expansions we can define finite state machines or non-deterministic pushdown automata that recognize such a language. Moreover, such machines can be defined as data, not control flow—therefore they achieve one of the objectives we have for our generalization effort. However, it appears clear at this point that expressing constraints in language-theoretic terms is just as complex as expressing them in program control flow—therefore we have not achieved our objective of additional simplicity and understandability. For this reason, we discontinued looking at formal language theory, and pursued a second option.

## C.4 Complexity of linear programming

The question will arise as to the complexity of linear programming—in other words, how efficient linear programming is as a method for solving constraints. We are interested in both the space (memory) and time requirements; further, we are interested both in the worst-case and average-case performance.

Complexity of linear programming is a long-studied problem; indeed, the basic result that integer linear programming is NP-complete was known as far back as Karp's paper on 21 NP-complete problems [81]. An NP-complete problem is one for which there is no known algorithm that, in the worst case, does not take time exponential to the size of the input. Thus, it may seem that we should be concerned about this approach. However, the situation is not this simple.

Consider first of all (non-integer) linear programming. This problem is known to be solvable in polynomial time via Khachiyan's ellipsoid algorithm [82]. However Khachiyan's algorithm is not practical to implement. Dantzig's simplex method, on the other hand, is known to be exponential in the worst case, especially for problems constructed specifically for their difficulty, such as Klee-Minty cubes [83]. But in practical use, the time complexity of the simplex algorithm appears to be linear, and hence it continues to be a popular algorithm.

We stated that the general integer linear programming problem is NP-complete. However, more restricted versions of the integer linear programming problem are polynomial. Lenstra showed that integer linear programming with a fixed number of variables is polynomial in time [84]. Lenstra's algorithm is exponential in the number of variables, but so long as the number of variables is fixed, then this becomes simply a large constant term. In our use, we expect there to be a bound on the number of variables, simply because we have to bound our expansions as well, and there are only so many terms we wish to compute. Thus, conceivably we could use Lenstra's algorithm and ensure polynomial time. However, as was the case with Khachiyan's algorithm, Lenstra's is more theoretical than practical (in particular, Lenstra's algorithm requires exponential space), and simplex method may still in practice be more efficient.

What about space complexity? A result by Lokshtanov indicates that integer linear programming requires space polynomial in the number of bits of the input [85]. The simplex method is also PSPACE-complete [86].

In summary: while integer linear programming can, in some cases, be exponential, it appears that for certain restricted subsets of the problem, it can be bounded to linear time and polynomial space; further, that the simplex algorithm is a widely-available algorithm that also performs efficiently for many practical problems. Thus, the best course of action is to use the simplex method to solve our ILPs unless and until we can demonstrate that we are producing a problem at which simplex proves to be too expensive in time.

# Appendix D

# Algorithms

In Section 2.2.4 results from a MH simulation are presented. The code used to generate those results is given in Listing D.1.

```python
# import packages
import matplotlib as mpl; mpl.use('pdf');
import matplotlib.pyplot as plt
from scipy.stats import norm, uniform
from scipy.integrate import quad
import numpy as np

# if choosing initial point randomly
from numpy.random import default_rng
rng = default_rng()


def target_density(x, mean=0, std_dev=1):
    """ The p.d.f. we are trying to approximate. """
    return_val = (
        (np.sin(x) ** 2.)
        * (np.sin(2 * x) ** 2.)
        * norm.pdf(x, loc=mean, scale=std_dev)
    )

    return return_val


def proposal_density(x_old, alpha):
    """ The p.d.f. we use to propose new moves/samples/steps.
    The uniform distribution should be distributed [x - alpha, x + alpha]
    the way scipy uses scale is [loc, loc+scale]
    so then we need scale = 2 * alpha
    """

    # scipy returns random values that are normalized
    return uniform.rvs(loc=x_old - alpha, scale=2. * alpha)


def metropolis(x_old, alpha=1.):
    """Take a step / increment the Markov-Chain forward 1 link """

    proposed_x = proposal_density(x_old, alpha)

    # acceptance ratio
    ratio = target_density(proposed_x) / target_density(x_old)

    # generate random variable between [0, 1]
    u = uniform.rvs(loc=0, scale=1)

    # accept
    if bool(u <= ratio):
        x_new = proposed_x

    # or reject
    else:
        x_new = x_old

    return x_new


def plot(x_simulated, n_points, alpha, xmin=-4.0, xmax=4.0):
    """ plot the distribution and the histogram of the results """

    plt.rcParams['text.usetex'] = True
    plt.rcParams['font.size'] = '12'

    # define a grid to evaluate the distribution on
    x_grid = np.arange(xmin, xmax, step=0.005)

    # compute normalization factor for target distribution
    norm_factor, abserr = quad(target_density, xmin, xmax)
    y_analytic = target_density(x_grid) / norm_factor

    fig, ax = plt.subplots()
    ax.plot(
        x_grid, y_analytic, label=r'\(\pi(x)\)',
        color='r', linestyle='--')

    n, bins, patches = plt.hist(
        x_simulated, 100, label=r'\(\tilde{\pi}(x)\)',
        density=True, edgecolor='black', facecolor='lightgrey', alpha=0.7,
    )

    ax.set(
        xlabel='x', ylabel='Density',
        # title='Illustrative approximation of a distribution using Metropolis Hastings'
    )
    ax.legend(loc='best')
    ax.grid()

    fig.savefig(f"mh_simulation_a{int(alpha):d}_N{n_points}.png", dpi=175)
    # plt.show()

    return


def simple_example():
    # set parameters
    max_steps = int(1e4)
    alpha = 2.0

    # store our metropolis steps in here
    chain = np.zeros(max_steps)

    # choose manually
    if True:
        chain[0] = 3  # 2.783

    # choose randomly
    else:
        chain[0] = rng.uniform(-alpha, alpha)
        print(f"Random initial value: {chain[0]}")

    # simulate the distribution
    for t in range(1, max_steps):
        chain[t] = metropolis(chain[t-1], alpha)

    plot(chain, max_steps, alpha)


if (__name__ == '__main__'):
    simple_example()
```

Listing D.1: Example of Python script for approximating a modified normal distribution using a Metropolis-Hastings approach

# Appendix E

# Example Files

These appendices contain various files that are referenced in the thesis. Appendices E.1 and E.2 contain output from `termfactory`. Listing E.2 is the output of the diabatization example outlined in Section 3.5. Two vibronic models are defined in Sections 3.4.2 and 3.4.3 and their corresponding file representations are presented in Listings E.3 and E.5 and Listings E.4 and E.6.

