Convergence of the Light-Front Coupled-Cluster Method in Scalar Yukawa Theory

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Austin Usselman

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

John Hiller, Sophia Chabysheva

September, 2017

Acknowledgements

I want to keep my acknowledgments short. Thank you John Hiller and Sophia Chabysheva for guiding me along the way. My family is very important to me. I want to make sure they know that by including them here. My sister and brother-in-law, Elysa and Dave Faut, offered precious help even when they were building a family of their own. My father, Paul Usselman, was always there when I was in trouble whether it be my car giving me troubles or my back breaking down. My mother, Bonita Usselman, could not be more supportive of the decisions I've made with my life. I could not be more grateful. I am thankful for all the time I got to spend with my grandmother, Elsie Boehm. Hardships may make you want to quit. Family will help you reach your strive.

Dedication

I want to dedicate this to my closest friends. Not all my friends were students in physics but that does not make them important. Support and love can really get you to where you need to be. Talking with Ross Huber on the phone helped relieve stress and form a strong bond. Relieving stress through online games and talking about hobbies with Joseph Roth eased my mind when working too hard. Mitchell Zubitch has the best memory out of anyone I know and kept me in check with it. Keeping a healthy body is important during stressful college days and it would have been significantly worse if not for Jesse Higgins. The coding portion of this work was very difficult and I am gracious to Chris Hasse for helping where he did. Doing a thesis also involved completing classes and without doing homework with Miranda Elkins, I would not have made it through in two years. Having friends to take you out to make you feel not trapped in your apartment are important. Good thing Keegan Rabelhofer joined the beer tour at Old Chicago. Guests can help make a person feel wanted. Making plans and sticking to them to keep in touch with Tyler Antony really kept loneliness in check. Dillon Sailer and Jaeden Wellner always made sure to have a place for me to stay along the long trip home. I am grateful for all my wonderful and delightful friends. They could all make me laugh with Ethan “Big Daddy” Olsen making me laugh the hardest.

Abstract

We use Fock-state expansions and the Light-Front Coupled-Cluster (LFCC) method to study mass eigenvalue problems in quantum field theory. Specifically, we study convergence of the method in scalar Yukawa theory. In this theory, a single charged particle is surrounded by a cloud of neutral particles. The charged particle can create or annihilate neutral particles, causing the n-particle state to depend on the $n+1$ and $n-1$-particle state. Fock state expansion leads to an infinite set of coupled equations where truncation is required. The wave functions for the particle states are expanded in a basis of symmetric polynomials and a generalized eigenvalue problem is solved for the mass eigenvalue. The mass eigenvalue problem is solved for multiple values for the coupling strength while the number of particle states and polynomial basis order are increased. Convergence of the mass eigenvalue solutions is then obtained. Three mass ratios between the charged particle and neutral particles were studied. This includes a massive charged particle, equal masses and massive neutral particles. Relative probability between states can also be explored for more detailed understanding of the process of convergence with respect to the number of Fock sectors. The reliance on higher order particle states depended on how large the mass of the charge particle was. The higher the mass of the charged particle, the more the system depended on higher order particle states. The LFCC method solves this same mass eigenvalue problem using an exponential operator. This exponential operator can then be truncated instead to form a finite system of equations that can be solved using a built in system solver provided in most computational enviroments, such as MatLab and Mathematica. First approximation in the LFCC method allows for only one particle to be created by the new operator and proved to be not powerful enough to match the Fock state expansion. The second order approximation allowed one and two particles to be created by the new operator and converged to the Fock state expansion results. This showed the LFCC method to be a reliable replacement method for solving quantum field theory problems.

Contents

List of Figures

Chapter 1

Introduction

An important aspect to physics is understanding real systems on a fundamental level. We can look at the building blocks of the universe and try to understand how they work. Classically, we want to find the equations of motion for a system. Understanding the equations of motion define how the system behaves. The way we do this is first defining the Lagrangian.

This method breaks down as objects get smaller and smaller and we have to go to a quantum frame. Quantum mechanically we care about the energy of a system. Here we define a system's Hamiltonian, then solve an eigenvalue problem to get the energy.

This system also breaks down once particles are moving at relativistic speeds. For this, we move to quantum field theory and solve for a mass eigenvalue for the system. In this work, we will be looking at a charged particle surrounded by a cloud of neutrals. This is primarily done through quenched scalar Yukawa theory [1,2].

The Lagrangian for a system, classically, is the kinetic energy function of a system with the potential energy function subtracted. Kinetic energy is defined nonrelativistically as $\frac{p^2}{2m}$ where p is the momentum and m is the mass. Quantum mechanically

$$p = -i\hbar \frac{\partial}{\partial x} \tag{1.1}$$

in one dimension. We can define a Hamiltonian to be the kinetic energy added to the potential energy. Schrödinger's equation is then defined as

$$E\psi = H\psi. \tag{1.2}$$

This is an eigenvalue problem. From a time independent one dimensional view we have

$$E\psi = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V \right] \psi. \tag{1.3}$$

Here, ψ is a wave function.

A wave function defines the probability for system to be in a state. For example, for a time independent single particle system

$$\int_b^a |\psi(x)|^2 dx \tag{1.4}$$

defines the probability for the particle to be between a and b. Wave functions mathematically behave as vectors in a Hilbert space. To get information out of them, we act on them with linear transformations we call operators. Operators behave like matrices and therefore are not always commutative. If A and B are both operators

$$[A, B] \equiv AB - BA, \tag{1.5}$$

and is called a commutation relation.

The Hilbert space is the collection of all functions that define a vector space with

$$\int_b^a |\psi(x)|^2 dx < \infty. \tag{1.6}$$

The state of the system will be represented using bra-ket notation. The ket is symbolized by $|\rangle$. The state of a quantum mechanical system will be represented by $|\psi\rangle$, and the corresponding wave function for this system is defined by

$$|\psi\rangle = \int_\infty^\infty dx\, |x\rangle \langle x|\psi\rangle\,, \tag{1.7}$$

with $\langle x|\psi\rangle = \psi(x)$ and $|x\rangle$ the eigenvector of position [3].

The particles in this work are behaving under relativistic assumptions. This means we have to treat our particles as fields for a proper quantum description [4]. A field here means a quantum operator defined at each point in space and time. This makes spatial coordinates difficult to work with. We will then use momentum to define everything and work in a Fock space instead. A Fock space is a Hilberts space built from states with a definite number of particles, each with a definite momentum. A Fock state is a particular basis state in the Fock space.

Fock state expansion works primarily with momentum so, if the total momentum changes, our wave functions will change. To fix this, we will work with light-front coordinates, which will be discussed in chapter 2. This coordinate system is based on momentum fractions and is invariant under a Lorentz transformation. Chapter 2 will also show the Lagrangian and Hamiltonian for our system. The mass eigenvalue is found by solving a generalized eigenvalue problem by truncating the number of particles used.

The light-front coupled-cluster method (LFCC) analyses the same physical situations but without using a Fock-space truncation [5]. Instead of truncating the number of particles, the way the wave functions interact are defined through an operator, T, which is then truncated instead [6].

Previous work involving the LFCC method focused the lowest level approximation to the T operator. This included work on ϕ^4 theory, which is a theory of neutrals only with a Hamiltonian into four parts which created and annihilated different number of neutrals [7]. The next step is to consider higher order approximation to the T operator. That is what we consider here, to see how many terms might be needed to accurately represent a quantum eigenstate. This will be done in quenched scalar Yukawa theory (a.k.a. the Wick-Cutosky model), where quenched means that pair creation and annihilation is excluded. This theory provides perhaps the simplest context within which to consider the LFCC method and therefore provides an ideal test. The Fock-state epansion method will be used to judge the accuracy of the LFCC results. To see why the quenched form of the theory is necessary, see earlier work such as [8].

The light-front coupled-cluster method will be discussed in detail in Chapter 3. This chapter will present both first and second order approximations. Chapter 4 will then compare the Fock state expansion to the LFCC method with the code for the work provided in Appendix A. Chapter 5 contains a brief summary.

Chapter 2

Light-Front Scalar Field Theory

To meet the objective of this work, we wish to study scalar Yukawa theory. To do this, we need to build a foundation by examining the field theory using light-front coordinates

2.1 Light-Front Coordinates

Light-front coordinates [9] are defined by $x^{\pm} \equiv t \pm z$, with x^{+} as the light-front time and with the transverse coordinates unchanged $\vec{x}_{\perp} \equiv (x, y)$. The spacetime four-vectors are defined as $x^{\mu} = (x^{+}, x^{-}, \vec{x}_{\perp})$ and the three vector as $\underline{x} = (x^{-}, \vec{x}_{\perp})$. The four-momentum is then $p^{\mu} = (p^{+}, p^{-}, \vec{p}_{\perp})$ with $p^{-} \equiv E - p_z$ and $p^{+} \equiv E + p_z$, the light-front energy and longitudinal momentum respectively. The dot product of a coordinate vector and momentum vector is $p \cdot x = \frac{1}{2}(p^{-}x^{+} + p^{+}x^{-}) - \vec{p}_{\perp} \cdot \vec{x}_{\perp}$. The mass shell condition $p^2 = m^2$ becomes $p^{-} = (m^2 + p_{\perp}^2)/p^{+}$. Spatial derivatives are defined by

$$\partial_{\pm} \equiv \frac{\partial}{\partial x^{\pm}} = \frac{1}{2}\left(\frac{\partial}{\partial t} \pm \frac{\partial}{\partial z}\right), \vec{\partial}_{\perp} \equiv \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right).$$

Systems of n particles with momentum p_i for the ith particle and invariant mass M have a total momentum P. Components are defined as normal but $P^{-} = (M^2 + P_{\perp}^2)/P^{+}$ Now define $x_i \equiv p_i^{+}/P^{+}$ and $\vec{k}_{i\perp} \equiv \vec{p}_{i\perp} - x_i \vec{P}_{\perp}$. Any two frames of reference can be connected by a combination of longitudinal boosts along z and a combination between transverse boosts and rotations. x^i and $\vec{k}_{i\perp}$ are invariant under these boosts and so, wave functions defined under these terms, are also invariant. This is the reason light-front quantization is used.

With these definitions, the light-front Hamiltonian eigenvalue problem becomes the set of two equations

$$\mathcal{P}^- |\psi(\underline{P})\rangle = \frac{M^2 + P_\perp^2}{P^+} |\psi(\underline{P})\rangle , \underline{\mathcal{P}} |\psi(\underline{P})\rangle = \underline{P} |\psi(\underline{P})\rangle.$$

The second equation is solved by expanding the eigenstate in Fock states, eigenstates of particle number with definite momentum for each particle.

2.2 Quenched Scalar Yukawa Theory

We will be looking at a charged scalar surrounded by a cloud of neutrals moving along in the z direction. The Lagrangian for this system and Hamiltonian density are

$$\mathcal{L} = \partial_\mu \mathcal{X}^* \partial^\mu \mathcal{X} - m^2 |\mathcal{X}|^2 + \frac{1}{2}(\partial_\mu \phi)^2 - \frac{1}{2}\mu^2 \phi^2 - g\phi|\mathcal{X}|^2, \tag{2.1}$$

$$\mathcal{H} = |\vec{\partial}_\perp \mathcal{X}|^2 + m^2 |\mathcal{X}|^2 + \frac{1}{2}(\vec{\partial}_\perp \phi)^2 + \frac{1}{2}\mu^2 \phi^2 + g\phi|\mathcal{X}|^2. \tag{2.2}$$

This system is very similar to a simple neutral scalar field and can be solved using Fourier decomposition

$$\phi(x) = \int \frac{dp^+ d^2 p_\perp}{\sqrt{16\pi^3 p^+}} \left[a(\underline{p}) e^{-ip\cdot x} + a^\dagger(\underline{p}) e^{ip\cdot x} \right] \tag{2.3}$$

$$\mathcal{X}(x) = \int \frac{dp^+ d^2 p_\perp}{\sqrt{16\pi^3 p^+}} \left[c_+(\underline{p}) e^{-ip\cdot x} + c_-^\dagger(\underline{p}) e^{ip\cdot x} \right]. \tag{2.4}$$

The operators a and c move between states by creating new particles with $a^\dagger$ and $c^\dagger$ or annihilating particles with a and c. They have the commutation relations

$$[a(\underline{p}), a(\underline{p}')] = 0, [a^\dagger(\underline{p}), a^\dagger(\underline{p}')] = 0, [a(\underline{p}), a^\dagger(\underline{p}')] = \delta(\underline{p} - \underline{p}'). \tag{2.5}$$

With non-zero commutation relation

$$[c_\pm(\underline{p}), c_\pm^\dagger(\underline{p}')] = \delta(\underline{p} - \underline{p}'). \tag{2.6}$$

The a operator is for the neutral particles with c being for the charged particle. For example, with $|0\rangle$ as the Fock vacuum, $a^\dagger(P)|0\rangle$ is the state of one neutral particle with momentum P. This then will lead to the Hamiltonian being written as $\mathcal{P}^- = \mathcal{P}_0^- + \mathcal{P}_{int}^-$.

$$\mathcal{P}_0^- = \int d\underline{p}\frac{m^2+\vec{p}_\perp^2}{p^+}\left[c_+^\dagger(\underline{p})c_+(\underline{p}) + c_-^\dagger(\underline{p})c_-(\underline{p})\right] + \int d\underline{q}\frac{\mu^2+\vec{q}_\perp^2}{q^+}a^\dagger(\underline{q})a(\underline{q}) \tag{2.7}$$

$$\begin{aligned}\mathcal{P}_{int}^- &= g\int\frac{d\underline{p}d\underline{q}}{\sqrt{16\pi^3 p^+q^+(p^++q^+)}}\Big[\Big(c_+^\dagger(\underline{p}+\underline{q})c_+(\underline{p}) \\ &\quad +c_-^\dagger(\underline{p}+\underline{q})c_-(\underline{p})\Big)a(\underline{q}) \\ &\quad +a^\dagger(\underline{q})\left(c_+^\dagger(\underline{p})c_+(\underline{p}+\underline{q}) + c_-^\dagger(\underline{p})c_-(\underline{p}+\underline{q})\right)\Big] \\ &\quad +g\int\frac{d\underline{p}d\underline{q}}{\sqrt{16\pi^3 p_1^+p_2^+(p_1^++p_1^+)}}\left[c_+^\dagger(\underline{p}_1)c_-^\dagger(\underline{p}_2)a(\underline{p}_1+\underline{p}_2)\right. \\ &\quad \left.+a^\dagger(\underline{p}_1+\underline{p}_2)c_+(\underline{p}_1)c_-(\underline{p}_2)\right]\end{aligned} \tag{2.8}$$

For the quenched theory, the pair production and annihilation term is dropped. The Fock-state expansion for a particle of charge ± 1 is

$$|\psi_\pm(\underline{P})\rangle = \sum_{n=0}^{\infty}(P^+)^{n/2}\int\left(\prod_i^n dx_i d^2k_{i\perp}\right)\theta(1-\sum_i^n x_i)\psi_n^\pm(x_i;\vec{k}_{i\perp})|x_i,\vec{k}_{i\perp},\underline{P},n,\pm\rangle, \tag{2.9}$$

with the Fock states written as

$$\begin{aligned}|x_i,\vec{k}_{i\perp},\underline{P},n,\pm\rangle &= \frac{1}{\sqrt{n!}}c_\pm^\dagger((1-\sum_i^n x_i)P^+,(1-\sum_i^n x_i)\vec{P}_\perp - \sum_i^n \vec{k}_{i\perp}) \\ &\quad \times\prod_i^n a^\dagger(x_iP^+,\vec{k}_{i\perp}+x_i\vec{P}_\perp)|0\rangle.\end{aligned} \tag{2.10}$$

Now this is substituted into the eigenvalue problem $\mathcal{P}^-|\psi_\pm(\underline{P})\rangle = \frac{M_\pm^2+P_\perp^2}{P^+}|\psi(\underline{P})\rangle$ which leads to a coupled system for the wave functions ψ_n

$$M_\pm^2\psi_n^\pm(x_i,\vec{k}_{i\perp}) = \left(\frac{m^2+(\sum_i\vec{k}_{i\perp})^2}{1-\sum_i x_i} + \sum_i^n\frac{\mu^2+k_{i\perp}^2}{x_i}\right)\psi_n^\pm(x_i;\vec{k}_{i\perp}) \tag{2.11}$$

$$+\frac{g}{\sqrt{16\pi^3 n}}\sum_j^n \frac{\psi^{\pm}_{n-1}(x_1,\vec{k_1}_\perp;\ldots;x_{j-1},\vec{k}_{j-1\perp};x_{j+1},\vec{k}_{j+1\perp};\ldots;x_n,\vec{k_n}_\perp)}{\sqrt{x_j(1-\sum_{i\neq j}x_i)(1-\sum_i x_i)}}$$
$$+\frac{g\sqrt{n+1}}{\sqrt{16\pi^3}}\int dy d^2q_\perp \theta(1-\sum_i^n x_i - y)\frac{\psi^{\pm}_{n+1}(x_1,\vec{k_1}_\perp;\ldots;x_n,\vec{k_n}_\perp;y,\vec{q}_\perp)}{\sqrt{y(1-\sum_i x_i - y)(1-\sum_i x_i)}}.$$

This system is then solved for $M_\pm$ and $\psi^\pm_n$.

2.3 Approximations and Equations

Eq. (2.7) can be simplified by assuming the system is in only one space dimension. We will also drop any superscript + or subscript $\pm$ from now on. We will divide both sides by μ^2 to reduce variables and rename $\frac{m^2}{\mu^2}=\tilde{m}^2$ and $\frac{g}{\sqrt{4\pi\mu^2}}=\lambda$. These simplifications result in

$$\begin{aligned}\frac{M^2}{\mu^2}\psi_n(x_i) &= \left(\frac{\tilde{m}^2}{1-\sum_i x_i}+\sum_i^n\frac{1}{x_i}\right)\psi_n(x_i) \\ &+\frac{\lambda}{\sqrt{n}}\sum_j^n\frac{\psi_{n-1}(x_1,\ldots,x_{j-1},x_{j+1},\ldots,x_n)}{\sqrt{x_j(1-\sum_{i\neq j}x_i)(1-\sum_i x_i)}} \\ &+\lambda\sqrt{n+1}\int dy\theta(1-\sum_i^n x_i - y)\frac{\psi_{n+1}(x_1,\ldots,x_n,y)}{\sqrt{y(1-\sum_i x_i - y)(1-\sum_i x_i)}}.\end{aligned} \tag{2.12}$$

Now we want to write $\psi_n(x_i)$ in terms of symmetric polynomials or

$$\psi_n(x_1,\ldots,x_n)=\sum_{N,k=0}b_{Nk}\sqrt{x_1x_2\ldots x_n\left(1-\sum_i^n x_i\right)}Q^{(n)}_{Nk}(x_1,\ldots,x_n). \tag{2.13}$$

$Q^{(n)}_{Nk}$ needs to be symmetric with respect to $(x_i \leftrightarrow x_j)$. This makes

$$Q^{(n)}_{Nk}=x_1^{m_1}x_2^{m_2}\ldots x_n^{m_n}+\sum_{i\neq j}(x_i\leftrightarrow x_j), \tag{2.14}$$

$N = m_1+m_2+\ldots+m_n$ where $N\geq m_i\geq 0$. N is the total degree of the basis polynomials with k summing over all basis polynomials with total degree N. To help illustrate this

$$
\begin{aligned}
Q_{N0}^{(n)} &= x_1^N + x_2^N + \ldots + x_n^N \\
\\
Q_{N1}^{(n)} &= x_1^{N-1}x_2 + x_1^{N-1}x_3 + \ldots + x_1^{N-1}x_n + x_2^{N-1}x_1 + x_2^{N-1}x_3 + \ldots \\
&\vdots \\
Q_{Nk}^{(n)} &= x_1^{m_1}x_2^{m_2}\ldots x_n^{m_n} + x_1^{m_2}x_2^{m_1}\ldots x_n^{m_n} + \ldots
\end{aligned} \tag{2.15}
$$

and so on. This sort of basis expansion will also be used in Chapter 3.

We can represent these polynomials using vectors. The vectors length will be the number of particles with elements consisting of m_i. This leads to the representation

$$
(m_1, m_2, \ldots, m_n) = x_1^{m_1}x_2^{m_2}\ldots x_n^{m_n} + \sum_{i\neq j}(x_i \leftrightarrow x_j). \tag{2.16}
$$

These vectors are easy to make in code using a recurrence function. The polynomials are then easier to work with and manipulate using these vectors instead.

Eq. (2.8) can now be re-written as

$$
\frac{M^2}{\mu^2}\sqrt{x_1x_2\ldots x_n\left(1-\sum_i^n x_i\right)}Q_{Nk}^{(n)}(x_1,\ldots,x_n) = \tag{2.17}
$$

$$
\begin{aligned}
&\left(\frac{\tilde{m}^2}{1-\sum_i x_i} + \sum_i^n \frac{1}{x_i}\right)\sqrt{x_1x_2\ldots x_n\left(1-\sum_i^n x_i\right)}Q_{Nk}^{(n)}(x_1,\ldots,x_n) \\
&+\frac{\lambda}{\sqrt{n}}\sum_j^n \frac{\sqrt{x_1\ldots x_{j-1}x_{j+1}\ldots x_n}Q_{Nk}^{(n-1)}(x_1,\ldots,x_{j-1},x_{j+1},\ldots,x_n)}{\sqrt{x_j(1-\sum_i x_i)}} \\
&+\lambda\sqrt{n+1}\int dy\theta(1-\sum_i^n x_i - y)\frac{\sqrt{x_1x_2\ldots x_n}Q_{Nk}^{(n+1)}(x_1,\ldots,x_n,y)}{\sqrt{(1-\sum_i x_i)}}.
\end{aligned}
$$

We will project onto Eq. (2.13) each state and part of the approximation will be how many projections we do. The projection will consist of multiplying and integrating over the momentum by

$$\int_0^1 \int_0^{1-x_1} \dots \int_0^{\left(1-\sum_i^{n-1} x_i\right)} \prod_i^n dx_i \sqrt{x_1 x_2 \dots x_n \left(1 - \sum_i^n x_i\right)} Q^{(n)}_{N'k'}(x_1, \dots, x_n) \quad (2.18)$$

This projection defines a generalized eigenvalue problem of the form

$$C\vec{\Psi} = \frac{M^2}{\mu^2} D\vec{\Psi}. \quad (2.19)$$

$\vec{\Psi}$ is a vector consisting of the b_{Nk}. C and D are block matrices of the form

$$C = \begin{pmatrix} c_{00} & c_{01} & 0 & 0 & \dots \\ c_{10} & c_{11} & c_{12} & 0 & \dots \\ 0 & c_{21} & c_{22} & c_{23} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \quad (2.20)$$

$$D = \begin{pmatrix} d_0 & 0 & 0 & \dots \\ 0 & d_1 & 0 & \dots \\ 0 & 0 & d_2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where d_i is a block with entries

$$\begin{aligned} d_{iN_1k_1,N_2,k_2} &= \int_0^1 \int_0^{1-x_1} \dots \int_0^{\left(1-\sum_j^{i-1} x_j\right)} \prod_j^i dx_j \, x_1 x_2 \dots x_i \left(1 - \sum_j^i x_j\right) \quad (2.21) \\ &\times Q^{(i)}_{N_1k_1}(x_1, \dots, x_i) Q^{(i)}_{N_2k_2}(x_1, \dots, x_i). \end{aligned}$$

C represents Eq. (2.13) where the diagonal matrices are contributed by the same sector term or the first term on the right hand side to give

$$c_{iiN_1k_1,N_2,k_2} = \int_0^1 \int_0^{1-x_1} \dots \int_0^{\left(1-\sum_j^{i-1} x_i\right)} \prod_j^i dx_j \left(\frac{\tilde{m}^2}{1 - \sum_j x_j} + \sum_j^i \frac{1}{x_j} \right) \quad (2.22)$$

$$\times x_1 x_2 \ldots x_i \left(1 - \sum_j^i x_j\right) Q^{(i)}_{N_1 k_1}(x_1, \ldots, x_i) Q^{(i)}_{N_2 k_2}(x_1, \ldots, x_i).$$

The upper diagonal matrices are contributions made by a neutral particle being annihilated. This means the ith+1 state makes a contribution to the ith projection given by the third term in Eq. (2.13) yielding

$$\begin{aligned} c_{ii+1 N_1 k_1, N_2, k_2} &= \lambda\sqrt{i+1} \int_0^1 \int_0^{1-x_1} \ldots \int_0^{\left(1-\sum_i^{n-1} x_i\right)} \int_0^{\left(1-\sum_j^i x_i\right)} \prod_j^i dx_j \, dy \, x_1 x_2 \ldots x_i \\ &\quad \times Q^{(i)}_{N_1 k_1}(x_1, \ldots, x_i) Q^{(i+1)}_{N_2 k_2}(x_1, \ldots, x_{i+1}). \end{aligned} \tag{2.23}$$

The lower diagonal terms correspond to a neutral particle being created. This means the ith-1 state makes a contribution to the ith projection given by the second term in Eq. (2.13) yielding

$$\begin{aligned} c_{i-1 i N_1 k_1, N_2, k_2} &= \frac{\lambda}{\sqrt{i}} \int_0^1 \int_0^{1-x_1} \ldots \int_0^{\left(1-\sum_i^{n-1} x_i\right)} \prod_j^i dx_j \\ &\quad \times \sum_m^i x_1 x_2 \ldots x_{m-1} x_{m+1} \ldots x_i Q^{(i-1)}_{N_1 k_1}(x_1, \ldots, x_{i-1}) Q^{(i)}_{N_2 k_2}(x_1, \ldots, x_i). \end{aligned} \tag{2.24}$$

With the basis polynomials and physical symmetry of the system, both C and D will be symmetric. This is a useful check when coding and looking at results.

2.4 Results for Fock State Expansion

We have defined multiple variables we are free to play with. The simplest is $\tilde{m}$. This is a ratio between the mass of the charged particle and the mass of a single neutral particle. We will be looking at four different values for $\tilde{m}$ namely 0.01,0.1,1, and 10. For each $\tilde{m}$ value we can control how many particle sectors we use to find our mass ratio, $\frac{M^2}{\mu^2}$, or total system mass over a neutral particle mass. We will use n to mean the n-neutral sector. Finally, we want to see how the mass ratio changes with the coupling strength, λ. The higher the coupling strength, the more sensitive the result should be to the number of constituents. We are looking for convergence within each sector with respect to the basis size as well as overall convergence with respect to the number of sectors. Since the result is a mass ratio, any result that gives a negative value is not a physical solution a negative mass squared corresponds to an imaginary mass. For a single neutral particle, Figure 2.1 shows immediate convergence between basis polynomial orders.

Figure 2.2 shows a similar result with some noticeable differences in mass ratio for coupling strengths higher than 0.8 but these difference appear after the mass ratio is negative and can be disregarded. Figure 2.3 shows differences can occur before the mass ratio becomes negative. Figure 2.3 shows why convergence with respect to basis polynomials is tested. We can see how the sectors are influencing each other to get a better idea of what is going on. We used relative probability defined by

$$\int \frac{\psi_i^2}{\psi_0^2} \prod_i dx_i. \tag{2.25}$$

The probabilities were then plotted with respect to the coupling strength shown in Figure 2.4. This shows that the zero-neutral sector is the most probable for $\tilde{m} = 1$. Figure 2.6 shows that as we increase the number of particles, the probability for higher sectors become more noticeable with Figure 2.8 implying convergence since the five-sector contribution is close to zero relative to the other sectors.

We see something similar in Figure 2.10 where we are comparing sectors. Notice how the four, five, and six-neutral sector are converged on top of each other. This is expected from the probability graph, Figure 2.8, giving such low probability for the five-neutral sector. We do the same analysis for $\tilde{m} = 10$. The mass of the charged particle being 10 times larger than the neutral particles runs into some complications. The final mass ratio is fairly large so to get to a point where it becomes negative, larger coupling strength needs to be tested. The larger $\tilde{m}$ value results in a higher dependence on higher particle sectors illustrated by Figure 2.14. The overall convergence, shown in Figure 2.15, is not very good for large coupling. This result is believed to be a limitation on the code itself. Since, more neutral sectors had to be used, this causes a larger generalized eigenvalue problem to have to be solved. This can cause compound errors to occur especially if any eigenvalues are small. One attempt at fixing this solution was to normalize the basis polynomials but this did not fully solve the problem. Notice from Figure 2.15 as the number of particles increased, the order of basis polynomial could only be first order.

This was more from the code giving errors than actual convergence.

For small values of $\tilde{m}$ the opposite occurs. Convergence is seen almost immediately shown in Figures 2.16-17 with evidence for convergence in Figure 2.18 which is the probability graph showing that only the one-neutral sector matters. We also have a limit with how little $\tilde{m}$ can be, which is illustrated in Figure 2.20. As the coupling strength goes to zero

$$\frac{M^2}{\mu^2} = \tilde{m}^2 \tag{2.26}$$

but with $\tilde{m} = 0.01$ the mass ratio appears to almost always be negative.

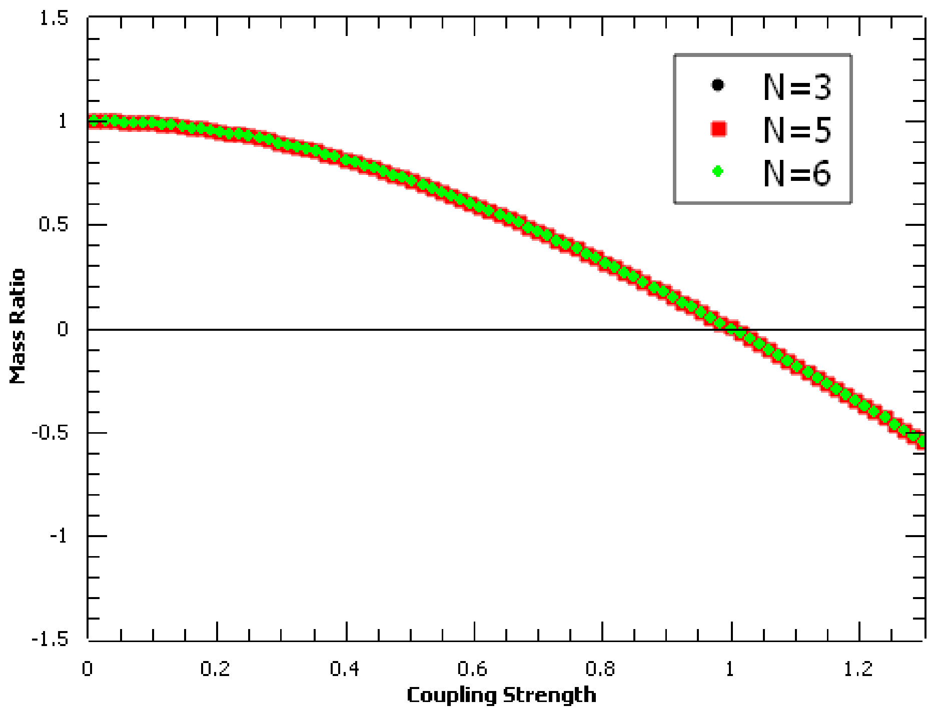

Figure 2.1: One-neutral sector convergence for $\tilde{m} = 1$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

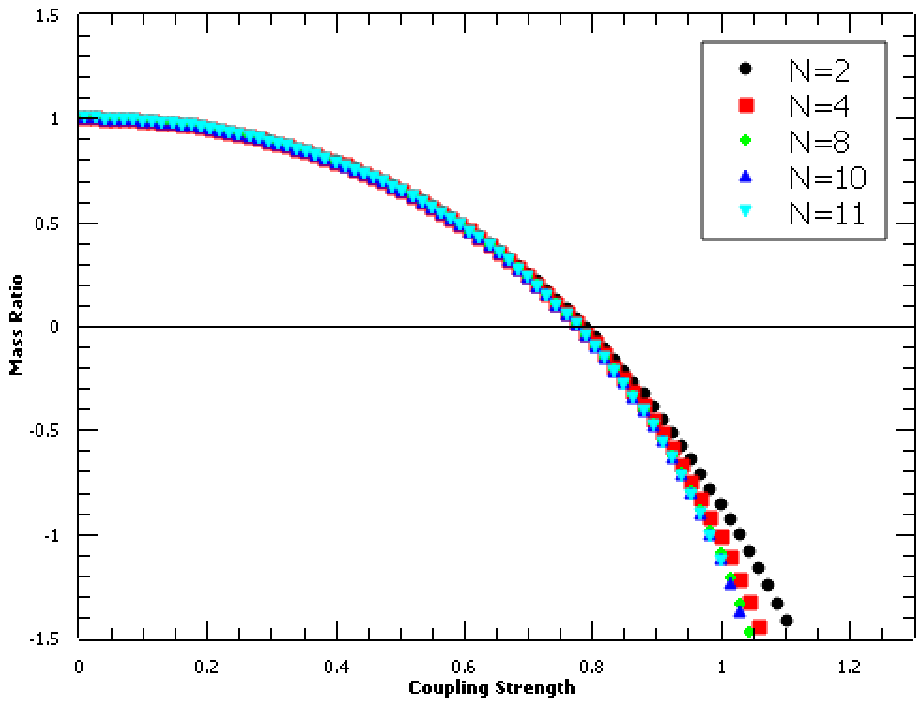

Figure 2.2: Two-neutral sector convergence for $\tilde{m} = 1$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

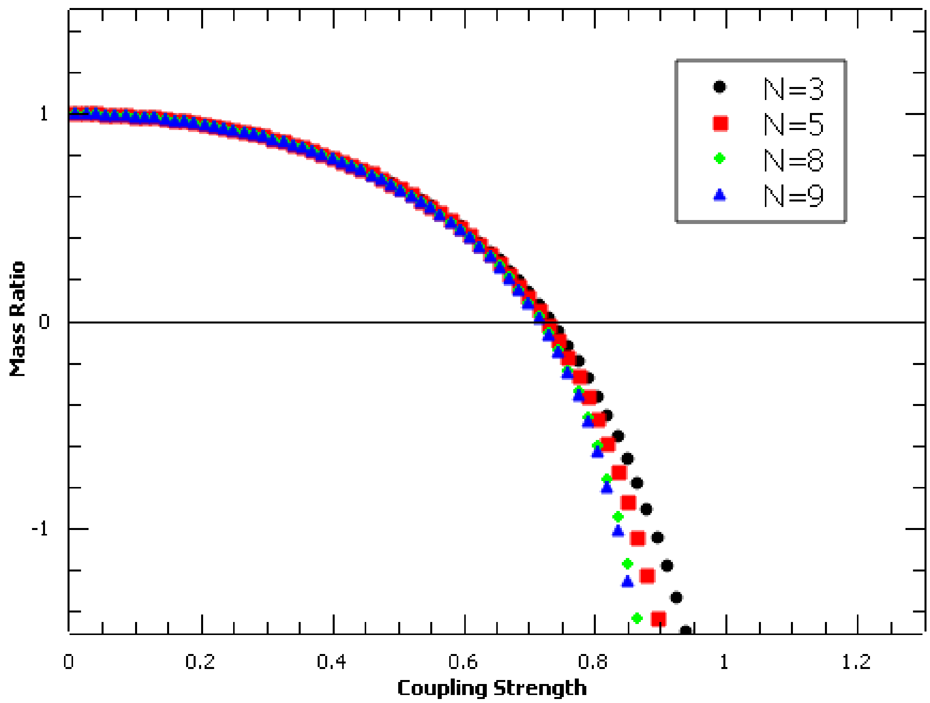

Figure 2.3: Three-neutral sector convergence for $\tilde{m} = 1$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

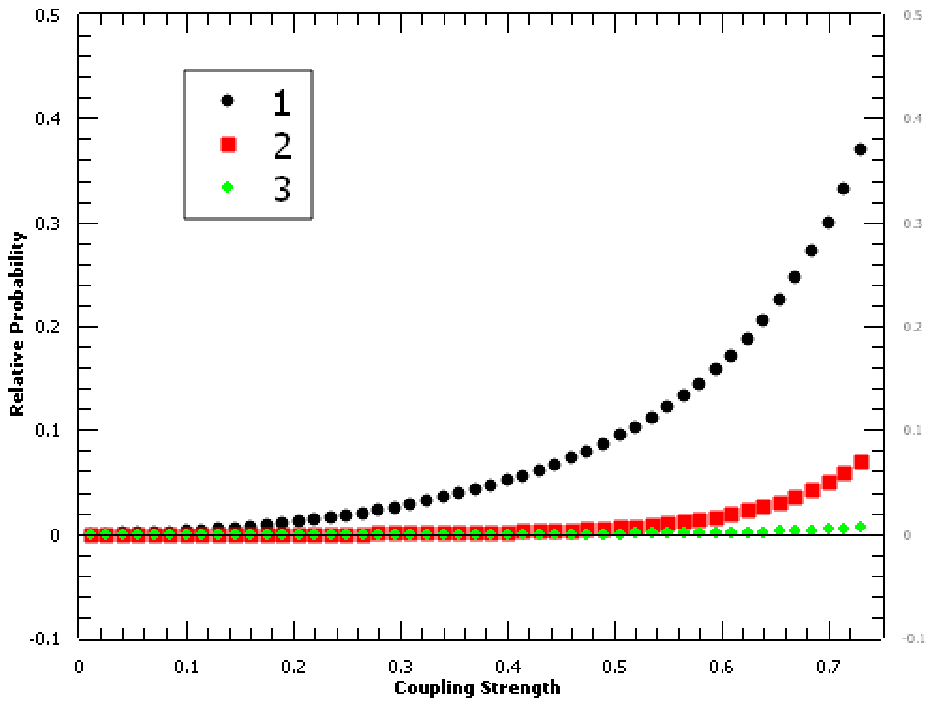

Figure 2.4: Relative probabilities with for up to three neutrals for $\tilde{m} = 1$, with each curve corresponding to the relative probability between ψ_0 and ψ_n. The graph shows how the probabilities between each sector change with the coupling strength. This graph shows that the most probable state is the zero-neutral state.