## E.1 `termfactory` $e^{\hat{T}}\hat{H}\hat{Z}$ LaTeX example output

This appendix is intended to be illustrative of the type of output generated by `termfactory` that is used by `t-amplitudes` to generate spectra, and is more complicated than the examples in Chapter 5. The magenta colouring is a visual aid for pairings between **`t`** and **`z`**. Generating this is as simple as `python3 driver.py -t 2 3 1 3 2` and takes $\approx 0.2$ seconds.

### E.1.1 Constant Equations $\hat{P}_0$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}_\gamma}{\mathrm{d}\tau}\Big) \\
RHS &= \mathbf{h}_{0,xb}(1-\delta_{x\gamma}) \\
&\quad + \sum\Big(\mathbb{1}\mathbf{h}_0\mathbf{z}_0 + f\mathbb{1}\mathbf{h}_k\mathbf{z}^k + f^2\frac{2!}{2!2!}\mathbb{1}\mathbf{h}_{kl}\mathbf{z}^{kl} + f\mathbf{t}_k\mathbf{h}_0\mathbf{z}^k \\
&\quad + f^2\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}_l\mathbf{z}^{kl} + f^3\frac{(3)2!}{2!3!}\mathbf{t}_k\mathbf{h}_{lm}\mathbf{z}^{klm} + f\mathbf{t}_k\mathbf{h}^k\mathbf{z}_0 + f^2\mathbf{t}_k\mathbf{h}_l^k\mathbf{z}^l \\
&\quad + f^2\frac{2!}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_0\mathbf{z}^{kl} + f^3\frac{(3)2!}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_m\mathbf{z}^{klm} + f^2\frac{1}{2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^l\mathbf{z}^k + f^3\frac{(2)}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_m^l\mathbf{z}^{km} \\
&\quad + f^2\frac{2!}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}_0 + f^3\frac{3!}{3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}_0\mathbf{z}^{klm} + f^3\frac{2!}{2!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^m\mathbf{z}^{kl} + f^4\frac{(3)2!}{3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}_n^m\mathbf{z}^{kln} \\
&\quad + f^3\frac{2!}{2!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^k\Big) \\
&\quad - i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma}\Big)
\end{aligned}
$$

### E.1.2 Linear Equations $\hat{P}_y$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}^i_\gamma}{\mathrm{d}\tau}\Big) \\
RHS &= \mathbf{h}^i(1-\delta_{x\gamma}) \\
&\quad + \sum\Big(\mathbb{1}\mathbf{h}_0\mathbf{z}^y + f\frac{(2)}{2!}\mathbb{1}\mathbf{h}_k\mathbf{z}^{ky} + f^2\frac{(3)2!}{2!3!}\mathbb{1}\mathbf{h}_{kl}\mathbf{z}^{kly} + \mathbb{1}\mathbf{h}^y\mathbf{z}_0 \\
&\quad + f\mathbb{1}\mathbf{h}_k^y\mathbf{z}^k + f\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}_0\mathbf{z}^{ky} + f^2\frac{(3)(2)}{3!}\mathbf{t}_k\mathbf{h}_l\mathbf{z}^{kly} + f\mathbf{t}_k\mathbf{h}^y\mathbf{z}^k \\
&\quad + f\mathbf{t}_k\mathbf{h}^k\mathbf{z}^y + f^2\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}_l^y\mathbf{z}^{kl} + f^2\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}_l^k\mathbf{z}^{ly} + f\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}^{ky}\mathbf{z}_0 \\
&\quad + f^2\frac{(3)2!}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_0\mathbf{z}^{kly} + f^2\frac{2!}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^y\mathbf{z}^{kl} + f^2\frac{(2)}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^l\mathbf{z}^{ky} + f^3\frac{(3)2!}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_m^y\mathbf{z}^{klm} \\
&\quad + f^3\frac{(3)(2)}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_m^l\mathbf{z}^{kmy} + f^2\frac{(2)}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{ly}\mathbf{z}^k + f^2\frac{2!}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}^y + f^3\frac{3!}{3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^y\mathbf{z}^{klm} \\
&\quad + f^3\frac{(3)2!}{3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^m\mathbf{z}^{kly} + f^3\frac{(2)2!}{2!2!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{my}\mathbf{z}^{kl} + f^3\frac{2!(2)}{2!2!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^{ky}\Big) \\
&\quad - i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}^i_\gamma}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^i_\gamma\Big)
\end{aligned}
$$

### E.1.3 Quadratic Equations $\hat{P}_{yx}$

$\hat{P}_{yx}$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}^{ij}_{\gamma}}{\mathrm{d}\tau}\Big)\\
RHS &= \mathbf{h}^{ij}(1-\delta_{x\gamma})\\
&\quad + \sum\Big(\frac{1}{2!}\mathbb{1}\mathbf{h}_0\mathbf{z}^{yx} + f\frac{(3)}{3!}\mathbb{1}\mathbf{h}_k\mathbf{z}^{kyx} + \mathbb{1}\mathbf{h}^{y}\mathbf{z}^{x} + f\frac{(2)}{2!}\mathbb{1}\mathbf{h}^{y}_{k}\mathbf{z}^{kx}\\
&\quad + \frac{1}{2!}\mathbb{1}\mathbf{h}^{yx}\mathbf{z}_0 + f\frac{(3)}{3!}\mathbf{t}_k\mathbf{h}_0\mathbf{z}^{kyx} + f\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}^{y}\mathbf{z}^{kx} + f\frac{1}{2!}\mathbf{t}_k\mathbf{h}^{k}\mathbf{z}^{yx}\\
&\quad + f^2\frac{(3)(2)}{3!}\mathbf{t}_k\mathbf{h}^{y}_{l}\mathbf{z}^{klx} + f^2\frac{(3)}{3!}\mathbf{t}_k\mathbf{h}^{k}_{l}\mathbf{z}^{lyx} + f\frac{1}{2!}\mathbf{t}_k\mathbf{h}^{yx}\mathbf{z}^{k} + f\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}^{ky}\mathbf{z}^{x}\\
&\quad + f^2\frac{(3)2!}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{y}\mathbf{z}^{klx} + f^2\frac{(3)}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}\mathbf{z}^{kyx} + f^2\frac{2!}{2!2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{yx}\mathbf{z}^{kl} + f^2\frac{(2)(2)}{2!2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{ly}\mathbf{z}^{kx}\\
&\quad + f^2\frac{2!}{2!2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}^{yx} + f^3\frac{3!}{2!3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{yx}\mathbf{z}^{klm} + f^3\frac{(2)(3)2!}{2!3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{my}\mathbf{z}^{klx} + f^3\frac{2!(3)}{2!3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^{kyx}\Big)\\
&\quad - i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}^{ij}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{i}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{j}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{ij}_{\gamma}\Big)
\end{aligned}
$$