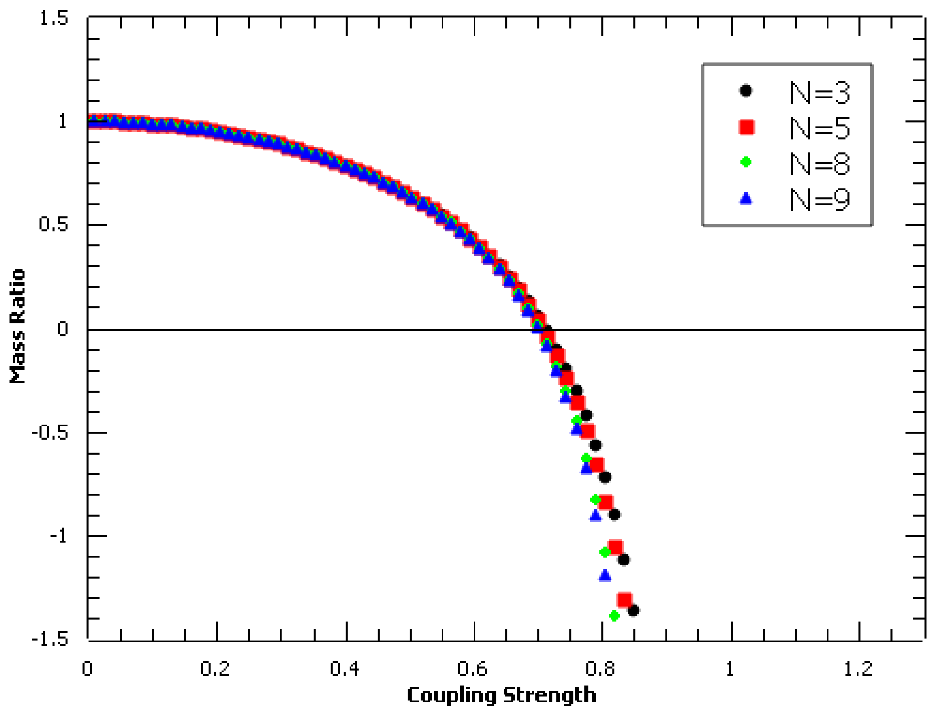

Figure 2.5: Four-neutral sector convergence for $\tilde{m} = 1$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

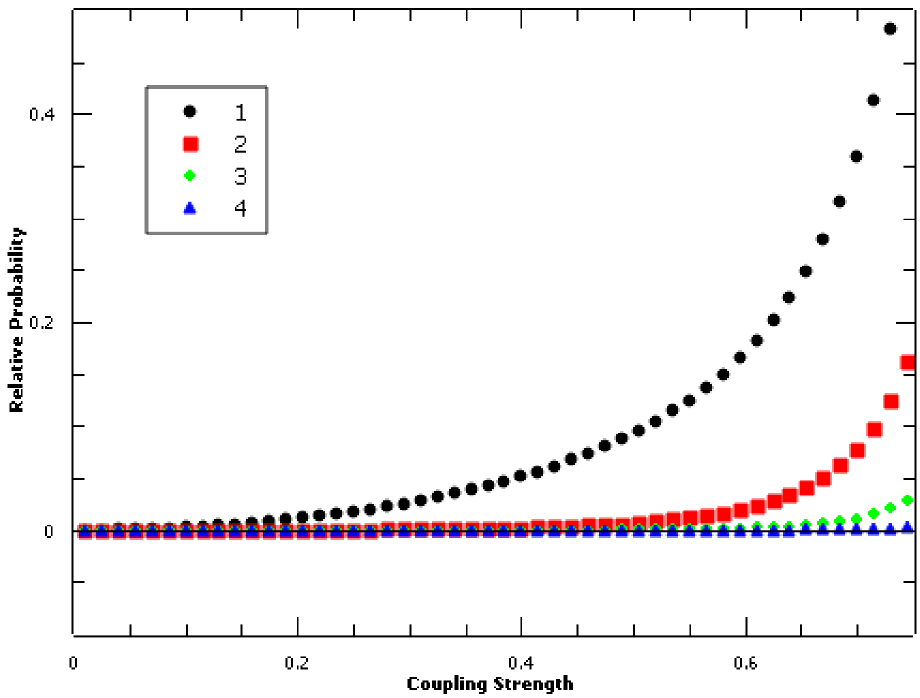

Figure 2.6: Relative probabilities for up to four neutrals for $\tilde{m} = 1$, with each curve corresponding to the relative probability between ψ_0 and ψ_n. The graph shows how the probabilities between each sector change with the coupling strength. This graph shows that the four-neutral sector has very little probability.

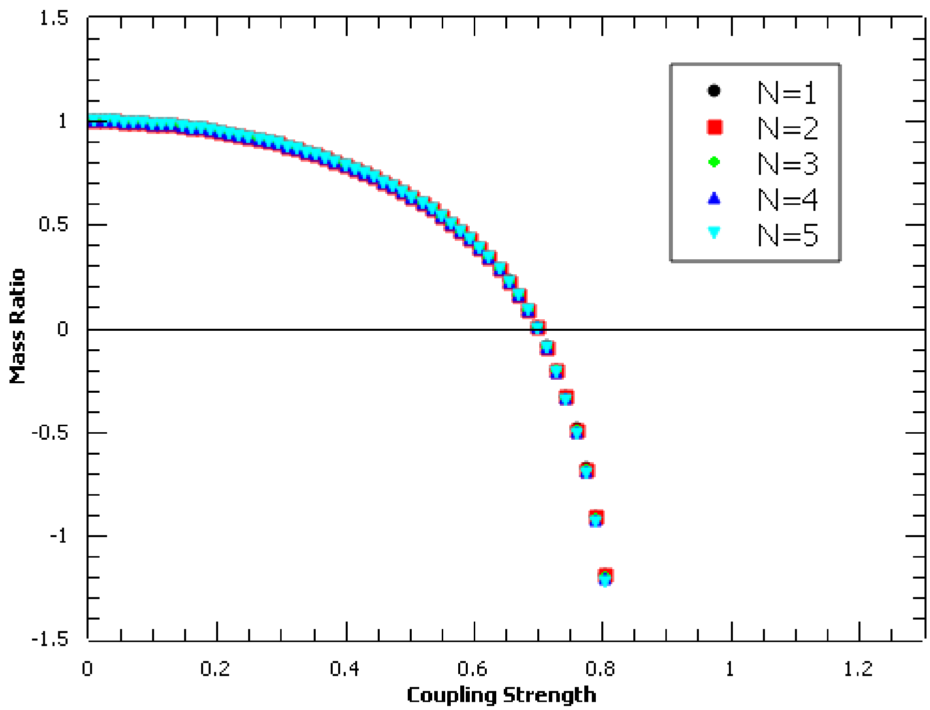

Figure 2.7: Five-neutral sector convergence for $\tilde{m} = 1$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

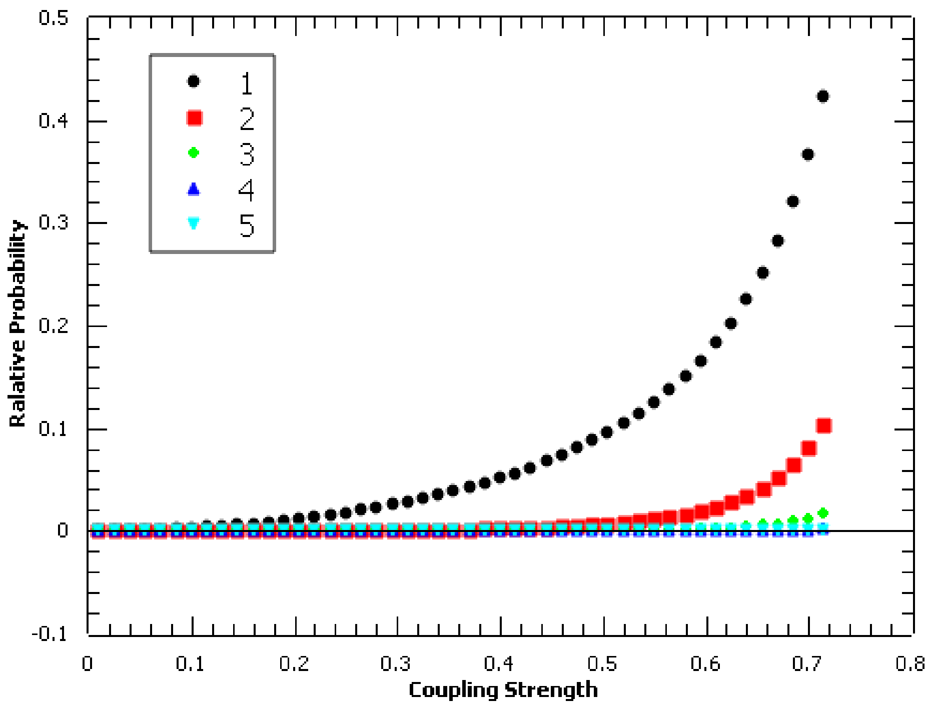

Figure 2.8: Relative probabilities for up to five neutrals for $\tilde{m} = 1$, with each curve corresponding to the relative probability between ψ_0 and ψ_n. The graph shows how the probabilities between each sector change with the coupling strength. This graph shows that as more particles are added, the probability gets less and less.

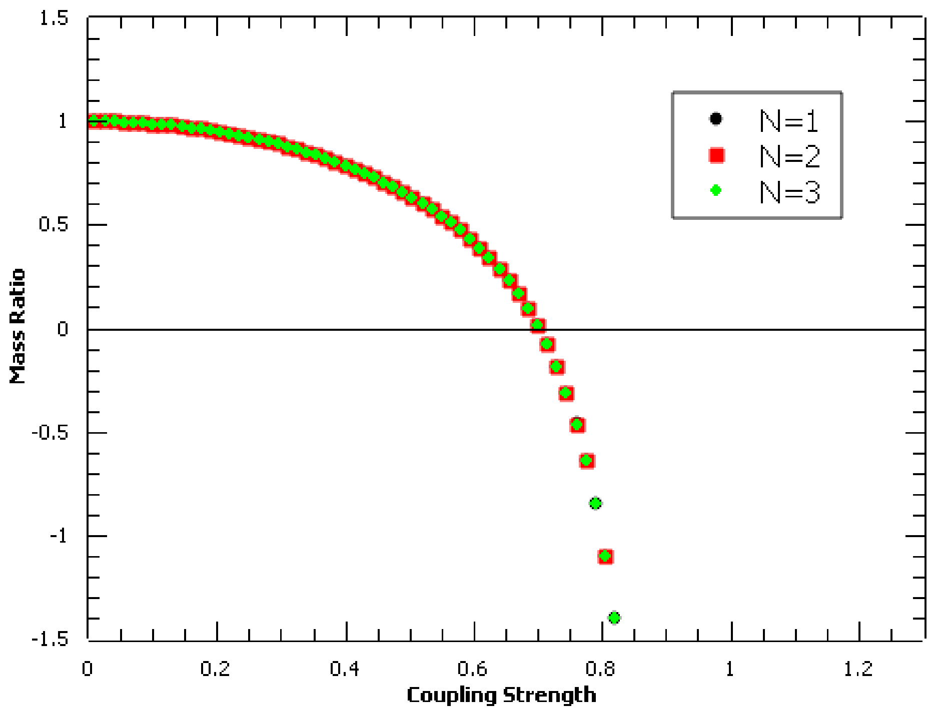

Figure 2.9: Six-neutral sector convergence for $\tilde{m} = 1$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

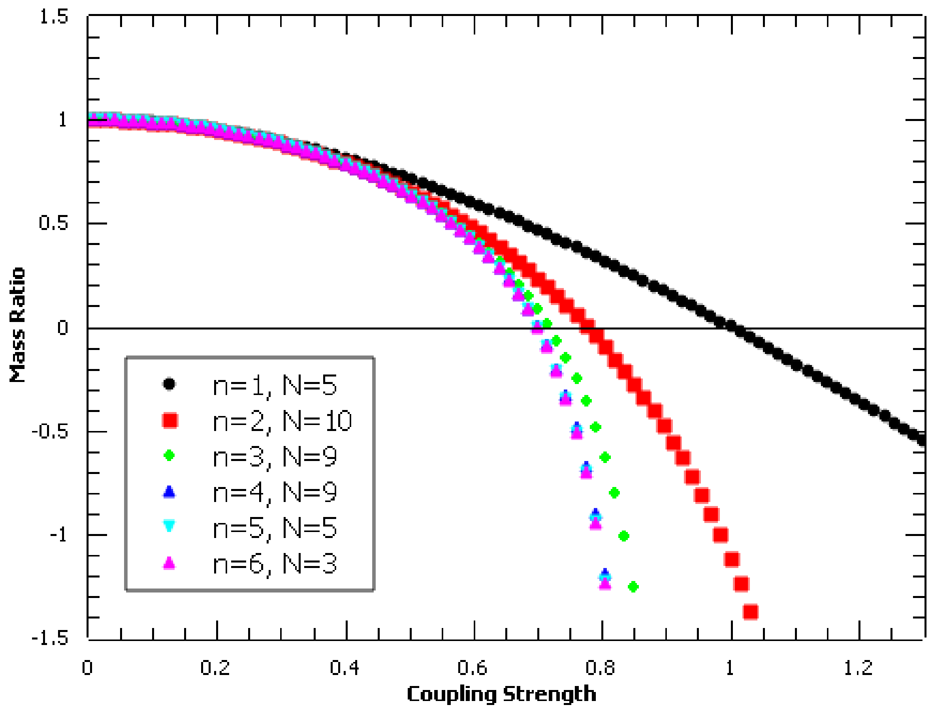

Figure 2.10: Sector convergence for $\tilde{m} = 1$ with n being the number of neutrals in the highest Fock sector and N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

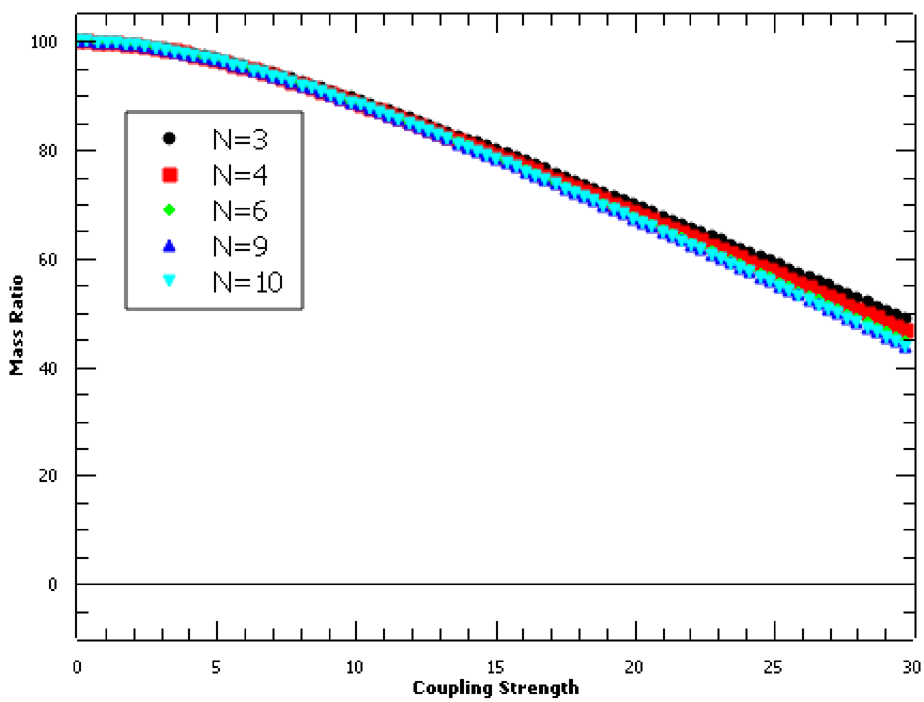

Figure 2.11: One-neutral sector convergence for $\tilde{m} = 10$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

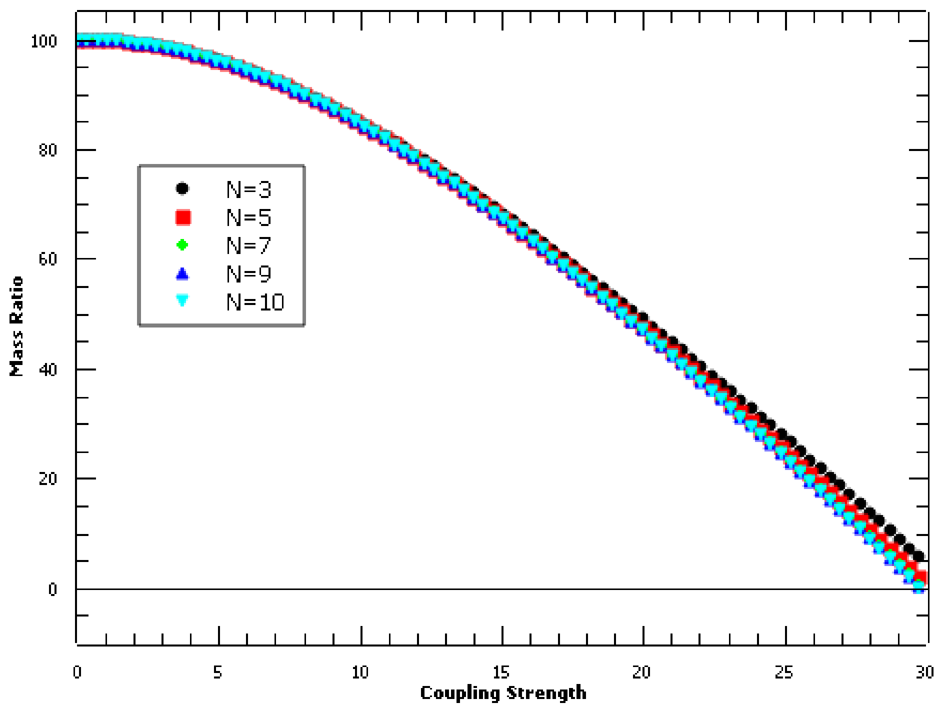

Figure 2.12: Two-neutral sector convergence for $\tilde{m} = 10$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

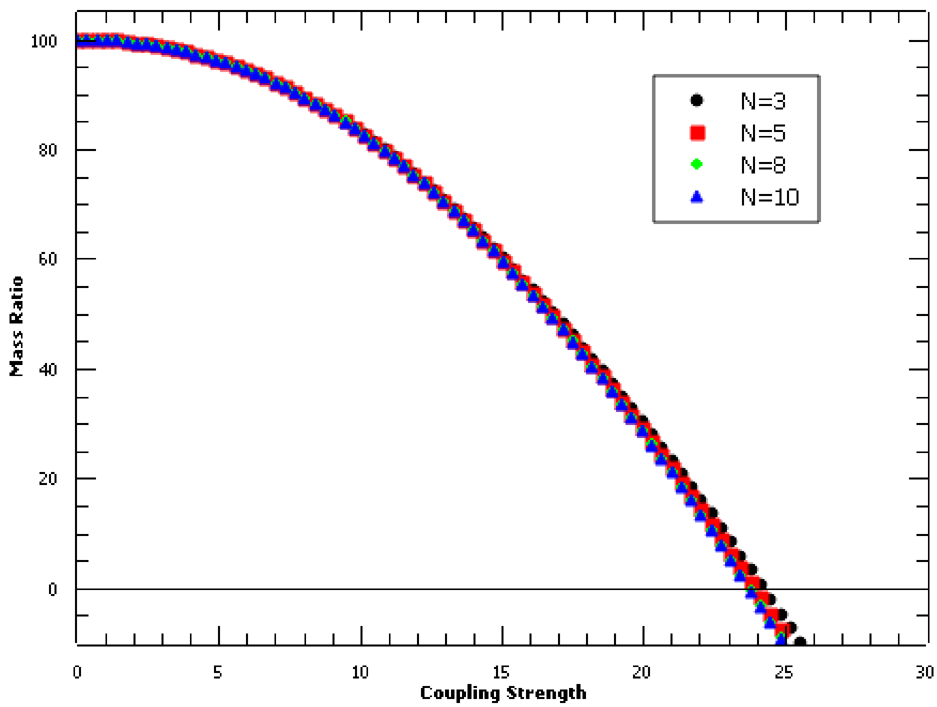

Figure 2.13: Three-neutral sector convergence for $\tilde{m} = 10$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

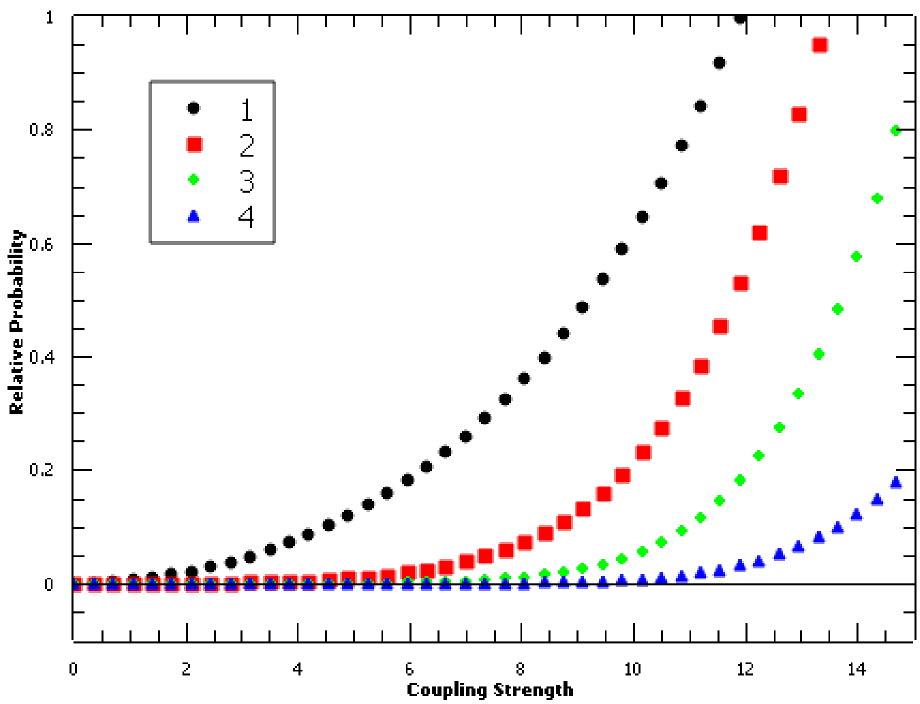

Figure 2.14: Relative probabilities for up to four neutrals for $\tilde{m} = 10$, with each curve corresponding to the relative probability between ψ_0 and ψ_n. The graph shows how the probabilities between each sector change with the coupling strength. This graph shows that the most probable state is the zero-neutral sector. For coupling strengths below 12, but that sectors with neutrals become more probable above 12.

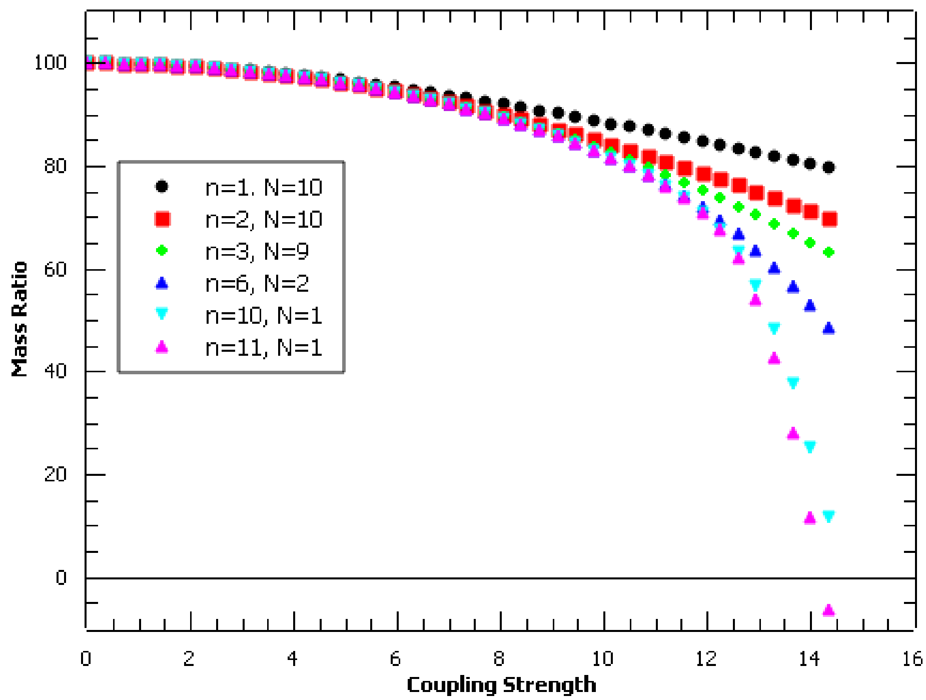

Figure 2.15: Sector convergence for $\tilde{m} = 10$ with n being the number of neutrals in the highest Fock sector and N being the highest order for basis polynomials. The graph is cut off at coupling strength 16 due to compound error in the code causing results to miss the desired eigenvalue. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

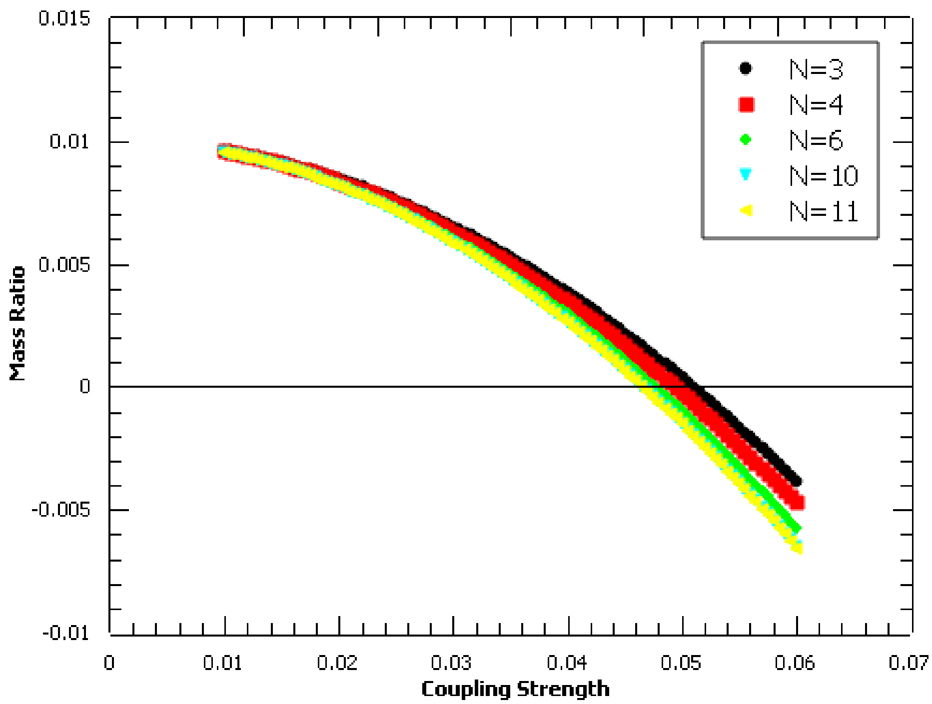

Figure 2.16: One-neutral sector convergence for $\tilde{m} = 0.1$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

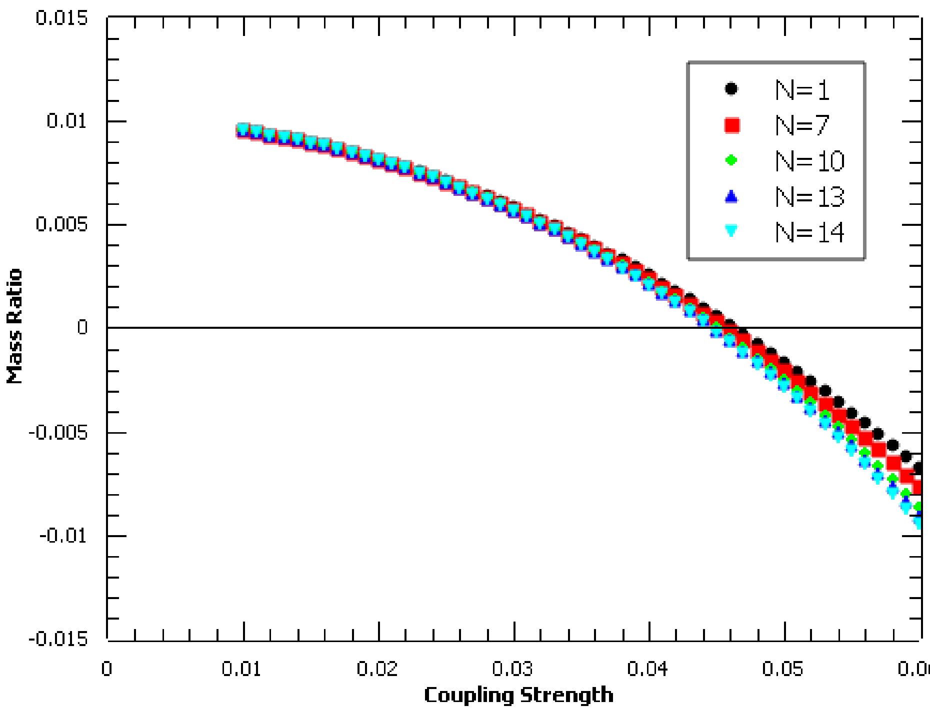

Figure 2.17: Two-neutral sector convergence for $\tilde{m} = 0.1$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence.

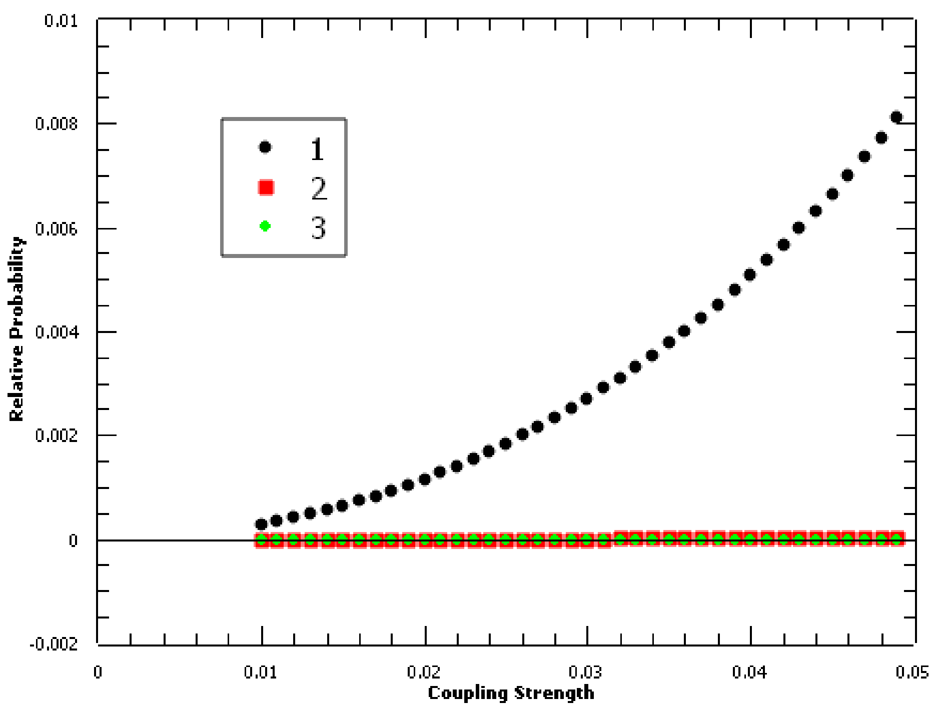

Figure 2.18: Relative probabilities for up to four neutrals for $\tilde{m} = 0.1$, with each curve corresponding to the relative probability between ψ_0 and ψ_n. The graph shows how the probabilities between each sector change with the coupling strength. This graph shows that the most probable state is by far the zero-neutral state.

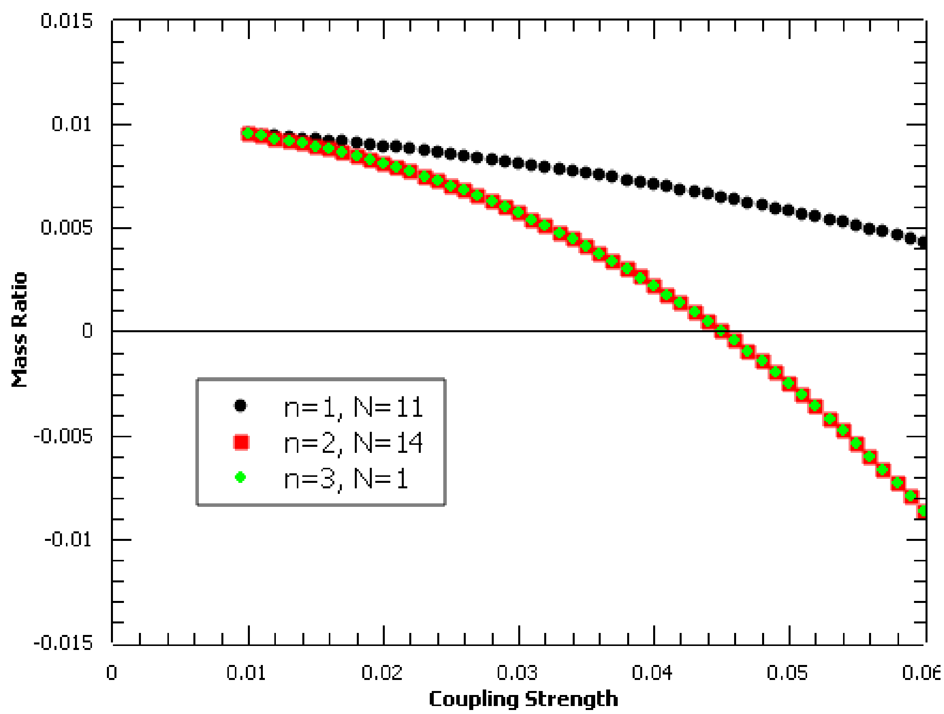

Figure 2.19: Sector convergence for $\tilde{m} = 0.1$ with n being the number of neutrals in the highest Fock sector and N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and can be disregarded. The graph shows a decreasing difference as the number of Fock sectors is increased, which implies convergence. For this small $\tilde{m}$ value, convergence is seen almost immediately.

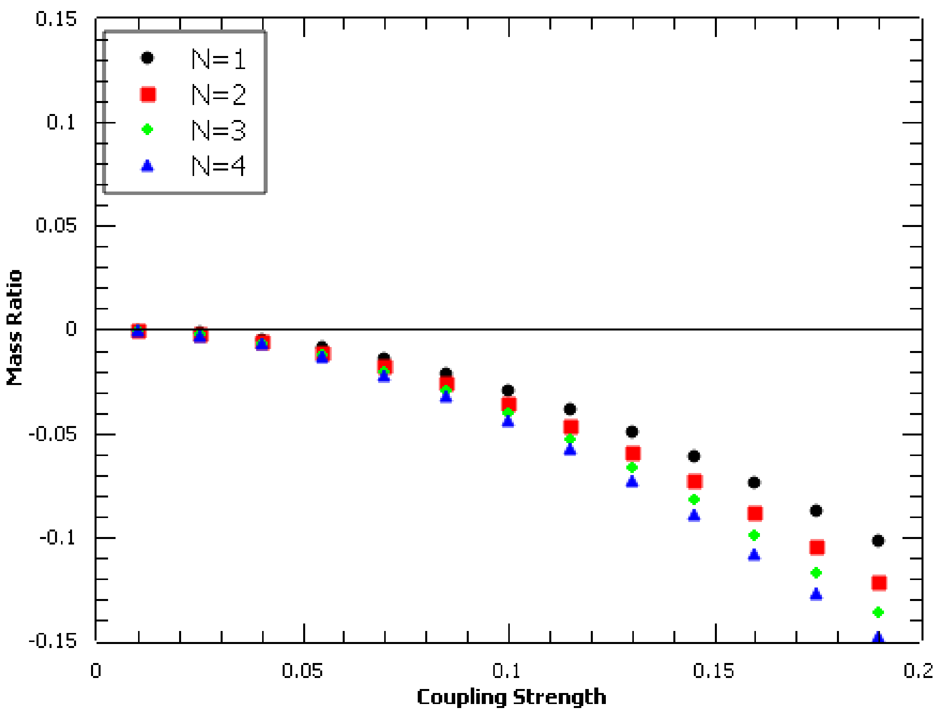

Figure 2.20: One-neutral sector convergence for $\tilde{m} = 0.01$ with N being the highest order for basis polynomials. Any value below 0 corresponds to an imaginary mass which is not physical and be disregarded. For low enough $\tilde{m}$, no coupling strength leads to a physical mass. The most probable cause is compound error in the code due to such small values.

Chapter 3

Light-Front Coupled-Cluster Method

3.1 Introduction

Solving the eigenvalue problem in Chap. 2 normally involves expanding the state in Fock states. This will lead to an infinite set of equations that need to be truncated. This will lead to multiple errors. We can avoid Fock-space truncation by expanding the state in a single Fock state and defining an exponential operator

$$\left|\psi_\pm(P^+)\right\rangle = \sqrt{Z}e^T \left|\phi\right\rangle , \tag{3.1}$$

with $|\phi\rangle = c^\dagger_\pm(P^+)\,|0\rangle$. T is defined as an infinite sum

$$T = \sum_n T_n, \tag{3.2}$$

where T_n when acting on the state creates n particles and is defined as

$$T_n = \int dp^+ \prod_i^n dq_i^+ t_n(q_1^+, \ldots, q_n^+, p^+) \left(\prod_i^n a^\dagger(q_i^+)\right) c^\dagger_\pm(p^+)c_\pm \left(p^+ + \sum_i^n q_i^+\right). \tag{3.3}$$

The original eigenvalue problem then becomes

$$P_v \overline{\mathcal{P}}^- \left|\phi\right\rangle = \frac{M_\pm^2 + P_\perp^2}{P^+} \left|\phi\right\rangle, (1 - P_v)\overline{\mathcal{P}}^- \left|\phi\right\rangle = 0. \tag{3.4}$$

With P_v being the projection onto the valence sector and $\overline{\mathcal{P}}^- = e^{-T}\mathcal{P}^- e^T$. In the first approximation, we only allow one particle to be created at a time. This results in $T = T_1$. $P_\perp^2$ will be set to zero again assuming the system is in one space dimension.