## E.2 Extreme `termfactory` $e^{\hat{T}}\hat{H}\hat{Z}$ LaTeX example output

This appendix is intended to be illustrative of the extreme end of LaTeX generated by **`termfactory`**. Note that equations this extensive are not used in **`t-amplitudes`**. Generating this is as simple as `python3 driver.py -t 2 5 1 6 6` and takes roughly 1.5 seconds. The magenta colouring is a visual aid for pairings between **`t`** and **`z`**.

### E.2.1 Constant Equations $\hat{P}_0$

$$
\begin{aligned}
LHS &= i\left(\frac{\mathrm{d}\hat{\mathbf{z}}_{\gamma}}{\mathrm{d}\tau}\right)\\
RHS &= \mathbf{h}_{0,xb}(1-\delta_{x\gamma})\\
&\quad+\sum\Bigg(\mathbb{1}\mathbf{h}_0\mathbf{z}_0+\bar{f}\mathbb{1}\mathbf{h}_k\mathbf{z}^{k}+\bar{f}^2\frac{2!}{2!2!}\mathbb{1}\mathbf{h}_{kl}\mathbf{z}^{kl}+\bar{f}\mathbf{t}_k\mathbf{h}_0\mathbf{z}^{k}\\
&\quad+\bar{f}^2\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}_l\mathbf{z}^{kl}+\bar{f}^3\frac{(3)2!}{2!3!}\mathbf{t}_k\mathbf{h}_{lm}\mathbf{z}^{klm}+\bar{f}\mathbf{t}_k\mathbf{h}^{k}\mathbf{z}_0+\bar{f}^2\mathbf{t}_k\mathbf{h}^{k}_{l}\mathbf{z}^{l}\\
&\quad+\bar{f}^2\frac{2!}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_0\mathbf{z}^{kl}+\bar{f}^3\frac{(3)2!}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_m\mathbf{z}^{klm}+\bar{f}^4\frac{(6)2!2!}{2!4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_{mn}\mathbf{z}^{klmn}+\bar{f}^2\frac{1}{2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}\mathbf{z}^{k}\\
&\quad+\bar{f}^3\frac{(2)}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}_{m}\mathbf{z}^{km}+\bar{f}^2\frac{2!}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}_0+\bar{f}^3\frac{3!}{3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}_0\mathbf{z}^{klm}+\bar{f}^4\frac{(4)3!}{4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}_n\mathbf{z}^{klmn}\\
&\quad+\bar{f}^5\frac{(10)2!3!}{2!5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}_{no}\mathbf{z}^{klmno}+\bar{f}^3\frac{2!}{2!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{m}\mathbf{z}^{kl}+\bar{f}^4\frac{(3)2!}{3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{m}_{n}\mathbf{z}^{kln}+\bar{f}^3\frac{2!}{2!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^{k}\\
&\quad+\bar{f}^4\frac{4!}{4!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}_0\mathbf{z}^{klmn}+\bar{f}^5\frac{(5)4!}{5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}_o\mathbf{z}^{klmno}+\bar{f}^4\frac{3!}{3!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{n}\mathbf{z}^{klm}+\bar{f}^5\frac{(4)3!}{4!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{n}_{o}\mathbf{z}^{klmo}\\
&\quad+\bar{f}^4\frac{2!2!}{2!2!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{mn}\mathbf{z}^{kl}+\bar{f}^5\frac{5!}{5!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}_0\mathbf{z}^{klmno}+\bar{f}^5\frac{4!}{4!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{o}\mathbf{z}^{klmn}+\bar{f}^6\frac{(5)4!}{5!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{o}\mathbf{z}^{klmn}\\
&\quad+\bar{f}^5\frac{2!3!}{2!3!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{no}\mathbf{z}^{klm}+\bar{f}^6\frac{5!}{5!6!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{t}\mathbf{h}\mathbf{z}^{klmno}+\bar{f}^6\frac{2!4!}{2!4!6!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{t}\mathbf{h}^{o}\mathbf{z}^{klmn}\Bigg)\\
&\quad-i\sum\left(\frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma}\right)
\end{aligned}
$$