3.2 First Approximation

Now we use a Taylor series expansion to rewrite the effective Hamiltonian as

$$\overline{\mathcal{P}}^- = \left(1 - T_1 + \frac{T_1^2}{2}\right)\mathcal{P}^- \left(1 + T_1 + \frac{T_1^2}{2}\right). \tag{3.5}$$

The truncation in the Taylor series is caused by only allowing a contribution up to the first sector, ψ_1. The T operator creates a neutral particle while the Hamiltonian, $\mathcal{P}^-$, can only annihilate one particle at most. This means any combination of T_1 acting on the state that creates more than two particles will not be allowed. Consequently, powers of T_1 higher than two can be ignored. Some terms remaining in $\mathcal{P}^-$ will also be dropped for the same reason resulting in

$$\overline{\mathcal{P}}^- = \mathcal{P}^- + \mathcal{P}^- T_1 - T_1\mathcal{P}^- + \frac{\mathcal{P}^- T_1^2}{2} - T_1\mathcal{P}^- T_1. \tag{3.6}$$

For the first approximation, we then project onto the valence sector and the one-particle sector to get a two-equation system of equations. In other words, the first equation will consist of contributions where only the charged particle survives whereas the second equation consists only of contributions where a single neutral and charged particle survive:

$$\frac{M^2}{P} = \frac{m^2}{P} + \int_0^P \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q, P-q)}{\sqrt{q(P-q)P}}, \tag{3.7}$$

$$0 = \frac{g}{\sqrt{4\pi}} \frac{1}{\sqrt{q_1'(P-q_1')P}} + \frac{\mu^2}{q_1'} t_1(q_1', P - q_1') \tag{3.8}$$

$$
\begin{aligned}
&+\frac{1}{2}\int_0^{P-q_1'} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q_1', P-q_1')t_1(q, P-q_1'-q)}{\sqrt{q(P-q_1')(P-q_1'-q)}} \\
&+\frac{m^2}{P-q_1'} t_1(q_1', P-q_1') \\
&+\frac{1}{2}\int_0^{P-q_1'} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q, P-q)t_1(q_1', P-q_1'-q)}{\sqrt{q(P-q_1')(P-q_1'-q)}} \\
&-\frac{m^2}{P} t_1(q_1', P-q_1') \\
&-\int_0^{P} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q_1', P-q_1')t_1(q, P-q)}{\sqrt{q(P-q)P}}.
\end{aligned}
$$

These equations are made through vertex rules similar to Feynman diagrams. Figure 3.1 consists of all possible outcomes of $\mathcal{P}^-$ acting on the current state. The charged particle is coming in from the right with total momentum P. The dotted lines are the neutral particles with momentum q, and the solid line is the charged particle that gains and loses momentum along the path. Figure 3.1(a) is represented by the first term in Eq. (3.8) on the right side. This vertex contributes the coefficient $\frac{g}{\sqrt{4\pi}}$ as well as an additional factor in the denominator consisting of the square root of the incoming and two outgoing momentum multiplied together. Figure 3.1(b) contributes the same coefficient out front but now the additional consists of the square root of the two incoming along with the single outgoing momentum multiplied together. This vertex also requires an integral from 0 to the total momentum with respect to the incoming momentum. This sort of integral is seen in the second term in Eq. (3.7) on the right side. Figure 3.1(c) and 3.1(d) give a fraction consisting of the mass of the particle squared that the vertex is acting on in the numerator and a denominator consisting of the particles momentum. The first term on the right side of Eq. (3.7) represents Figure 3.1(c). Figure 3.2 is is the diagram representation of T_1 acting on a single charged particle with momentum P. This contributes the function t_1, represented by the grey circle, where the arguments will consist of the two outgoing momentum. Figure 3.2 will contribute $t_1(q_1', P-q_1')$ specifically.

Each term in Equations 3.7 and 3.8 correspond to a particle diagram. The two terms on the right hand side of Equation 3.7 corresponds to Figure 3.1c and 3.3 respectively. The first term in Equation 3.8 corresponds to Figure 3.1a with the term linked to Figure

3.4 added next. The next line corresponds to Figure 3.5. The next mass term originates from Figure 3.6 with the following line being from Figure 3.7. The next term is the first term with a negative coefficient and is related to Figure 3.8. The last term in Equation 3.8 is related to Figure 3.9.

We still want these equations to be made up of momentum fractions. Recall from earlier $q_1' = x_1 P$ and $q = yP$. These substitutions are made and the extra factors of P from dq and under the square roots are combined, to obtain

$$
\begin{aligned}
\frac{M^2}{P} &= \frac{m^2}{P} + \int_0^1 \frac{g}{\sqrt{4\pi}} \frac{dy\ t_1(yP,(1-y)P)}{\sqrt{y(1-y)P}}, \qquad (3.9) \\
0 &= \frac{g}{\sqrt{4\pi}} \frac{1}{\sqrt{x_1(1-x_1)P^3}} + \frac{\mu^2}{x_1 P} t_1(x_1P,(1-x_1)P) \qquad (3.10) \\
&+ \frac{1}{2}\int_0^{1-x_1} \frac{g}{\sqrt{4\pi}} \frac{dy\ t_1(x_1P,(1-x_1)P)t_1(yP,(1-x_1-y)P)}{\sqrt{y(1-x_1)(1-x_1-y)P}} \\
&+ \frac{m^2}{(1-x_1)P} t_1(x_1P,(1-x_1)P) \\
&+ \frac{1}{2}\int_0^{1-x_1} \frac{g}{\sqrt{4\pi}} \frac{dy\ t_1(yP,(1-y)P)t_1(x_1P,(1-x_1-y)P)}{\sqrt{y(1-x_1)(1-x_1-y)P}} \\
&- \frac{m^2}{P} t_1(x_1P,(1-x_1)P) \\
&- \int_0^1 \frac{g}{\sqrt{4\pi}} \frac{dy\ t_1(x_1P,(1-x_1)P)t_1(yP,(1-y)P)}{\sqrt{y(1-y)P}}.
\end{aligned}
$$

We want $T_1 |\phi\rangle$ to match the second term from $|\psi_\pm(\underline{P})\rangle$ where for n=1

$$|\psi_\pm(P)\rangle = (P)^{1/2} \int dx_1 \psi_1^\pm(x_1) c_\pm^\dagger((1-x_1)P) a^\dagger(x_1P) |0\rangle . \qquad (3.11)$$

We need to then re-scale t_1 such that

$$t_1(x_1P,(1-x_1)P) = P^\alpha \tilde{t}_1(x_1). \qquad (3.12)$$

To determine α, rewrite T_1 with momentum fractions and this re-scaling for t_1 gives

$$T_1 |\phi\rangle = P^{\alpha+1} \int dx_1 \tilde{t}_1(x_1) a^\dagger(x_1P) c_\pm^\dagger((1-x_1)P) c_\pm(P) |\phi\rangle . \qquad (3.13)$$

The only difference we care about is the different powers on P. Defining α to be $-\frac{1}{2}$ fixes this issue. The re-scaling depends on the two argument of t_1. The method is taking the first argument, adding it to the second argument then dividing the first argument by this total. This is the argument for $\tilde{t}_1$ multiplied by the total raised to α,

$$t_1(yP,(1-x_1-y)P) = ((1-x_1)P)^{-\frac{1}{2}}\tilde{t}_1\left(\frac{y}{(1-x_1)}\right). \tag{3.14}$$

We will then use this substitution for our system of equations and simplify as much as we can which includes canceling P everywhere yielding

$$\begin{aligned}
M^2 &= m^2 + \int_0^1 \frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(y)}{\sqrt{y(1-y)}}, && (3.15)\\
0 &= \frac{g}{\sqrt{4\pi}}\frac{1}{\sqrt{x_1(1-x_1)}} + \frac{\mu^2}{x_1}\tilde{t}_1(x_1) && (3.16)\\
&+\frac{1}{2}\int_0^{1-x_1}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(x_1)\tilde{t}_1(\frac{y}{1-x_1})}{(1-x_1)\sqrt{y(1-x_1-y)}}\\
&+\frac{m^2}{1-x_1}\tilde{t}_1(x_1)\\
&+\frac{1}{2}\int_0^{1-x_1}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(y)\tilde{t}_1(\frac{x_1}{1-y})}{\sqrt{y(1-x_1)(1-y)(1-x_1-y)}}\\
&-m^2\tilde{t}_1(x_1)\\
&-\int_0^1\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(x_1)\tilde{t}_1(y)}{\sqrt{y(1-y)}}.
\end{aligned}$$

Now we want to apply numerical methods similar to those in section 2.4. For $\tilde{t}_1(x)$ we will be using a weighted Jacobi polynomial substitution to require orthogonality. We will use the same notation as

$$\tilde{t}_1(x) = \sum_n a_n\sqrt{x(1-x)}P_n^{(1)}(x), \tag{3.17}$$

where $P_n^{(1)}(x)$ is the nth Jacobi polynomial [10]. The superscript 1 indicates the first order approximation. Jacobi polynomials are defined as

$$\begin{aligned} P_n^{(\alpha,\beta)}(x) &= (n+\alpha)!\,(n+\beta)!\sum_s \frac{1}{s!\,(n+\alpha-s)!\,(\beta+s)!\,(n-s)!} \\ &\quad \times \left(\frac{x-1}{2}\right)^{n-s}\left(\frac{x+1}{2}\right)^{s}. \end{aligned} \tag{3.18}$$

where the superscripts α and β need to be determined through the orthogonality condition

$$\begin{aligned} \int_{-1}^{1}(1-x)^{\alpha}(1+x)^{\beta}P_m^{(\alpha,\beta)}(x)P_n^{(\alpha,\beta)}(x)dx &= \frac{2^{\alpha+\beta+1}}{2n+\alpha+\beta+1} \\ &\quad \times\frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)n!}. \end{aligned} \tag{3.19}$$

First, we want the integral to go from 0 to 1. Substitute $x = 2x - 1$. This gives,

$$\int_0^1 2^{\alpha+\beta}(1-x)^{\alpha}(x)^{\beta}P_m^{(\alpha,\beta)}(x)P_n^{(\alpha,\beta)}(x)dx. \tag{3.20}$$

Overall we want

$$\int_0^1 (1-x)xP_m^{(1)}(x)P_n^{(1)}(x)dx = \delta_{nm}. \tag{3.21}$$

Matching the weight functions gives $\alpha = \beta = 1$ and since we made the integral only positive, $\Gamma(n) = (n-1)!$. This means

$$\frac{2^{\alpha+\beta+1}}{2n+\alpha+\beta+1}\frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)n!} = \frac{2^3}{2n+3}\frac{(n+1)!\,(n+1)!}{(n+2)!\,n!}. \tag{3.22}$$

Finally we simplify the factorials and move everything to one side to result in

$$\int_0^1 x(1-x)\frac{2n+3}{2}\frac{n+2}{n+1}P_n^{(1,1)}(x)P_m^{(1,1)}(x)dx = \delta_{nm}. \tag{3.23}$$

Comparing this with what we wanted yields

$$P_n^{(1)}(x) = \sqrt{\left(\frac{2n+3}{2}\frac{n+2}{n+1}\right)}P_n^{(1,1)}(x). \tag{3.24}$$

Each equation from our system of equations earlier is then projected onto each basis function. The first equation will stay the same while each term in the second equation will be multiplied by $\sqrt{x_1(1-x_1)}P_n^{(1)}(x_1)$ and integrated from 0 to 1 with respect to x_1. This will make each term in the equation a matrix that we can multiply by a vector containing a_0 to a_N. We will then divide both sides in both equations by μ^2 to make our mass a ratio to essentially eliminate a variable and also replace $\frac{g}{\mu^2\sqrt{4\pi}}$ with λ for simplicity. We will define

$$\frac{m^2}{\mu^2} = \tilde{m}^2. \tag{3.25}$$

Summations over the index of the coefficients a_n will be implied from now on. With these substitutions and simplification we get

$$\frac{M^2}{\mu^2} = \tilde{m}^2 + \lambda \int_0^1 dy\ a_n P_n^{(1)}(y), \tag{3.26}$$

$$\begin{aligned}
0 = {} & \lambda \int_0^1 dx_1\ P_n^{(1)}(x_1) + \int_0^1 dx_1\ (1-x_1) a_m P_n^{(1)}(x_1) P_m^{(1)}(x_1) \\
& + \frac{1}{2}\lambda \int_0^1 \int_0^{1-x_1} \frac{dy dx_1\ x_1 a_m a_l P_n^{(1)}(x_1) P_m^{(1)}(x_1) P_l^{(1)}(\frac{y}{1-x_1})}{(1-x_1)} \\
& + \tilde{m}^2 \int_0^1 dx_1\ x_1 a_m P_n^{(1)}(x_1) P_m^{(1)}(x_1) \\
& + \frac{1}{2}\lambda \int_0^1 \int_0^{1-x_1} \frac{dy dx_1\ x_1 a_m a_l P_n^{(1)}(x_1) P_m^{(1)}(y) P_l^{(1)}(\frac{x_1}{1-y})}{(1-y)} \\
& - \tilde{m}^2 \int_0^1 dx_1\ x_1(1-x_1) a_m P_n^{(1)}(x_1) P_m^{(1)}(x_1) \\
& - \lambda \int_0^1 dy\ a_l P_l^{(1)}(y) \int_0^1 dx_1\ x_1(1-x_1) a_m P_n^{(1)}(x_1) P_m^{(1)}(x_1).
\end{aligned} \tag{3.27}$$

We don't want to be working with any fractions when we actually try to solve these equations. To simplify things, any fraction inside the basis function will be substituted with the most common substitutions being $z = \frac{y}{1-x_1}$. This does not always simplify completely and the order of integration may need to be swapped as well. For this set of equations, only one such case exists and is

$$\int_0^1 \int_0^{1-x_1} dy dx_1 \;=\; \int_0^1 \int_0^{1-y} dx_1 dy. \tag{3.28}$$

Any further simplifications will be done such as simplification and combination of terms. This results in

$$\frac{M^2}{\mu^2} \;=\; \tilde{m}^2 + \lambda \int_0^1 dy\ a_n P_n^{(1)}(y), \tag{3.29}$$

$$\begin{aligned} 0 \;=\; & \lambda \int_0^1 dx_1\ P_n^{(1)}(x_1) + \int_0^1 dx_1\ (1-x_1) a_m P_n^{(1)}(x_1) P_m^{(1)}(x_1) \\ & + \frac{1}{2}\lambda \int_0^1 dz\ a_l P_l^{(1)}(z) \int_0^1 dx_1\ x_1 a_m P_n^{(1)}(x_1) P_m^{(1)}(x_1) \\ & + \frac{1}{2}\lambda \int_0^1 \int_0^1 dz dy\ z(1-y) a_m a_l P_n^{(1)}(z(1-y)) P_m^{(1)}(y) P_l^{(1)}(z) \\ & + \tilde{m}^2 \int_0^1 dx_1\ x_1^2 a_m P_n^{(1)}(x_1) P_m^{(1)}(x_1) \\ & - \lambda \int_0^1 dy\ a_l P_l^{(1)}(y) \int_0^1 dx_1\ x_1(1-x_1) a_m P_n^{(1)}(x_1) P_m^{(1)}(x_1). \end{aligned} \tag{3.30}$$

Finally, each term in the second equation will be re-written in terms of matrices and vectors. Each element in the matrix is a term in the sum and the vectors will consist of the appropriate coefficients for that matrix defined

$$A \;=\; \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, \tag{3.31}$$

$$B = \begin{pmatrix} a_1a_1 \\ a_2a_1 \\ \vdots \\ a_na_1 \\ a_1a_2 \\ \vdots \\ a_na_n \end{pmatrix}.$$

The names for the matrices are picked to help recall which diagram they represent. This makes the equation become

$$\frac{M^2}{\mu^2} = \tilde{m}^2 + \lambda \int_0^1 dy\ a_n P_n^{(1)}(y), \tag{3.32}$$

$$\begin{aligned} 0 &= \lambda PTloop + PTu.A + \frac{\lambda}{2} PTloopT.B \\ &+ \frac{\lambda}{2} PTTloop.B + \tilde{m}^2 TP.A - \lambda TPT.B. \end{aligned} \tag{3.33}$$

These are now ready to be solved using a system solver function in Mathematica. The code is included in Appendix A.

3.3 Second Approximation

Now we will allow one or two particles to be created at a time making $T = T_1 + T_2$. This makes our effective Hamiltonian

$$\begin{aligned} \overline{\mathcal{P}}^- &= \Big(1 - (T_1+T_2) + \frac{(T_1+T_2)^2}{2} - \frac{(T_1+T_2)^3}{6}\Big)\mathcal{P}^-\Big(1 + (T_1+T_2) \\ &+ \frac{(T_1+T_2)^2}{2} + \frac{(T_1+T_2)^3}{6}\Big). \end{aligned} \tag{3.34}$$

The same truncation was applied and results in the final effective Hamiltonian to be

$$\overline{\mathcal{P}}^- = \mathcal{P}^- + \mathcal{P}^-T_1 + \mathcal{P}^-T_2 + \frac{\mathcal{P}^-T_1^2}{2} + \frac{\mathcal{P}^-T_1T_2}{2} + \frac{\mathcal{P}^-T_2T_1}{2} + \frac{\mathcal{P}^-T_1^3}{6} \tag{3.35}$$

$$-T_1\mathcal{P}^- - T_1\mathcal{P}^-T_1 - T_1\mathcal{P}^-T_2 - \frac{T_1\mathcal{P}^-T_1^2}{2} - T_2\mathcal{P}^- - T_2\mathcal{P}^-T_1$$
$$+\frac{T_1^2\mathcal{P}^-}{2} + \frac{T_1^2\mathcal{P}^-T_1}{2}.$$

We will use the same diagram rules used for the first order approximation to make our equations with one additional rule. Figure 3.10 consists of T_2 acting on an incoming charged particle with momentum P. The contribution this vertex makes is a factor of $t_2(q_1', q_2', P - q_1' - q_2')$ with arguments consisting of all three outgoing momentum.