### E.2.2 Linear Equations $\hat{P}_y$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}^i_\gamma}{\mathrm{d}\tau}\Big)\\
RHS &= \mathbf{h}^i(1-\delta_{x\gamma})\\
&\quad+\sum\Big(\mathbb{1}\mathbf{h}_0\mathbf{z}^{y}+\bar{f}\frac{(2)}{2!}\mathbb{1}\mathbf{h}_k\mathbf{z}^{ky}+\bar{f}^2\frac{(3)2!}{2!3!}\mathbb{1}\mathbf{h}_{kl}\mathbf{z}^{kly}+\mathbb{1}\mathbf{h}^{y}\mathbf{z}_0\\
&\quad+\bar{f}\mathbb{1}\mathbf{h}^{y}_{k}\mathbf{z}^{k}+\bar{f}\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}_0\mathbf{z}^{ky}+\bar{f}^2\frac{(3)(2)}{3!}\mathbf{t}_k\mathbf{h}_l\mathbf{z}^{kly}+\bar{f}^3\frac{(4)(3)2!}{2!4!}\mathbf{t}_k\mathbf{h}_{lm}\mathbf{z}^{klmy}\\
&\quad+\bar{f}\mathbf{t}_k\mathbf{h}^{y}\mathbf{z}^{k}+\bar{f}\mathbf{t}_k\mathbf{h}^{k}\mathbf{z}^{y}+\bar{f}^2\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}^{y}_{l}\mathbf{z}^{kl}+\bar{f}^2\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}^{k}_{l}\mathbf{z}^{ly}\\
&\quad+\bar{f}\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}^{ky}\mathbf{z}_0+\bar{f}^2\frac{(3)2!}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_0\mathbf{z}^{kly}+\bar{f}^3\frac{(4)(3)2!}{4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_m\mathbf{z}^{klmy}+\bar{f}^4\frac{(5)(6)2!2!}{2!5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_{mn}\mathbf{z}^{klmny}\\
&\quad+\bar{f}^2\frac{2!}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{y}\mathbf{z}^{kl}+\bar{f}^2\frac{(2)}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}\mathbf{z}^{ky}+\bar{f}^3\frac{(3)2!}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{y}_{m}\mathbf{z}^{klm}+\bar{f}^3\frac{(3)(2)}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}_{m}\mathbf{z}^{kmy}\\
&\quad+\bar{f}^2\frac{(2)}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{ly}\mathbf{z}^{k}+\bar{f}^2\frac{2!}{2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}^{y}+\bar{f}^3\frac{(4)3!}{4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}_0\mathbf{z}^{klmy}+\bar{f}^4\frac{(5)(4)3!}{5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}_n\mathbf{z}^{klmny}\\
&\quad+\bar{f}^3\frac{3!}{3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{y}\mathbf{z}^{klm}+\bar{f}^3\frac{(3)2!}{3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{m}\mathbf{z}^{kly}+\bar{f}^4\frac{(4)3!}{4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{y}_{n}\mathbf{z}^{klmn}+\bar{f}^4\frac{(4)(3)2!}{4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{m}_{n}\mathbf{z}^{klny}\\
&\quad+\bar{f}^3\frac{(2)2!}{2!2!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{my}\mathbf{z}^{kl}+\bar{f}^3\frac{2!(2)}{2!2!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^{ky}+\bar{f}^4\frac{(5)4!}{5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}_0\mathbf{z}^{klmny}+\bar{f}^4\frac{4!}{4!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{y}\mathbf{z}^{klmn}\\
&\quad+\bar{f}^4\frac{(4)3!}{4!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{n}\mathbf{z}^{klmy}+\bar{f}^5\frac{(5)4!}{5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{y}_{o}\mathbf{z}^{klmno}+\bar{f}^5\frac{(5)(4)3!}{5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{n}_{o}\mathbf{z}^{klmoy}+\bar{f}^4\frac{(2)3!}{2!3!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{ny}\mathbf{z}^{klm}\\
&\quad+\bar{f}^4\frac{2!(3)2!}{2!3!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{mn}\mathbf{z}^{kly}+\bar{f}^5\frac{5!}{5!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{y}\mathbf{z}^{klmno}+\bar{f}^5\frac{(5)4!}{5!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{o}\mathbf{z}^{klmny}+\bar{f}^5\frac{(2)4!}{2!4!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{oy}\mathbf{z}^{klmn}\\
&\quad+\bar{f}^5\frac{2!(4)3!}{2!4!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{no}\mathbf{z}^{klmy}+\bar{f}^6\frac{(2)5!}{2!5!6!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{t}\mathbf{h}^{y}\mathbf{z}^{klmno}+\bar{f}^6\frac{2!(5)4!}{2!5!6!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{t}\mathbf{h}^{o}\mathbf{z}^{klmny}\Big)\\
&\quad-i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}^i_\gamma}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma}+\frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^i_\gamma\Big)
\end{aligned}
$$

### E.2.3 Quadratic Equations $\hat{P}_{yx}$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}^{ij}_{\gamma}}{\mathrm{d}\tau}\Big)\\
RHS &= \mathbf{h}^{ij}(1-\delta_{x\gamma})\\
&\quad + \sum\Big(\frac{1}{2!}\mathbb{1}\mathbf{h}_0\mathbf{z}^{yx} + \bar{f}\frac{(3)}{3!}\mathbb{1}\mathbf{h}_k\mathbf{z}^{kyx} + \bar{f}^2\frac{(6)2!}{2!4!}\mathbb{1}\mathbf{h}_{kl}\mathbf{z}^{klyx} + \mathbb{1}\mathbf{h}^{y}\mathbf{z}^{x}\\
&\quad + \bar{f}\frac{(2)}{2!}\mathbb{1}\mathbf{h}^{y}_{k}\mathbf{z}^{kx} + \frac{1}{2!}\mathbb{1}\mathbf{h}^{yx}\mathbf{z}_0 + \bar{f}\frac{(3)}{3!}\mathbf{t}_k\mathbf{h}_0\mathbf{z}^{kyx} + \bar{f}^2\frac{(6)(2)}{4!}\mathbf{t}_k\mathbf{h}_l\mathbf{z}^{klyx}\\
&\quad + \bar{f}^3\frac{(10)(3)2!}{2!5!}\mathbf{t}_k\mathbf{h}_{lm}\mathbf{z}^{klmyx} + \bar{f}\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}^{y}\mathbf{z}^{kx} + \bar{f}\frac{1}{2!}\mathbf{t}_k\mathbf{h}^{k}\mathbf{z}^{yx} + \bar{f}^2\frac{(3)(2)}{3!}\mathbf{t}_k\mathbf{h}^{y}_{l}\mathbf{z}^{klx}\\
&\quad + \bar{f}^2\frac{(3)}{3!}\mathbf{t}_k\mathbf{h}^{k}_{l}\mathbf{z}^{lyx} + \bar{f}\frac{1}{2!}\mathbf{t}_k\mathbf{h}^{yx}\mathbf{z}^{k} + \bar{f}\frac{(2)}{2!}\mathbf{t}_k\mathbf{h}^{ky}\mathbf{z}^{x} + \bar{f}^2\frac{(6)2!}{4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_0\mathbf{z}^{klyx}\\
&\quad + \bar{f}^3\frac{(10)(3)2!}{5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_m\mathbf{z}^{klmyx} + \bar{f}^2\frac{(3)2!}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{y}\mathbf{z}^{klx} + \bar{f}^2\frac{(3)}{3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}\mathbf{z}^{kyx} + \bar{f}^3\frac{(4)(3)2!}{4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{y}_{m}\mathbf{z}^{klmx}\\
&\quad + \bar{f}^3\frac{(6)(2)}{4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}_{m}\mathbf{z}^{kmyx} + \bar{f}^2\frac{2!}{2!2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{yx}\mathbf{z}^{kl} + \bar{f}^2\frac{(2)(2)}{2!2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{ly}\mathbf{z}^{kx} + \bar{f}^2\frac{2!}{2!2!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}^{yx}\\
&\quad + \bar{f}^3\frac{(10)3!}{5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}_0\mathbf{z}^{klmyx} + \bar{f}^3\frac{(4)3!}{4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{y}\mathbf{z}^{klmx} + \bar{f}^3\frac{(6)2!}{4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{m}\mathbf{z}^{klyx} + \bar{f}^4\frac{(5)(4)3!}{5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{y}_{n}\mathbf{z}^{klmnx}\\
&\quad + \bar{f}^4\frac{(10)(3)2!}{5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{m}_{n}\mathbf{z}^{klnyx} + \bar{f}^3\frac{3!}{2!3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{yx}\mathbf{z}^{klm} + \bar{f}^3\frac{(2)(3)2!}{2!3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{my}\mathbf{z}^{klx} + \bar{f}^3\frac{2!(3)}{2!3!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^{kyx}\\
&\quad + \bar{f}^4\frac{(5)4!}{5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{y}\mathbf{z}^{klmnx} + \bar{f}^4\frac{(10)3!}{5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{n}\mathbf{z}^{klmyx} + \bar{f}^4\frac{4!}{2!4!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{yx}\mathbf{z}^{klmn} + \bar{f}^4\frac{(2)(4)3!}{2!4!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{ny}\mathbf{z}^{klmx}\\
&\quad + \bar{f}^4\frac{2!(6)2!}{2!4!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{mn}\mathbf{z}^{klyx} + \bar{f}^5\frac{5!}{2!5!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{yx}\mathbf{z}^{klmno} + \bar{f}^5\frac{(2)(5)4!}{2!5!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{oy}\mathbf{z}^{klmnx} + \bar{f}^5\frac{2!(10)3!}{2!5!5!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{t}_o\mathbf{h}^{no}\mathbf{z}^{klmyx}\Big)\\
&\quad - i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}^{ij}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{i}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{j}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{ij}_{\gamma}\Big)
\end{aligned}
$$

### E.2.4 Cubic Equations $\hat{P}_{yxw}$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}^{ijk}_{\gamma}}{\mathrm{d}\tau}\Big)\\
RHS &= \mathbf{h}^{ijk}(1-\delta_{x\gamma})\\
&\quad + \sum\Big(\frac{1}{3!}\mathbb{1}\mathbf{h}_0\mathbf{z}^{yxw} + \bar{f}\frac{(4)}{4!}\mathbb{1}\mathbf{h}_k\mathbf{z}^{kyxw} + \bar{f}^2\frac{(10)2!}{2!5!}\mathbb{1}\mathbf{h}_{kl}\mathbf{z}^{klyxw} + \frac{1}{2!}\mathbb{1}\mathbf{h}^{y}\mathbf{z}^{xw}\\
&\quad + \bar{f}\frac{(3)}{3!}\mathbb{1}\mathbf{h}^{y}_{k}\mathbf{z}^{kxw} + \frac{1}{2!}\mathbb{1}\mathbf{h}^{yx}\mathbf{z}^{w} + \bar{f}\frac{(4)}{4!}\mathbf{t}_k\mathbf{h}_0\mathbf{z}^{kyxw} + \bar{f}^2\frac{(10)(2)}{5!}\mathbf{t}_k\mathbf{h}_l\mathbf{z}^{klyxw}\\
&\quad + \bar{f}\frac{(3)}{3!}\mathbf{t}_k\mathbf{h}^{y}\mathbf{z}^{kxw} + \bar{f}\frac{1}{3!}\mathbf{t}_k\mathbf{h}^{k}\mathbf{z}^{yxw} + \bar{f}^2\frac{(6)(2)}{4!}\mathbf{t}_k\mathbf{h}^{y}_{l}\mathbf{z}^{klxw} + \bar{f}^2\frac{(4)}{4!}\mathbf{t}_k\mathbf{h}^{k}_{l}\mathbf{z}^{lyxw}\\
&\quad + \bar{f}\frac{(2)}{2!2!}\mathbf{t}_k\mathbf{h}^{yx}\mathbf{z}^{kw} + \bar{f}\frac{(2)}{2!2!}\mathbf{t}_k\mathbf{h}^{ky}\mathbf{z}^{xw} + \bar{f}^2\frac{(10)2!}{5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_0\mathbf{z}^{klyxw} + \bar{f}^2\frac{(6)2!}{4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{y}\mathbf{z}^{klxw}\\
&\quad + \bar{f}^2\frac{(4)}{4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}\mathbf{z}^{kyxw} + \bar{f}^3\frac{(10)(3)2!}{5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{y}_{m}\mathbf{z}^{klmxw} + \bar{f}^3\frac{(10)(2)}{5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{l}_{m}\mathbf{z}^{kmyxw} + \bar{f}^2\frac{(3)2!}{2!3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{yx}\mathbf{z}^{klw}\\
&\quad + \bar{f}^2\frac{(2)(3)}{2!3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{ly}\mathbf{z}^{kxw} + \bar{f}^2\frac{2!}{2!3!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}^{yxw} + \bar{f}^3\frac{(10)3!}{5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{y}\mathbf{z}^{klmxw} + \bar{f}^3\frac{(10)2!}{5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{m}\mathbf{z}^{klyxw}\\
&\quad + \bar{f}^3\frac{(4)3!}{2!4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{yx}\mathbf{z}^{klmw} + \bar{f}^3\frac{(2)(6)2!}{2!4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{my}\mathbf{z}^{klxw} + \bar{f}^3\frac{2!(4)}{2!4!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^{kyxw} + \bar{f}^4\frac{(5)4!}{2!5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{yx}\mathbf{z}^{klmnw}\\
&\quad + \bar{f}^4\frac{(2)(10)3!}{2!5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{ny}\mathbf{z}^{klmxw} + \bar{f}^4\frac{2!(10)2!}{2!5!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{mn}\mathbf{z}^{klyxw}\Big)\\
&\quad - i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}^{ijk}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{ij}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{k}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{i}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{jk}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{ijk}_{\gamma}\Big)
\end{aligned}
$$