Our system of equations will now consist of three equations. One for the valance sector, and one each for the sectors with one and two neutrals. Putting these together gives

$$\frac{M^2}{P} = \frac{m^2}{P} + \int_0^P \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q, P-q)}{\sqrt{q(P-q)P}}, \tag{3.36}$$

$$\begin{aligned} 0 = {} & \frac{g}{\sqrt{4\pi}} \frac{1}{\sqrt{q_1'(P-q_1')P}} + \frac{\mu^2}{q_1'} t_1(q_1', P-q_1') + \frac{m^2}{P-q_1'} t_1(q_1', P-q_1') \\ & +2\int_0^{P-q_1'} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_2(q_1', q, P-q_1'-q)}{\sqrt{q(P-q_1'-q)(P-q_1')}} - \frac{m^2}{P} t_1(q_1', P-q_1') \\ & +\frac{1}{2}\int_0^{P-q_1'} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q_1', P-q_1') t_1(q, P-q_1'-q)}{\sqrt{q(P-q_1')(P-q_1'-q)}} \\ & +\frac{1}{2}\int_0^{P-q_1'} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q, P-q) t_1(q_1', P-q_1'-q)}{\sqrt{q(P-q_1')(P-q_1'-q)}} \\ & -\int_0^{P} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q_1', P-q_1') t_1(q, P-q)}{\sqrt{q(P-q)P}}, \end{aligned} \tag{3.37}$$

$$\begin{aligned} 0 = {} & \Big[\frac{g}{\sqrt{4\pi}} \frac{t_1(q_1', P-q_1')}{\sqrt{q_2'(P-q_1')(P-q_1'-q_2')}} + (q_1' \leftrightarrow q_2')\Big] \\ & +2\frac{m^2}{P-q_1'-q_2'} t_2(q_1', q_2', P-q_1'-q_2') \\ & +2\frac{\mu^2}{q_1'} t_2(q_1', q_2', P-q_1'-q_2') + 2\frac{\mu^2}{q_2'} t_2(q_1', q_2', P-q_1'-q_2') \\ & +\Big[\frac{1}{2}\frac{\mu^2}{q_1'} t_1(q_1', P-q_1') t_1(q_2', P-q_1'-q_2') + (q_1' \leftrightarrow q_2')\Big] \end{aligned} \tag{3.38}$$

$$+\Big[\frac{1}{2}\frac{\mu^2}{q_2'}t_1(q_1',P-q_1')t_1(q_2',P-q_1'-q_2')+(q_1'\leftrightarrow q_2')\Big]$$

$$+\Big[\frac{1}{2}\frac{m^2}{P-q_1'-q_2'}t_1(q_1',P-q_1')t_1(q_2',P-q_1'-q_2')+(q_1'\leftrightarrow q_2')\Big]$$

$$+\int_0^{P-q_1'-q_2'}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q,P-q_1'-q_2'-q)t_2(q_1',q_2',P-q_1'-q_2')}{\sqrt{q(P-q_1'-q_2')(P-q_1'-q_2'-q)}}$$

$$+\Big[\int_0^{P-q_1'-q_2'}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q_1',P-q_1'-q_2'-q)t_2(q_1',q,P-q_1'-q)}{\sqrt{q(P-q_1'-q_2')(P-q_1'-q_2'-q)}}+(q_1'\leftrightarrow q_2')\Big]$$

$$+\Big[\int_0^{P-q_1'-q_2{}'}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q_1',P-q_1')t_2(q,q_2',P-q_1'-q_2'-q)}{\sqrt{q(P-q_1'-q_2')(P-q_1'-q_2'-q)}}+(q_1'\leftrightarrow q_2')\Big]$$

$$+\Big[\int_0^{P-q_1'-q_2{}'}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q,P-q)t_2(q_1',q_2',P-q_1'-q_2'-q)}{\sqrt{q(P-q_1'-q_2')(P-q_1'-q_2'-q)}}\Big]$$

$$+\Big[\frac{1}{6}\int_0^{P-q_1'-q_2{}'}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q,P-q)t_1(q_1',P-q_1'-q)}{\sqrt{q(P-q_1'-q_2')(P-q_1'-q_2'-q)}}$$

$$\times t_1(q_2',P-q_1'-q_2'-q)+(q_1'\leftrightarrow q_2')\Big]$$

$$+\Big[\frac{1}{6}\int_0^{P-q_1'-q_2{}'}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q,P-q_1'-q)t_1(q_1',P-q_1')}{\sqrt{q(P-q_1'-q_2')(P-q_1'-q_2'-q)}}$$

$$\times t_1(q_2',P-q_1'-q_2'-q)+(q_1'\leftrightarrow q_2')\Big]$$

$$+\Big[\frac{1}{6}\int_0^{P-q_1'-q_2{}'}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q,P-q_1'-q_2'-q)t_1(q_1',P-q_1')}{\sqrt{q(P-q_1'-q_2')(P-q_1'-q_2'-q)}}$$

$$\times t_1(q_2',P-q_1'-q_2')+(q_1'\leftrightarrow q_2')\Big]$$

$$-\Big[\frac{g}{\sqrt{4\pi}}\frac{t_1(q_2',P-q_1'-q_2')}{\sqrt{q_1'P(P-q_1')}}-(q_1'\leftrightarrow q_2')\Big]$$

$$-\Big[\frac{m^2}{P-q_1'}t_1(q_1',P-q_1')t_1(q_2',P-q_1'-q_2')-(q_1'\leftrightarrow q_2')\Big]$$

$$-\Big[\frac{\mu^2}{q_2'}t_1(q_1',P-q_1')t_1(q_2',P-q_1'-q_2')-(q_1'\leftrightarrow q_2')\Big]$$

$$-\Big[2\int_0^{P-q_1'}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q_2',P-q_1'-q_2')t_2(q_1',q,P-q_1'-q)}{\sqrt{q(P-q_1')(P-q_1'-q)}}$$

$$-(q_1'\leftrightarrow q_2')\Big]$$

$$-\Big[\frac{1}{2}\int_0^{P-q_1'-}\frac{g}{\sqrt{4\pi}}\frac{dq\ t_1(q,P-q)t_1(q_1',P-q_1'-q)t_1(q_2',P-q_1'-q_2')}{\sqrt{q(P-q_1')(P-q_1'-q)}}$$

$$
\begin{aligned}
&-(q_1' \leftrightarrow q_2')\Big] \\
&-\Big[\frac{1}{2}\int_0^{P-q_1'} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q,P-q_1'-q)t_1(q_1',P-q_1')t_1(q_2',P-q_1'-q_2')}{\sqrt{q(P-q_1')(P-q_1'-q)}} \\
&-(q_1' \leftrightarrow q_2')\Big] \\
&-\frac{m^2}{P}t_2(q_1',q_2',P-q_1'-q_2') \\
&-2\int_0^{P} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q,P-q)t_2(q_1',q_2',P-q_1'-q_2')}{\sqrt{q(P-q)P}} \\
&+\Big[\frac{1}{2}\frac{m^2}{P}t_1(q_1',P-q_1')t_1(q_2',P-q_1'-q_2') + (q_1' \leftrightarrow q_2')\Big] \\
&+\Big[\frac{1}{2}\int_0^{P} \frac{g}{\sqrt{4\pi}} \frac{dq\ t_1(q,P-q)t_1(q_1',P-q_1'-q)t_1(q_2',P-q_1'-q_2')}{\sqrt{q(P-q)(P)}} \\
&+(q_1' \leftrightarrow q_2')\Big].
\end{aligned}
$$

Equation 3.36 does not change from Equation 3.7. Equation 3.37 has only one term different from 3.8. This extra term is the fourth term in Equation 3.37 and corresponds to Figure 3.11. All other terms in Equations 3.36-37 correspond to the same diagrams as Equations 3.7-8 respectively.

Equation 3.38 is the projection onto the 2-neutral sector where each diagram will have three final particles. The first term on the right hand side corresponds to Figure 3.12. The figures are kept in order of the terms with the next term, that is multiplied by mass m, being from Figure 3.13. The next line, with the two terms with neutral mass μ, both correspond to Figure 3.14. The Hamiltonian can act on either particle which gives two separate terms. The next neutral mass term originates from Figure 3.15. This has the Hamiltonian still acting on the first particle but acts last on the system. Figure 3.16 illustrates the next term with the Hamiltonian acting on the second particle while acting on the system last. The next term comes from Figure 3.17. Most the terms from now on will include a neutral particle being annihilated. This is illustrated by a loop in the diagrams and an integral in the equations. The first term with an integral corresponds to Figure 3.18. The next term is for Figure 3.19. The next term has a loop going from the T_1 to the Hamiltonian. This is shown in Figure 3.21 and notice how the interchange of q_1' and q_2' does not change the diagram. This is why the 11th term in Equation 3.38 does not have an interchange term added on. The next three terms are all from $\mathcal{P}^- T_1^3$. They differ by having a different neutral in each case annihilated and originate from Figures 3.22-24 respectively. The remaining terms all have the T operator acting at least once after the Hamiltonian. The first term has the Hamiltonian creating a particle and is represented by Figure 3.25. The next two terms are mass terms that will later be combined with earlier terms for simplification issues. They match with Figure 3.26 and Figure 3.27. The next is a mixing term between T_1 and T_2 that are from Figure 3.28. All mixing terms are difficult to work with. They tend to need the most work to simplify and are not symmetric in nature. This leads to problems in a computational environment. The next two terms correspond to $T\mathcal{P}^- T^2$ with different neutrals being annihilated. These are matched to Figure 3.30-31. Figure 3.31 creates the next charged mass term. Figure 3.32 corresponds to the next term. The second to last term is another charged mass term and is represented by Figure 3.33 while the last term in Equation 3.38 is from Figure 3.34.

We will substitute the momentum fractions as we did with the first order approximation with the addition $q_2' = x_2 P$, giving

$$\frac{M^2}{P} = \frac{m^2}{P} + \int_0^1 \frac{dy\ t_1(yP,(1-y)P)}{\sqrt{y(1-y)P}}, \tag{3.39}$$

$$\begin{aligned} 0 = {} & \frac{g}{\sqrt{4\pi}} \frac{1}{\sqrt{x_1(1-x_1)P^3}} + \frac{\mu^2}{x_1 P} t_1(x_1 P,(1-x_1)P) \\ & + \frac{m^2}{(1-x_1)P} t_1(x_1 P,(1-x_1)P) \\ & + 2\int_0^{1-x_1} \frac{g}{\sqrt{4\pi}} \frac{dy\ t_2(x_1 P, yP,(1-x_1-y)P)}{\sqrt{y(1-x_1-y)(1-x_1)P}} - \frac{m^2}{P} t_1(x_1 P,(1-x_1)P) \\ & + \frac{1}{2}\int_0^{1-x_1} \frac{g}{\sqrt{4\pi}} \frac{dy\ t_1(x_1 P,(1-x_1)P) t_1(yP,(1-x_1-y)P)}{\sqrt{y(1-x_1)(1-x_1-y)P}} \\ & + \frac{1}{2}\int_0^{1-x_1} \frac{g}{\sqrt{4\pi}} \frac{dy\ t_1(yP,(1-y)P) t_1(x_1 P,(1-x_1-y)P)}{\sqrt{y(1-x_1)(1-x_1-y)P}} \\ & - \int_0^1 \frac{g}{\sqrt{4\pi}} \frac{dy\ t_1(x_1 P,(1-x_1)P) t_1(yP,(1-y)P)}{\sqrt{y(1-y)P}}, \end{aligned} \tag{3.40}$$

$$0 = \Big[\frac{g}{\sqrt{4\pi}} \frac{t_1(x_1 P,(1-x_1)P)}{\sqrt{x_2(1-x_1)(1-x_1-x_2)P^3}} + (x_1 \leftrightarrow x_2)\Big] \tag{3.41}$$

$$+2\frac{m^2}{(1-x_1-x_2)P}t_2(x_1P,x_2P,(1-x_1-x_2)P)$$
$$+2\frac{\mu^2}{x_1P}t_2(x_1P,x_2P,(1-x_1-x_2)P)+2\frac{\mu^2}{x_2P}t_2(x_1P,x_2P,(1-x_1-x_2)P)$$
$$+\Big[\frac{1}{2}\frac{\mu^2}{x_1P}t_1(x_1P,(1-x_1)P)t_1(x_2P,(1-x_1-x_2)P)+(x_1\leftrightarrow x_2)\Big]$$
$$+\Big[\frac{1}{2}\frac{\mu^2}{x_2P}t_1(x_1P,(1-x_1)P)t_1(x_2P,(1-x_1-x_2)P)+(x_1\leftrightarrow x_2)\Big]$$
$$+\Big[\frac{1}{2}\frac{m^2}{(1-x_1-x_2)P}t_1(x_1P,(1-x_1)P)t_1(x_2P,(1-x_1-x_2)P)+(x_1\leftrightarrow x_2)\Big]$$
$$+\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-x_1-x_2-y)P)}{\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)P}}$$
$$\times t_2(x_1P,x_2P,(1-x_1-x_2)P)$$
$$+\Big[\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(x_1P,(1-x_1-x_2-y)P)}{\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)P}}$$
$$\times t_2(x_1P,yP,(1-x_1-y)P)+(x_1\leftrightarrow x_2)\Big]$$
$$+\Big[\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(x_1P,(1-x_1)P)}{\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)P}}$$
$$\times t_2(yP,x_2P,(1-x_1-x_2-y)P)+(x_1\leftrightarrow x_2)\Big]$$
$$+\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-y)P)t_2(x_1P,x_2P,(1-x_1-x_2-y)P)}{\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)P}}$$
$$+\Big[\frac{1}{6}\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-y)P)t_1(x_1P,(1-x_1-y)P)}{\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)}}$$
$$\times t_1(x_2P,(1-x_1-x_2-y)P)+(x_1\leftrightarrow x_2)\Big]$$
$$+\Big[\frac{1}{6}\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-x_1-y)P)t_1(x_1,(1-x_1)P)}{\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)P}}$$
$$\times t_1(x_2P,(1-x_1-x_2-y)P)+(x_1\leftrightarrow x_2)\Big]$$
$$+\Big[\frac{1}{6}\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-x_1-x_2-y)t_1(x_1P,(1-x_1)P)}{\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)}}$$
$$\times t_1(x_2P,(1-x_1-x_2)P)+(x_1\leftrightarrow x_2)\Big]$$
$$-\Big[\frac{g}{\sqrt{4\pi}}\frac{t_1(x_2P,(1-x_1-x_2)P)}{\sqrt{x_1(1-x_1)P^3}}-(x_1\leftrightarrow x_2)\Big]$$

$$
\begin{aligned}
&-\Big[\frac{m^2}{(1-x_1)P}t_1(x_1P,(1-x_1)P)t_1(x_2P,(1-x_1-x_2)P)-(x_1\leftrightarrow x_2)\Big]\\
&-\Big[\frac{\mu^2}{x_2P}t_1(x_1P,(1-x_1)P)t_1(x_2P,(1-x_1-x_2)P)-(x_1\leftrightarrow x_2)\Big]\\
&-\Big[2\int_0^{1-x_1}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(x_2P,(1-x_1-x_2)P)t_2(x_1P,yP,(1-x_1-y)P)}{\sqrt{y(1-x_1)(1-x_1-y)P}}\\
&-(x_1\leftrightarrow x_2)\Big]\\
&-\Big[\frac{1}{2}\int_0^{1-x_1}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-y)P)t_1(x_1P,(1-x_1-y)P)}{\sqrt{y(1-x_1)(1-x_1-y)P}}\\
&\times t_1(x_2P,(1-x_1-x_2)P)-(x_1\leftrightarrow x_2)\Big]\\
&-\Big[\frac{1}{2}\int_0^{1-x_1}\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-x_1-y)P)t_1(x_1P,(1-x_1)P)}{\sqrt{y(1-x_1)(1-x_1-y)P}}\\
&\times t_1(x_2P,(1-x_1-x_2)P)-(x_1\leftrightarrow x_2)\Big]\\
&-2\frac{m^2}{P}t_2(x_1P,x_2P,(1-x_1-x_2)P)\\
&-2\int_0^1\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-y)P)t_2(x_1P,x_2P,(1-x_1-x_2)P)}{\sqrt{y(1-y)P}}\\
&+\Big[\frac{1}{2}\frac{m^2}{P}t_1(x_1P,(1-x_1)P)t_1(x_2P,(1-x_1-x_2)P)+(x_1\leftrightarrow x_2)\Big]\\
&+\Big[\frac{1}{2}\int_0^1\frac{g}{\sqrt{4\pi}}\frac{dy\ t_1(yP,(1-y)P)t_1(x_1P,(1-x_1-y)P)}{\sqrt{y(1-y)P}}\\
&\times t_1(x_2,(1-x_1-x_2)P)+(x_1\leftrightarrow x_2)\Big].
\end{aligned}
$$

We now change from t_n to $\tilde{t}_n$ as we did for the first approximation but

$$t_2(x_1P,x_2P,(1-x_1-x_2)P)=P^{\beta}\tilde{t}_2(x_1,x_2), \tag{3.42}$$

with β potentially being different from $-\frac{1}{2}$. The same rules apply for determining the argument for $\tilde{t}_2$ where we add all the arguments from t_2 but this time we are left with the first and second argument divided by the result. For example

$$t_2(x_1P,x_2P,(1-x_1-x_2-y)P)=((1-y)P)^{\beta}\tilde{t}_2\left(\frac{x_1}{1-y},\frac{x_2}{1-y}\right). \tag{3.43}$$

We determine β in the same manner as for t_1. This time with the n=2 term compared to $T_2\left|\phi\right\rangle$, we have

$$\left|\psi_\pm(P)\right\rangle = (P)^1 \int dx_1 dx_2 \psi_2^\pm(x_1,x_2) c_\pm^\dagger((1-x_1-x_2)P) a^\dagger(x_1P) a^\dagger(x_2P)\left|0\right\rangle \tag{3.44}$$

and

$$T_2\left|\phi\right\rangle = P^{\beta+2} \int dx_1 dx_2 \tilde{t}_2(x_1,x_2) a^\dagger(x_1P) a^\dagger(x_2P) c_\pm^\dagger((1-x_1-x_2)P) c_\pm(P)\left|\phi\right\rangle. \tag{3.45}$$

This clearly results in $\beta = -1$. Using this substitution we rewrite our equations again with canceling any P and simplifying wherever we can. This yields

$$M^2 = m^2 + \int_0^1 \frac{g}{\sqrt{4\pi}} \frac{dy\ \tilde{t}_1(y)}{\sqrt{y(1-y)}}, \tag{3.46}$$

$$\begin{aligned} 0 = {} & \frac{g}{\sqrt{4\pi}} \frac{1}{\sqrt{x_1(1-x_1)}} + \frac{\mu^2}{x_1}\tilde{t}_1(x_1) \\ & + \frac{m^2}{1-x_1}\tilde{t}_1(x_1) \\ & + 2\int_0^{1-x_1} \frac{g}{\sqrt{4\pi}} \frac{dy\ \tilde{t}_2(x_1,y)}{\sqrt{y(1-x_1-y)(1-x_1)}} - m^2\tilde{t}_1(x_1) \\ & + \frac{1}{2}\int_0^{1-x_1} \frac{g}{\sqrt{4\pi}} \frac{dy\ \tilde{t}_1(x_1)\tilde{t}_1(\frac{y}{1-x_1})}{(1-x_1)\sqrt{y(1-x_1-y)}} \\ & + \frac{1}{2}\int_0^{1-x_1} \frac{g}{\sqrt{4\pi}} \frac{dy\ \tilde{t}_1(y)\tilde{t}_1(\frac{x_1}{1-y})}{\sqrt{y(1-x_1)(1-y)(1-x_1-y)}} \\ & - \int_0^1 \frac{g}{\sqrt{4\pi}} \frac{dy\ \tilde{t}_1(x_1)\tilde{t}_1(y)}{\sqrt{y(1-y)}}, \end{aligned} \tag{3.47}$$

$$\begin{aligned} 0 = {} & \Big[\frac{g}{\sqrt{4\pi}} \frac{\tilde{t}_1(x_1)}{\sqrt{x_2(1-x_1)(1-x_1-x_2)}} + (x_1 \leftrightarrow x_2)\Big] \\ & + 2\frac{m^2}{1-x_1-x_2}\tilde{t}_2(x_1,x_2) \\ & + 2\frac{\mu^2}{x_1}\tilde{t}_2(x_1,x_2) + 2\frac{\mu^2}{x_2}\tilde{t}_2(x_1,x_2) \\ & + \Big[\frac{1}{2}\frac{\mu^2}{x_1}\frac{\tilde{t}_1(x_1)\tilde{t}_1(\frac{x_2}{1-x_1})}{\sqrt{(1-x_1)}} + (x_1 \leftrightarrow x_2)\Big] \end{aligned} \tag{3.48}$$

$$+\Big[\frac{1}{2}\frac{\mu^2}{x_2}\frac{\tilde{t}_1(x_1)\tilde{t}_1(\frac{x_2}{1-x_1})}{\sqrt{(1-x_1)}}+(x_1\leftrightarrow x_2)\Big]$$

$$+\Big[\frac{1}{2}\frac{m^2}{1-x_1-x_2}\frac{\tilde{t}_1(x_1)\tilde{t}_1(\frac{x_2}{1-x_1})}{\sqrt{(1-x_1)}}+(x_1\leftrightarrow x_2)\Big]$$

$$+\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(\frac{y}{1-x_1-x_2})\tilde{t}_2(x_1,x_2)}{(1-x_1-x_2)\sqrt{y(1-x_1-x_2-y)}}$$

$$+\Big[\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(\frac{x_2}{1-x_1-y})\tilde{t}_2(x_1,y)}{\sqrt{y(1-x_1-y)(1-x_1-x_2)(1-x_1-x_2-y)}}+(x_1\leftrightarrow x_2)\Big]$$

$$+\Big[\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(x_1)\tilde{t}_2(\frac{y}{1-x_1},\frac{x_2}{1-x_1})}{(1-x_1)\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)}}+(x_1\leftrightarrow x_2)\Big]$$

$$+\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(y)\tilde{t}_2(\frac{x_1}{1-y},\frac{x_2}{1-y})}{(1-y)\sqrt{y(1-x_1-x_2)(1-x_1-x_2-y)}}$$

$$+\Big[\frac{1}{6}\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(y)\tilde{t}_1(\frac{x_1}{1-y})}{\sqrt{y(1-y)(1-x_1-x_2)(1-x_1-y)(1-x_1-x_2-y)}}$$

$$\times\tilde{t}_1(\frac{x_2}{1-x_1-y})+(x_1\leftrightarrow x_2)\Big]$$

$$+\Big[\frac{1}{6}\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(\frac{y}{1-x_1})\tilde{t}_1(x_1)}{\sqrt{y(1-x_1)(1-x_1-x_2)(1-x_1-y)(1-x_1-x_2-y)}}$$

$$\times\tilde{t}_1(\frac{x_2}{1-x_1-y})+(x_1\leftrightarrow x_2)\Big]$$

$$+\Big[\frac{1}{6}\int_0^{1-x_1-x_2}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(\frac{y}{1-x_1-x_2})\tilde{t}_1(x_1)}{(1-x_1-x_2)\sqrt{y(1-x_1)(1-x_1-x_2-y)}}$$

$$\times\tilde{t}_1(\frac{x_2}{1-x_1})+(x_1\leftrightarrow x_2)\Big]$$

$$-\Big[\frac{g}{\sqrt{4\pi}}\frac{\tilde{t}_1(\frac{x_2}{1-x_1})}{(1-x_1)\sqrt{x_1}}-(x_1\leftrightarrow x_2)\Big]-\Big[\frac{m^2}{(1-x_1)^{\frac{3}{2}}}\tilde{t}_1(x_1)\tilde{t}_1(\frac{x_2}{1-x_1})-(x_1\leftrightarrow x_2)\Big]$$

$$-\Big[\frac{\mu^2}{x_1}\frac{\tilde{t}_1(x_1)\tilde{t}_1(\frac{x_2}{1-x_1})}{\sqrt{(1-x_1)}}-(x_1\leftrightarrow x_2)\Big]$$

$$-\Big[2\int_0^{1-x_1}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(\frac{x_2}{1-x_1})\tilde{t}_2(x_1,y)}{(1-x_1)\sqrt{y(1-x_1-y)}}-(x_1\leftrightarrow x_2)\Big]$$

$$-\Big[\frac{1}{2}\int_0^{1-x_1}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(y)\tilde{t}_1(\frac{x_1}{1-y})\tilde{t}_1(\frac{x_2}{1-x_1})}{(1-x_1)\sqrt{y(1-y)(1-x_1-y)}}-(x_1\leftrightarrow x_2)\Big]$$

$$-\Big[\frac{1}{2}\int_0^{1-x_1}\frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(\frac{y}{1-x_1})\tilde{t}_1(x_1)\tilde{t}_1(\frac{x_2}{1-x_1})}{(1-x_1)^{\frac{3}{2}}\sqrt{y(1-x_1-y)}}-(x_1\leftrightarrow x_2)\Big]$$

$$-2m^2\tilde{t}_2(x_1,x_2) - 2\int_0^1 \frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(y)\tilde{t}_2(x_1,x_2)}{\sqrt{y(1-y)}}$$
$$+\Big[\frac{1}{2}\frac{m^2}{\sqrt{(1-x_1)}}\tilde{t}_1(x_1)\tilde{t}_1(\frac{x_2}{1-x_1}) + (x_1 \leftrightarrow x_2)\Big]$$
$$+\Big[\frac{1}{2}\int_0^1 \frac{g}{\sqrt{4\pi}}\frac{dy\ \tilde{t}_1(y)\tilde{t}_1(x_1)\tilde{t}_1(\frac{x_2}{1-x_1})}{\sqrt{y(1-y)(1-x_1)}} + (x_1 \leftrightarrow x_2)\Big].$$

Now again, we want to apply numerical methods similar to those in section 2.4. We will use the same substitution for $\tilde{t}_2(x_1,x_2)$ but for $\tilde{t}_1(x)$ we will be using the weighted Jacobi Polynomial substitution as before. We will use the same notation as before

$$\tilde{t}_1(x) = \sum_n a_n\sqrt{x(1-x)}P_n^{(1)}(x), \tag{3.49}$$
$$\tilde{t}_2(x,y) = \sum_n c_n\sqrt{xy(1-x-y)}P_n^{(2)}(x,y). \tag{3.50}$$

The summations are again implied. Here, $P_n^{(1)}(x)$ is a Jacobi polynomials while $P_n^{(2)}(x,y) = Q_{Nk}^{(2)}(x_1,x_2)$ from section 2.4. Here n is a generic index that corresponds to both the order N and the label k, the latter making the distinction between multivariate polynomials of the same order. The same projection is made but with the third equation getting

$$\int_0^1\int_0^{1-x_1} dx_2dx_1\ \sqrt{x_1x_2(1-x_1-x_2)}P_n^{(2)}(x_1,x_2).$$

With these substitutions and simplification we get

$$\frac{M^2}{\mu^2} = \tilde{m}^2 + \lambda\int_0^1 dy\ a_nP_n^{(1)}(y), \tag{3.51}$$

$$0 = \lambda\int_0^1 dx_1\ P_n^{(1)}(x_1) + \int_0^1 dx_1\ (1-x_1)a_mP_n^{(1)}(x_1)P_m^{(1)}(x_1) \tag{3.52}$$
$$+\frac{1}{2}\lambda\int_0^1\int_0^{1-x_1}\frac{dydx_1\ x_1a_ma_lP_n^{(1)}(x_1)P_m^{(1)}(x_1)P_l^{(1)}(\frac{y}{1-x_1})}{(1-x_1)}$$
$$+2\lambda\int_0^1\int_0^{1-x_1} dydx_1\ x_1P_n^{(1)}(x_1)c_mP_m^{(2)}(x_1,y)$$

$$
\begin{aligned}
&+\tilde{m}^2\int_0^1 dx_1\ x_1 a_m P_n^{(1)}(x_1)P_m^{(1)}(x_1)\\
&+\frac{1}{2}\lambda\int_0^1\int_0^{1-x_1}\frac{dydx_1\ x_1 a_m a_l P_n^{(1)}(x_1)P_m^{(1)}(y)P_l^{(1)}(\frac{x_1}{1-y})}{(1-y)}\\
&-\tilde{m}^2\int_0^1 dx_1\ x_1(1-x_1)a_m P_n^{(1)}(x_1)P_m^{(1)}(x_1)\\
&-\lambda\int_0^1 dy\ a_l P_l^{(1)}(y)\int_0^1 dx_1\ x_1(1-x_1)a_m P_n^{(1)}(x_1)P_m^{(1)}(x_1),
\end{aligned}
$$

$$
\begin{aligned}
0 \;=\;& \Big[\lambda\int_0^1\int_0^{1-x_1} dx_2dx_1\ x_1 a_m P_n^{(2)}(x_1,x_2)P_m^{(1)}(x_1)+(x_1\leftrightarrow x_2)\Big] \qquad (3.53)\\
&+2\tilde{m}^2\int_0^1\int_0^{1-x_1} dx_2dx_1\ x_1x_2 c_m P_n^{(2)}(x_1,x_2)P_m^{(2)}(x_1,x_2)\\
&+\Big[2\int_0^1\int_0^{1-x_1} dx_2dx_1\ x_2(1-x_1-x_2)c_m P_n^{(2)}(x_1,x_2)P_m^{(2)}(x_1,x_2)+(x_1\leftrightarrow x_2)\Big]\\
&+\Big[\frac{1}{2}\int_0^1\int_0^{1-x_1} dx_2dx_1\ \frac{x_2(1-x_1-x_2)a_m a_l P_n^{(2)}(x_1,x_2)P_m^{(1)}(x_1)P_l^{(1)}(\frac{x_2}{1-x_1})}{1-x_1}\\
&+(x_1\leftrightarrow x_2)\Big]\\
&+\Big[\frac{1}{2}\int_0^1\int_0^{1-x_1} dx_2dx_1\ \frac{x_1(1-x_1-x_2)a_m a_l P_n^{(2)}(x_1,x_2)P_m^{(1)}(x_1)P_l^{(1)}(\frac{x_2}{1-x_1})}{1-x_1}\\
&+(x_1\leftrightarrow x_2)\Big]\\
&+\Big[\frac{\tilde{m}^2}{2}\int_0^1\int_0^{1-x_1} dx_2dx_1\ \frac{x_1x_2 a_m a_l P_n^{(2)}(x_1,x_2)P_m^{(1)}(x_1)P_l^{(1)}(\frac{x_2}{1-x_1})}{1-x_1}+(x_1\leftrightarrow x_2)\Big]\\
&+\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\frac{dydx_2dx_1\ x_1x_2 a_m c_l P_n^{(2)}(x_1,x_2)P_m^{(1)}(\frac{y}{1-x_1-x_2})}{1-x_1-x_2}\\
&\times P_l^{(2)}(x_1,x_2)\\
&+\Big[\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\frac{dydx_2dx_1\ x_1x_2 a_m c_l P_n^{(2)}(x_1,x_2)P_m^{(1)}(\frac{x_2}{1-x_1-y})}{1-x_1-y}\\
&\times P_l^{(2)}(x_1,y)+(x_1\leftrightarrow x_2)\Big]\\
&+\Big[\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\frac{dydx_2dx_1\ x_1x_2 a_m c_l P_n^{(2)}(x_1,x_2)P_m^{(1)}(x_1)}{(1-x_1)^2}\\
&\times P_l^{(2)}\left(\frac{y}{1-x_1},\frac{x_2}{1-x_1}\right)+(x_1\leftrightarrow x_2)\Big]
\end{aligned}
$$

$$+\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\frac{dydx_2dx_1\ x_1x_2a_mc_lP_n^{(2)}(x_1,x_2)P_m^{(1)}(y)}{(1-y)^2}$$
$$\times P_l^{(2)}\left(\frac{x_1}{1-y},\frac{x_2}{1-y}\right)$$
$$+\Big[\frac{1}{6}\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\frac{dydx_2dx_1\ x_1x_2a_ma_la_sP_n^{(2)}(x_1,x_2)P_m^{(1)}(y)}{(1-x_1-y)(1-y)}$$
$$\times P_l^{(1)}(\frac{x_1}{1-y})P_s^{(1)}\left(\frac{x_2}{1-x_1-y}\right)+(x_1\leftrightarrow x_2)\Big]$$
$$+\Big[\frac{1}{6}\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\frac{dydx_2dx_1\ x_1x_2a_ma_la_sP_n^{(2)}(x_1,x_2)P_m^{(1)}(\frac{y}{1-x_1})}{(1-x_1-y)(1-x_1)}$$
$$\times P_l^{(1)}(x_1)P_s^{(1)}\left(\frac{x_2}{1-x_1-y}\right)+(x_1\leftrightarrow x_2)\Big]$$
$$+\Big[\frac{1}{6}\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\frac{dydx_2dx_1\ x_1x_2a_ma_la_sP_n^{(2)}(x_1,x_2)P_m^{(1)}(\frac{y}{1-x_1-x_2})}{(1-x_1)(1-x_1-x_2)}$$
$$\times P_l^{(1)}(x_1)P_s^{(1)}\left(\frac{x_2}{1-x_1}\right)+(x_1\leftrightarrow x_2)\Big]$$
$$-\Big[\lambda\int_0^1\int_0^{1-x_1}\frac{dx_2dx_1\ x_2(1-x_1-x_2)a_mP_n^{(2)}(x_1,x_2)P_m^{(1)}(\frac{x_2}{1-x_1})}{(1-x_1)^2}-(x_1\leftrightarrow x_2)\Big]$$
$$-\Big[\tilde{m}^2\int_0^1\int_0^{1-x_1}\frac{dx_2dx_1\ x_1x_2(1-x_1-x_2)a_ma_lP_n^{(2)}(x_1,x_2)P_m^{(1)}(x_1)}{(1-x_1)^2}$$
$$\times P_l^{(1)}(\frac{x_2}{1-x_1})-(x_1\leftrightarrow x_2)\Big]$$
$$-\Big[\int_0^1\int_0^{1-x_1}\frac{dx_2dx_1\ x_2(1-x_1-x_2)a_ma_lP_n^{(2)}(x_1,x_2)P_m^{(1)}(x_1)P_l^{(1)}(\frac{x_2}{1-x_1})}{1-x_1}$$
$$-(x_1\leftrightarrow x_2)\Big]$$
$$-\Big[2\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1}\frac{dydx_2dx_1\ x_1x_2(1-x_1-x_2)a_mc_lP_n^{(2)}(x_1,x_2)}{(1-x_1)^2}$$
$$\times P_m^{(1)}\left(\frac{x_2}{1-x_1}\right)P_l^{(2)}(x_1,y)-(x_1\leftrightarrow x_2)\Big]$$
$$-\Big[\frac{1}{2}\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1}\frac{dydx_2dx_1\ x_1x_2(1-x_1-x_2)a_ma_la_sP_n^{(2)}(x_1,x_2)}{(1-x_1)^2(1-y)}$$
$$\times P_m^{(1)}(y)P_l^{(1)}\left(\frac{x_1}{1-y}\right)P_s^{(1)}\left(\frac{x_2}{1-x_1}\right)-(x_1\leftrightarrow x_2)\Big]$$
$$-\Big[\frac{1}{2}\lambda\int_0^1\int_0^{1-x_1}\int_0^{1-x_1}\frac{dydx_2dx_1\ x_1x_2(1-x_1-x_2)a_ma_la_sP_n^{(2)}(x_1,x_2)}{(1-x_1)^3}$$

$$\times P_m^{(1)}\left(\frac{y}{1-x_1}\right)P_l^{(1)}(x_1)P_s^{(1)}\left(\frac{x_2}{1-x_1}\right)-(x_1\leftrightarrow x_2)\Big]$$
$$-2\tilde{m}^2\int_0^1\int_0^{1-x_1}dx_2dx_1\ x_1x_2(1-x_1-x_2)c_mP_n^{(2)}(x_1,x_2)P_m^{(2)}(x_1,x_2)$$
$$-2\lambda\int_0^1\int_0^{1-x_1}\int_0^1 dydx_2dx_1\ x_1x_2(1-x_1-x_2)a_mc_lP_n^{(2)}(x_1,x_2)P_m^{(1)}(y)$$
$$\times P_l^{(2)}(x_1,x_2)$$
$$+\Big[\frac{1}{2}\tilde{m}^2\int_0^1\int_0^{1-x_1}dx_2dx_1\ \frac{x_1x_2(1-x_1-x_2)a_ma_lP_n^{(2)}(x_1,x_2)}{1-x_1}$$
$$\times P_m^{(1)}(x_1)P_l^{(1)}(\frac{x_2}{1-x_1})+(x_1\leftrightarrow x_2)\Big]$$
$$+\Big[\frac{1}{2}\lambda\int_0^1\int_0^{1-x_1}\int_0^1\frac{dydx_2dx_1\ x_1x_2(1-x_1-x_2)a_ma_la_sP_n^{(2)}(x_1,x_2)P_m^{(1)}(y)}{1-x_1}$$
$$\times P_l^{(1)}(x_1)P_s^{(1)}\left(\frac{x_2}{1-x_1}\right)+(x_1\leftrightarrow x_2)\Big].$$