### E.2.5 Quartic Equations $\hat{P}_{yxwv}$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}^{ijkl}_{\gamma}}{\mathrm{d}\tau}\Big)\\
RHS &= \mathbf{h}^{ijkl}(1-\delta_{x\gamma})\\
&+\sum\Big(\frac{1}{4!}\mathbb{1}\mathbf{h}_0\mathbf{z}^{yxwv}+\bar{f}\frac{(5)}{5!}\mathbb{1}\mathbf{h}_k\mathbf{z}^{kyxwv}+\bar{f}^2\frac{(15)2!}{2!6!}\mathbb{1}\mathbf{h}_{kl}\mathbf{z}^{klyxwv}+\frac{1}{3!}\mathbb{1}\mathbf{h}^y\mathbf{z}^{xwv}\\
&+\bar{f}\frac{(4)}{4!}\mathbb{1}\mathbf{h}^y_k\mathbf{z}^{kxwv}+\frac{1}{2!2!}\mathbb{1}\mathbf{h}^{yx}\mathbf{z}^{wv}+\bar{f}\frac{(5)}{5!}\mathbf{t}_k\mathbf{h}_0\mathbf{z}^{kyxwv}+\bar{f}^2\frac{(15)(2)}{6!}\mathbf{t}_k\mathbf{h}_l\mathbf{z}^{klyxwv}\\
&+\bar{f}\frac{(4)}{4!}\mathbf{t}_k\mathbf{h}^y\mathbf{z}^{kxwv}+\bar{f}\frac{1}{4!}\mathbf{t}_k\mathbf{h}^k\mathbf{z}^{yxwv}+\bar{f}^2\frac{(10)(2)}{5!}\mathbf{t}_k\mathbf{h}^y_l\mathbf{z}^{klxwv}+\bar{f}^2\frac{(5)}{5!}\mathbf{t}_k\mathbf{h}^k_l\mathbf{z}^{lyxwv}\\
&+\bar{f}\frac{(3)}{2!3!}\mathbf{t}_k\mathbf{h}^{yx}\mathbf{z}^{kwv}+\bar{f}\frac{(2)}{2!3!}\mathbf{t}_k\mathbf{h}^{ky}\mathbf{z}^{xwv}+\bar{f}^2\frac{(15)2!}{6!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}_0\mathbf{z}^{klyxwv}+\bar{f}^2\frac{(10)2!}{5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^y\mathbf{z}^{klxwv}\\
&+\bar{f}^2\frac{(5)}{5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^l\mathbf{z}^{kyxwv}+\bar{f}^3\frac{(20)(3)2!}{6!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^y_m\mathbf{z}^{klmxwv}+\bar{f}^3\frac{(15)(2)}{6!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^l_m\mathbf{z}^{kmyxwv}+\bar{f}^2\frac{(6)2!}{2!4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{yx}\mathbf{z}^{klwv}\\
&+\bar{f}^2\frac{(2)(4)}{2!4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{ly}\mathbf{z}^{kxwv}+\bar{f}^2\frac{2!}{2!4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}^{yxwv}+\bar{f}^3\frac{(20)3!}{6!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^y\mathbf{z}^{klmxwv}+\bar{f}^3\frac{(15)2!}{6!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^m\mathbf{z}^{klyxwv}\\
&+\bar{f}^3\frac{(10)3!}{2!5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{yx}\mathbf{z}^{klmwv}+\bar{f}^3\frac{(2)(10)2!}{2!5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{my}\mathbf{z}^{klxwv}+\bar{f}^3\frac{2!(5)}{2!5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^{kyxwv}+\bar{f}^4\frac{(15)4!}{2!6!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{yx}\mathbf{z}^{klmnwv}\\
&+\bar{f}^4\frac{(2)(20)3!}{2!6!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{ny}\mathbf{z}^{klmxwv}+\bar{f}^4\frac{2!(15)2!}{2!6!4!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{t}_n\mathbf{h}^{mn}\mathbf{z}^{klyxwv}\Big)\\
&-i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}^{ijkl}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma}+\frac{\mathrm{d}\hat{\mathbf{t}}^{ijk}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{l}_{\gamma}+\frac{\mathrm{d}\hat{\mathbf{t}}^{ij}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{kl}_{\gamma}+\frac{\mathrm{d}\hat{\mathbf{t}}^{i}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{jkl}_{\gamma}+\frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{ijkl}_{\gamma}\Big)
\end{aligned}
$$

### E.2.6 Quintic Equations $\hat{P}_{yxwvu}$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}^{ijkl}_{\gamma}}{\mathrm{d}\tau}\Big)\\
RHS &= \mathbf{h}^{ijkl}(1-\delta_{x\gamma})\\
&\quad + \sum\Big(\frac{1}{4!}\mathbb{1}\mathbf{h}_0\mathbf{z}^{yxwv} + \bar{f}\frac{(5)}{5!}\mathbb{1}\mathbf{h}_k\mathbf{z}^{kyxwv} + \frac{1}{3!}\mathbb{1}\mathbf{h}^y\mathbf{z}^{xwv} + \bar{f}\frac{(4)}{4!}\mathbb{1}\mathbf{h}^y_k\mathbf{z}^{kxwv}\\
&\quad + \frac{1}{2!2!}\mathbb{1}\mathbf{h}^{yx}\mathbf{z}^{wv} + \bar{f}\frac{(5)}{5!}\mathbf{t}_k\mathbf{h}_0\mathbf{z}^{kyxwv} + \bar{f}\frac{(4)}{4!}\mathbf{t}_k\mathbf{h}^y\mathbf{z}^{kxwv} + \bar{f}\frac{1}{4!}\mathbf{t}_k\mathbf{h}^k\mathbf{z}^{yxwv}\\
&\quad + \bar{f}^2\frac{(10)(2)}{5!}\mathbf{t}_k\mathbf{h}^y_l\mathbf{z}^{klxwv} + \bar{f}^2\frac{(5)}{5!}\mathbf{t}_k\mathbf{h}^k_l\mathbf{z}^{lyxwv} + \bar{f}\frac{(3)}{2!3!}\mathbf{t}_k\mathbf{h}^{yx}\mathbf{z}^{kwv} + \bar{f}\frac{(2)}{2!3!}\mathbf{t}_k\mathbf{h}^{ky}\mathbf{z}^{xwv}\\
&\quad + \bar{f}^2\frac{(10)2!}{5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^y\mathbf{z}^{klxwv} + \bar{f}^2\frac{(5)}{5!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^l\mathbf{z}^{kyxwv} + \bar{f}^2\frac{(6)2!}{2!4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{yx}\mathbf{z}^{klwv} + \bar{f}^2\frac{(2)(4)}{2!4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{ly}\mathbf{z}^{kxwv}\\
&\quad + \bar{f}^2\frac{2!}{2!4!2!}\mathbf{t}_k\mathbf{t}_l\mathbf{h}^{kl}\mathbf{z}^{yxwv} + \bar{f}^3\frac{(10)3!}{2!5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{yx}\mathbf{z}^{klmwv} + \bar{f}^3\frac{(2)(10)2!}{2!5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{my}\mathbf{z}^{klxwv} + \bar{f}^3\frac{2!(5)}{2!5!3!}\mathbf{t}_k\mathbf{t}_l\mathbf{t}_m\mathbf{h}^{lm}\mathbf{z}^{kyxwv}\Big)\\
&\quad - i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}^{ijkl}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}_{0,\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{ijk}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{l}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{ij}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{kl}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{i}_{\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{jkl}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\,\hat{\mathbf{z}}^{ijkl}_{\gamma}\Big)
\end{aligned}
$$

### E.2.7 Sextic Equations $\hat{P}_{yxwvu}$

$$
\begin{aligned}
LHS &= i\Big(\frac{\mathrm{d}\hat{\mathbf{z}}^{ijklm}_{\gamma}}{\mathrm{d}\tau}\Big)\\
RHS &= \mathbf{h}^{ijklm}(1-\delta_{x\gamma})\\
&\quad + \sum\Big(\frac{1}{5!}\mathbb{1}\mathbf{h}_0\mathbf{z}^{yxwvu} + \frac{1}{4!}\mathbb{1}\mathbf{h}^{y}\mathbf{z}^{xwvu} + \bar{f}\frac{(5)}{5!}\mathbb{1}\mathbf{h}^{y}_{k}\mathbf{z}^{kxwvu} + \frac{1}{2!3!}\mathbb{1}\mathbf{h}^{yx}\mathbf{z}^{wvu}\\
&\quad + \bar{f}\frac{(5)}{5!}\mathbf{t}_{k}\mathbf{h}^{y}\mathbf{z}^{kxwvu} + \bar{f}\frac{1}{5!}\mathbf{t}_{k}\mathbf{h}^{k}\mathbf{z}^{yxwvu} + \bar{f}\frac{(4)}{2!4!}\mathbf{t}_{k}\mathbf{h}^{yx}\mathbf{z}^{kwvu} + \bar{f}\frac{(2)}{2!4!}\mathbf{t}_{k}\mathbf{h}^{ky}\mathbf{z}^{xwvu}\\
&\quad + \bar{f}^2\frac{(10)2!}{2!5!2!}\mathbf{t}_{k}\mathbf{t}_{l}\mathbf{h}^{yx}\mathbf{z}^{klwvu} + \bar{f}^2\frac{(2)(5)}{2!5!2!}\mathbf{t}_{k}\mathbf{t}_{l}\mathbf{h}^{ly}\mathbf{z}^{kxwvu} + \bar{f}^2\frac{2!}{2!5!2!}\mathbf{t}_{k}\mathbf{t}_{l}\mathbf{h}^{kl}\mathbf{z}^{yxwvu}\Big)\\
&\quad - i\sum\Big(\frac{\mathrm{d}\hat{\mathbf{t}}^{ijklm}_{\gamma}}{\mathrm{d}\tau}\hat{\mathbf{z}}_{0,\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{ijkl}_{\gamma}}{\mathrm{d}\tau}\hat{\mathbf{z}}^{m}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{ijk}_{\gamma}}{\mathrm{d}\tau}\hat{\mathbf{z}}^{lm}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{ij}_{\gamma}}{\mathrm{d}\tau}\hat{\mathbf{z}}^{klm}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}^{i}_{\gamma}}{\mathrm{d}\tau}\hat{\mathbf{z}}^{jklm}_{\gamma} + \frac{\mathrm{d}\hat{\mathbf{t}}_{0,\gamma}}{\mathrm{d}\tau}\hat{\mathbf{z}}^{ijklm}_{\gamma}\Big)
\end{aligned}
$$

## E.3 Example MCTDH input file

```
RUN-SECTION
title = wavefunction propagation of h2o_FC_constant
name = h2o_FC_constant
propagation
tout = 0.10 tfinal = 25.00
geninwf
psi=single
auto=once
end-run-section

OPERATOR-SECTION
opname = h2o_FC_constant
end-operator-section

SPF-BASIS-SECTION
multi-set
      v01            =  3, 3, 6, 1
      v02, v03       =  3, 3, 6, 1
end-spf-basis-section


PRIMITIVE-BASIS-SECTION
    v01     HO      30    0.0    1.0    1.0
    v02     HO      30    0.0    1.0    1.0
    v03     HO      30    0.0    1.0    1.0
    el      el      4
end-primitive-basis-section

INTEGRATOR-SECTION
CMF/var     =   0.5,     1.0d-05
BS/spf      =     7,     1.0d-05,     2.5d-04
SIL/A       =     5,     1.0d-05
end-integrator-section

INIT_WF-SECTION
build
   init_state=4
-----------------------------------------------------------
#  mode    type   center   moment.   freq.     mass
-----------------------------------------------------------
    v01     HO      0.0     0.0       1.0      1.0
    v02     HO      0.0     0.0       1.0      1.0
    v03     HO      0.0     0.0       1.0      1.0
-----------------------------------------------------------
end-build
operate=Ex
end-init_wf-section

end-input
```

Listing E.1: Input file for MCTDH to propagate a ground state wavepacket for the linear vibronic model of$H_2O$

## E.4 Diabatization water model

```
OP_DEFINE-SECTION
title
water 3 modes 3 states
end-title
end-op_define-section

PARAMETER-SECTION

# frequencies
    w01  =        0.19867693  ,   ev
    w02  =        0.47347270  ,   ev
    w03  =        0.48650398  ,   ev

# vertical energies

    delta1  =        0.00000000  ,   ev
    delta2  =        1.53130000  ,   ev
    delta3  =        2.12440000  ,   ev

# linear intrastate parameters
    KD11_01  =        0.15126954  ,   ev
    KD11_02  =        0.13700266  ,   ev
    KD11_03  =       -0.16662290  ,   ev

    KD22_01  =        0.14413728  ,   ev
    KD22_02  =        0.13055576  ,   ev
    KD22_03  =       -0.12270191  ,   ev

    KD33_01  =        0.23522834  ,   ev
    KD33_02  =        0.21306226  ,   ev
    KD33_03  =        0.06221878  ,   ev

# linear interstate parameters
    KD12_01  =        0.08494233  ,   ev
    KD12_02  =        0.07689810  ,   ev
    KD12_03  =       -0.07576917  ,   ev

    KD13_01  =        0.27002321  ,   ev
    KD13_02  =       -0.12513436  ,   ev

    KD23_01  =        0.00001514  ,   ev
    KD23_02  =       -0.00000574  ,   ev
    KD23_03  =       -0.00004295  ,   ev

end-parameter-section

HAMILTONIAN-SECTION
modes |  el
modes |  v01
modes |  v02
modes |  v03
```

```

1.0*w01          |2   KE
1.0*w02          |3   KE
1.0*w03          |4   KE

0.5*w01          |2   q^2
0.5*w02          |3   q^2
0.5*w03          |4   q^2

delta1           |1  S1&1
delta2           |1  S2&2
delta3           |1  S3&3

KD11_01          |1  S1&1   |2   q
KD11_02          |1  S1&1   |3   q
KD11_03          |1  S1&1   |4   q

KD22_01          |1  S2&2   |2   q
KD22_02          |1  S2&2   |3   q
KD22_03          |1  S2&2   |4   q

KD33_01          |1  S3&3   |2   q
KD33_02          |1  S3&3   |3   q
KD33_03          |1  S3&3   |4   q


KD12_01          |1  S1&2   |2   q
KD12_02          |1  S1&2   |3   q
KD12_03          |1  S1&2   |4   q

KD13_01          |1  S1&3   |2   q
KD13_02          |1  S1&3   |3   q

KD23_01          |1  S2&3   |2   q
KD23_02          |1  S2&3   |3   q
KD23_03          |1  S2&3   |4   q


end-hamiltonian-section

end-operator
```

Listing E.2: Example of *.op file format of Water molecule ($H_2O$) obtained through diabatization process as explained in Section 3.5

## E.5 Displaced JSON

```json
{
    "number of modes": 2,
    "number of surfaces": 2,
    "energies":
    [
        [ 0.0996,   0.0    ],
        [ 0.0,      0.1996 ]
    ],
    "frequencies":
    [
        0.02, 0.04
    ],
    "linear couplings":
    [   [
            [   0.072,  0.0    ],
            [   0.0,   -0.072  ]
        ],[
            [   0.0,    0.04   ],
            [   0.04,   0.0    ]
        ]
    ]
}
```

Listing E.3: Example of JSON file format of Displaced model defined in Equation (3.38) with ($\lambda = 0.072$, $\gamma = 0.04$)

## E.6 Displaced op

```
OP_DEFINE-SECTION
title
IP
end-title
end-op_define-section


PARAMETER-SECTION


#        Frequencies
# ------------------------

w01                     =       0.02    , ev
w02                     =       0.04    , ev

#   Electronic Hamitonian
#   Vertical energies - Zeropoint
# -------------------------------

EH_s01_s01              =      0.0996                 , ev
EH_s02_s02              =      0.1996                 , ev

#  Linear Coupling Constants
# --------------------------

C1_s01_s01_v01          =       0.072     , ev
C1_s02_s02_v01          =      -0.072     , ev
C1_s01_s02_v02          =       0.04      , ev
C1_s02_s01_v02          =       0.04      , ev


end-parameter-section
```

Listing E.4: Example of *.op file format of Displaced model defined in Equation (3.38) with ($\lambda = 0.072$, $\gamma = 0.04$)

## E.7 Jahn-Teller JSON

```
{
    "number of modes": 2,
    "number of surfaces": 2,
    "energies":
    [
        [ -0.003333333333329107, 0.0 ],
        [ 0.0, -0.003333333333329107 ]
    ],
    "frequencies":
    [
        0.03, 0.03
    ],
    "linear couplings":
    [   [
            [   0.04,   0.0    ],
            [   0.0,   -0.04   ]
        ],[
            [   0.0,    0.04   ],
            [   0.04,   0.0    ]
        ]
    ]
}
```

Listing E.5: Example of JSON file format of Jahn-Teller model defined in Equation (3.39) with $\lambda = 0.072$, $\gamma = 0.04$

## E.8 Jahn-Teller op

```
OP_DEFINE-SECTION
title
IP
end-title
end-op_define-section


PARAMETER-SECTION


#         Frequencies
# ------------------------

w01                      =        0.03    , ev
w02                      =        0.03    , ev

#    Electronic Hamitonian
#    Vertical energies - Zeropoint
# -------------------------------

EH_s01_s01               =      -0.0033333333333329107     , ev
EH_s02_s02               =      -0.0033333333333329107     , ev

#   Linear Coupling Constants
# --------------------------

C1_s01_s01_v01           =        0.04       , ev
C1_s02_s02_v01           =       -0.04       , ev
C1_s01_s02_v02           =        0.04       , ev
C1_s02_s01_v02           =        0.04       , ev


end-parameter-section
```

Listing E.6: Example of *.op file format of Jahn-Teller model defined in Equation (3.39) with ($\lambda = 0.072$, $\gamma = 0.04$)