Now we will do the same simplifications we did for the first-order approximation. This includes substituting fractions for single variables and switching integration order. We will also make an interchange of x_1 and x_2 to combine the symmetric terms. Simplification will be used as much as possible but terms will not be actively combined. With so many terms, it becomes difficult to keep track of combined terms. This yields

$$\frac{M^2}{\mu^2} = \tilde{m}^2+\lambda\int_0^1 dy\ a_nP_n^{(1)}(y), \tag{3.54}$$

$$\begin{aligned}0 = {} & \lambda\int_0^1 dx_1\ P_n^{(1)}(x_1)+\int_0^1 dx_1\ (1-x_1)a_mP_n^{(1)}(x_1)P_m^{(1)}(x_1) \\ & +\frac{1}{2}\lambda\int_0^1 dz\ a_lP_l^{(1)}(z)\int_0^1 dx_1\ x_1a_mP_n^{(1)}(x_1)P_m^{(1)}(x_1) \\ & +\frac{1}{2}\lambda\int_0^1\int_0^1 dzdy\ z(1-y)a_ma_lP_n^{(1)}(z(1-y))P_m^{(1)}(y)P_l^{(1)}(z) \\ & +\tilde{m}^2\int_0^1 dx_1\ x_1^2a_mP_n^{(1)}(x_1)P_m^{(1)}(x_1) \\ & -\lambda\int_0^1 dy\ a_lP_l^{(1)}(y)\int_0^1 dx_1\ x_1(1-x_1)a_mP_n^{(1)}(x_1)P_m^{(1)}(x_1). \\ & +2\lambda\int_0^1\int_0^{1-x_1}dydx_1\ x_1c_mP_n^{(1)}(x_1)P_m^{(2)}(x_1,y)\end{aligned} \tag{3.55}$$

$$
\begin{aligned}
0 \;=\; & 2\lambda \int_0^1 \int_0^{1-x_1} dx_2 dx_1 \; x_1 a_m P_n^{(2)}(x_1,x_2) P_m^{(1)}(x_1) \qquad (3.56)\\
& +2\tilde{m}^2 \int_0^1 \int_0^{1-x_1} dx_2 dx_1 \; x_1 x_2 c_m P_n^{(2)}(x_1,x_2) P_m^{(2)}(x_1,x_2)\\
& +4 \int_0^1 \int_0^{1-x_1} dx_2 dx_1 \; x_2(1-x_1-x_2) c_m P_n^{(2)}(x_1,x_2) P_m^{(2)}(x_1,x_2)\\
& + \int_0^1 \int_0^1 dz_2 dx_1 \; z_2(1-x_1)^2(1-z_2) a_m a_l P_n^{(2)}(x_1, z_2(1-x_1)) P_m^{(1)}(x_1)\\
& \times P_l^{(1)}(z_2)\\
& + \int_0^1 \int_0^1 dz_2 dx_1 \; x_1(1-x_1)(1-z_2) a_m a_l P_n^{(2)}(x_1, z_2(1-x_1)) P_m^{(1)}(x_1) P_l^{(1)}(z_2)\\
& +\tilde{m}^2 \int_0^1 \int_0^1 dz_2 dx_1 \; x_1(1-x_1) z_2 a_m a_l P_n^{(2)}(x_1, z_2(1-x_1) P_m^{(1)}(x_1) P_l^{(1)}(z_2)\\
& +\lambda \int_0^1 \int_0^{1-x_1} \int_0^1 dz_1 a_m P_m^{(1)}(z_1) dx_2 dx_1 \; x_1 x_2 c_l P_n^{(2)}(x_1,x_2) P_l^{(2)}(x_1,x_2)\\
& +2\lambda \int_0^1 \int_0^1 \int_0^1 dz_2 dz_1 dx_1 \; x_1 z_2 (1-x_1)^2 (1-z_1) a_m c_l P_n^{(2)}(x_1, z_2(1-x_1)(1-z_1))\\
& \times P_m^{(1)}(z_2) P_l^{(2)}(x_1, z_1(1-x_1))\\
& +2\lambda \int_0^1 \int_0^1 \int_0^1 dz_1 dz_2 dx_1 \; x_1(1-x_1) z_2 (1-z_2) a_m c_l P_n^{(2)}(x_1, z_2(1-x_1))\\
& \times P_m^{(1)}(x_1) P_l^{(2)}(z_1(1-z_2), z_2)\\
& +\lambda \int_0^1 \int_0^1 \int_0^1 dz_2 dz_1 dy \; z_1 z_2 (1-z_1)^2 (1-y)^2 a_m c_l\\
& \times P_n^{(2)}(z_1(1-y), z_2(1-y)(1-z_1)) P_m^{(1)}(y) P_l^{(2)}(z_1, z_2(1-z_1))\\
& +\frac{1}{3}\lambda \int_0^1 \int_0^1 \int_0^1 dz_2 dy dz_1 \; z_1 z_2 (1-y)^2 (1-z_1) a_m a_l a_s\\
& \times P_n^{(2)}(z_1(1-y), z_2(1-z_1)(1-y)) P_m^{(1)}(y) P_l^{(1)}(z_1) P_s^{(1)}(z_2)\\
& +\frac{1}{3}\lambda \int_0^1 \int_0^1 \int_0^1 dz_2 dz_1 dx_1 \; x_1 z_2 (1-x_1)(1-z_1) a_m a_l a_s\\
& \times P_n^{(2)}(x_1, z_2(1-x_1)(1-z_1)) P_m^{(1)}(z_1) P_l^{(1)}(x_1) P_s^{(1)}(z_2)\\
& +\frac{1}{3}\lambda \int_0^1 \int_0^1 \int_0^1 dz_1 dz_2 dx_1 \; x_1 z_2 (1-x_1) a_m a_l a_s P_n^{(2)}(x_1, z_2(1-x_1)) P_m^{(1)}(z_1)\\
& \times P_l^{(1)}(x_1) P_s^{(1)}(z_2)
\end{aligned}
$$

$$
\begin{aligned}
&-2\lambda\int_0^1\int_0^1 dz_2dx_1\ z_2(1-x_1)(1-z_2)a_mP_n^{(2)}(x_1,z_2(1-x_1))P_m^{(1)}(z_2)\\
&-2\tilde{m}^2\int_0^1\int_0^1 dz_2dx_1\ x_1z_2(1-x_1)(1-z_2)a_mP_n^{(2)}(x_1,z_2(1-x_1))P_m^{(1)}(x_1)\\
&\times P_l^{(1)}(z_2)\\
&-2\int_0^1\int_0^1 dz_2dx_1\ z_2(1-x_1)^2(1-z_2)a_ma_lP_n^{(2)}(x_1,z_2(1-x_1))P_m^{(1)}(x_1)P_l^{(1)}(z_2)\\
&-4\lambda\int_0^1\int_0^1\int_0^1 dydz_2dx_1\ x_1z_2(1-x_1)^2(1-z_2)a_mc_lP_n^{(2)}(x_1,z_2(1-x_1))\\
&\times P_m^{(1)}(z_2)P_l^{(2)}(x_1,y(1-x_1))\\
&-\lambda\int_0^1\int_0^1\int_0^1 dydz_2dz_1\ z_1z_2(1-y)(1-(1-y)z_1)^2(1-z_2)a_ma_la_s\\
&\times P_n^{(2)}(z_1(1-y),z_2(1-(1-y)z_1))P_m^{(1)}(y)P_l^{(1)}(z_1)P_s^{(1)}(z_2)\\
&-\lambda\int_0^1\int_0^1\int_0^1 dz_1dz_2dx_1\ x_1z_2(1-x_1)(1-z_2)a_ma_la_sP_n^{(2)}(x_1,z_2(1-x_1))\\
&\times P_m^{(1)}(z_1)P_l^{(1)}(x_1)P_s^{(1)}(z_2)\\
&-2\tilde{m}^2\int_0^1\int_0^{1-x_1} dx_2dx_1\ x_1x_2(1-x_1-x_2)c_mP_n^{(2)}(x_1,x_2)P_m^{(2)}(x_1,x_2)\\
&-2\lambda\int_0^1\int_0^{1-x_1}\int_0^1 dydx_2dx_1\ x_1x_2(1-x_1-x_2)a_mc_lP_n^{(2)}(x_1,x_2)P_m^{(1)}(y)\\
&\times P_l^{(2)}(x_1,x_2)\\
&+\tilde{m}^2\int_0^1\int_0^1 dz_2dx_1\ x_1z_2(1-x_1)^2(1-z_2)a_ma_lP_n^{(2)}(x_1,z_2(1-x_1))P_m^{(1)}(x_1)\\
&\times P_l^{(1)}(z_2)\\
&+\lambda\int_0^1\int_0^1\int_0^1 dydz_2dx_1\ x_1z_2(1-x_1)^2(1-z_2)a_ma_la_sP_n^{(2)}(x_1,z_2(1-x_1))\\
&\times P_m^{(1)}(y)P_l^{(1)}(x_1)P_s^{(1)}(z_2).
\end{aligned}
$$

We want to re-write these equations as matrices multiplied by a coefficient vector. The coefficient matrices will be defined as

$$
\begin{aligned}
A &= \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, \qquad (3.57) \\
A2 &= \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \\
B &= \begin{pmatrix} a_1 a_1 \\ a_2 a_1 \\ \vdots \\ a_n a_n \end{pmatrix}, \\
B2 &= \begin{pmatrix} a_1 a_1 a_1 \\ a_2 a_1 a_1 \\ \vdots \\ a_n a_n a_n \end{pmatrix}, \\
Bmix21 &= \begin{pmatrix} a_1 c_1 \\ a_1 c_2 \\ \vdots \\ a_n c_n \end{pmatrix}.
\end{aligned}
$$

The names for the matrices are picked to help recall which diagram they represent. This makes the system become

$$
\begin{aligned}
\frac{M^2}{\mu^2} &= \tilde{m}^2 + \lambda \int_0^1 dy \; a_n P_n^{(1)}(y), \qquad (3.58) \\
0 &= \lambda PTloop + PTu.A + \lambda PTloopT.B/2 + \frac{\lambda}{2} PTTloop.B \\
&\quad + \tilde{m}^2 TP.A - \lambda TPT.B + 2\lambda PT2loop.A2
\end{aligned}
$$

$$
\begin{aligned}
0 \;=\; & 2\lambda P1T.A + 2\tilde{m}^2 PxT2.A2 + 4PuT2.A2 + PTTu.B \\
& + PTuT.B + \tilde{m}^2 PxTT.B + \lambda PTloopT2.Bmix21 \\
& + 2\lambda PTT2loop.Bmix21 + 2\lambda PT2loopT.Bmix21 \\
& + \lambda PT2Tloop.Bmix21 + \frac{\lambda}{3} PTTTloop.B2 + \frac{\lambda}{3} PTTloopT.B2 \\
& + \frac{\lambda}{3} PTloopTT.B2 - 2\lambda TP1.A - 2\tilde{m}^2 TPxT.B \\
& - 2PTTu.B - 4\lambda TPT2.Bmix21 - \lambda TPTTloop.B2 \\
& - \lambda TPTloopT.B2 - 2\tilde{m}^2 T2Psub.A2 - 2\lambda T2PT.Bmix21 \\
& + \tilde{m}^2 TTP.B + \lambda TTPT.B2.
\end{aligned}
$$

These are now ready to solve using a system solver function in Mathematica. The code is provided in Appendix A.

3.4 Results for LFCC Method

Solving the system of equations can be tricky. We want to make sure we are getting the solution we want. We want a physical solution. The most brute force way of making sure we get our solution is to find them all and plot the coefficients. Looking at our system, as λ goes to zero $\frac{M^2}{\mu^2}$ approaches $\tilde{m}$. This implies a_n goes to zero for all n. a_0 is plotted in Figure 3.35 for all solutions as a function of λ. The plot is zoomed in on the origin to help find our physical solution. The points corresponding to our solution are red. The red points stop short due for two reasons. Around $\lambda = 0.7$, the path abruptly changes direction and intersects another path. This could be caused by the mass ratio becoming negative which would also be not physical. This means we only care about the points leading up to $\lambda = 0.7$. There are other points along the desired path. Those points are essentially the same solution.

The Solve[] command in Mathematica returns solutions with imaginary and real parts. Along the path of the physical solution, the imaginary portions are different but significantly small enough to be treated as zero while the real parts are the same.

The next step is to plot all the solutions the code gives in Figure 3.36. The solution that corresponds to the red points in Figure 3.35 is also red in Figure 3.36.

Now that we have an idea of what the physical solution looks like, we can use FindRoot[] in the code to find the solutions we want. FindRoot[] requires an initial guess. Zero for all coefficients is used as the initial guess for the first λ value for the first order system of equations. Every λ value from then on has their initial guess be the previous solution. We can control $\tilde{m}$ and the order of basis polynomials. We are looking for mass ratio convergence as we increase the basis order where k is the basis polynomial order.

Figure 3.37 shows convergence happens quickly with $\tilde{m} = 1$ for first order. Second order caused some troubles with convergence. Convergence happened slowly in Figure 3.38 and running the code for order 8 and higher took too much time. The graph does suggest convergence before the order goes above order 8. We put the highest order results together in Figure 3.39 showing the addition of second order does change the result. This same analysis was made for $\tilde{m} = 0.1$ and $\tilde{m} = 10$. The lower $\tilde{m}$ value did not change much by adding second order equations. The higher $\tilde{m}$ value caused troubles. The code returned non-solutions before the mass ratio became negative. This gave a very small window to compare changes between first and second order. These results are shown in Figures 3.40-45.

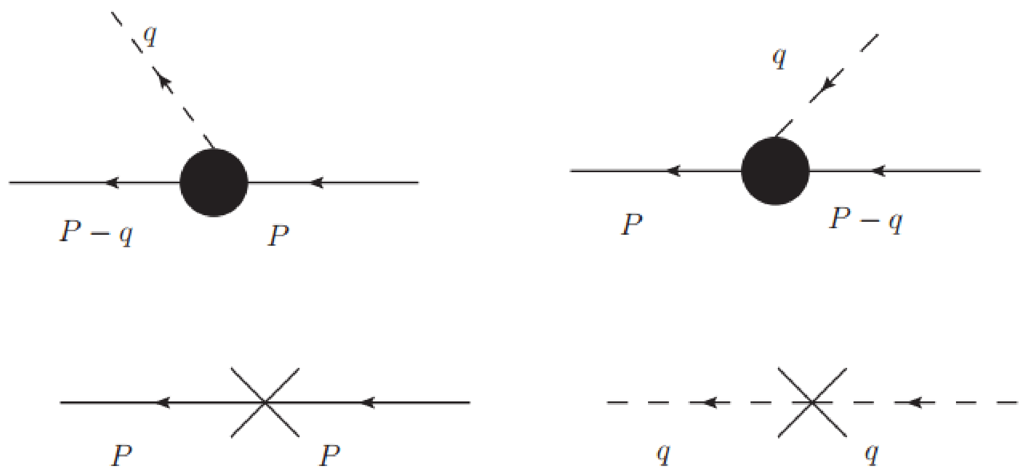

Figure 3.1: Diagram representations of $\mathcal{P}^-$ with the charged particle coming in from the right with total momentum P. The black circle is the interaction part of the Hamiltonian, $\mathcal{P}^-$; the cross represents the free part. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains or loses momentum along the path.

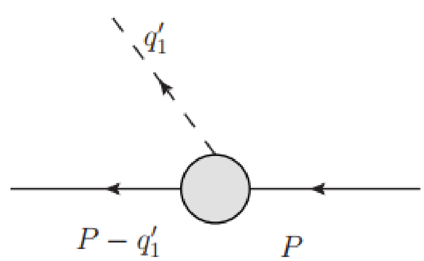

Figure 3.2: Diagram representation of T_1 with the charged particle coming in from the right with total momentum P. The gray circle is the T operator. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

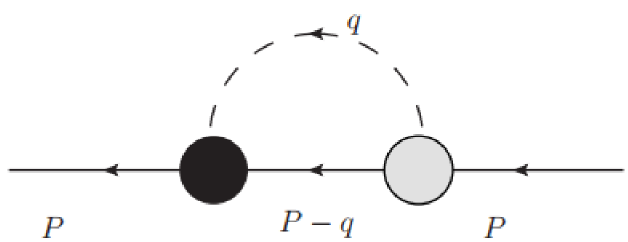

Figure 3.3: One of the diagram representations of $\mathcal{P}^- T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

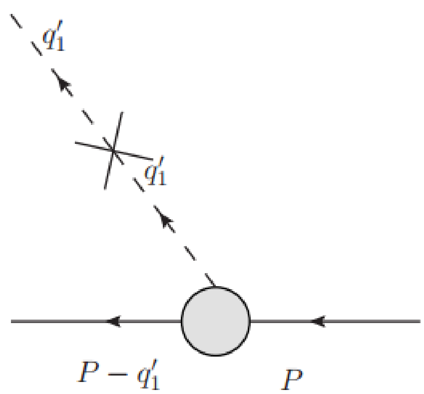

Figure 3.4: One of the diagram representations of $\mathcal{P}^- T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

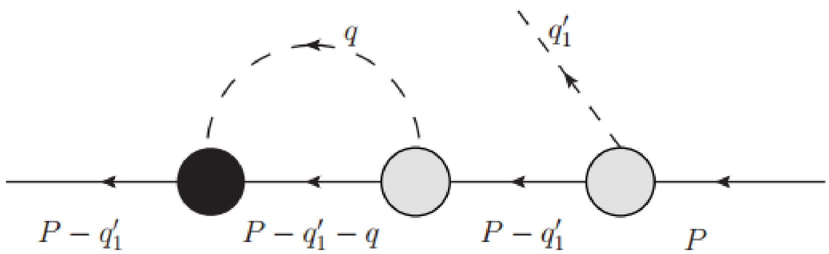

Figure 3.5: One of the diagram representations of $\mathcal{P}^- T_1^2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

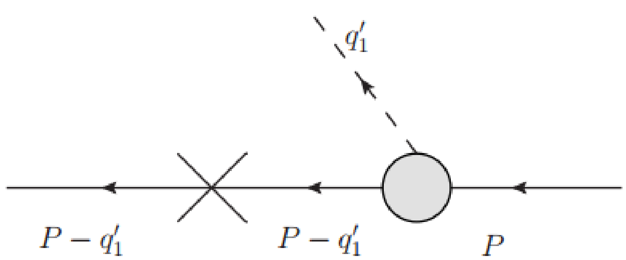

Figure 3.6: One of the diagram representations of $\mathcal{P}^- T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

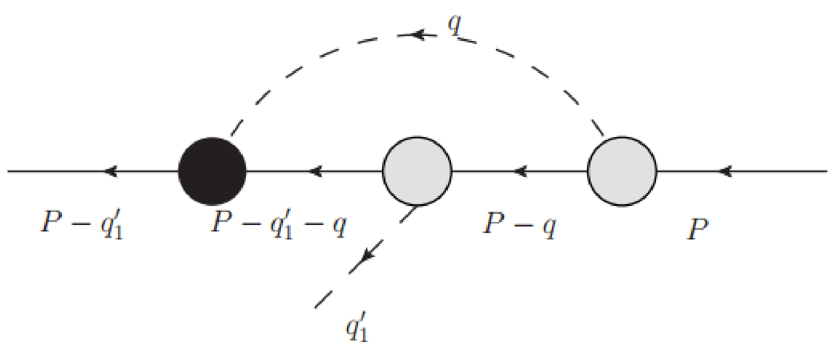

Figure 3.7: One of the diagram representations of $\mathcal{P}^- T_1^2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

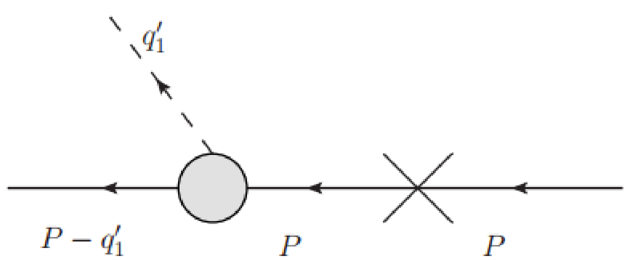

Figure 3.8: One of the diagram representations of $T_1 \mathcal{P}^-$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

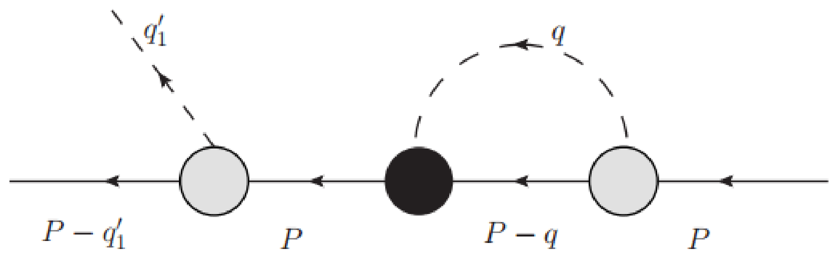

Figure 3.9: One of the diagram representations of $T_1\mathcal{P}^-T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

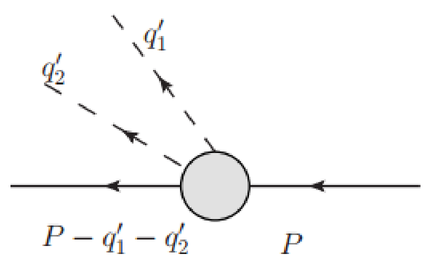

Figure 3.10: Diagram representation of T_2 with the charged particle coming in from the right with total momentum P. The gray circle is the T operator. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

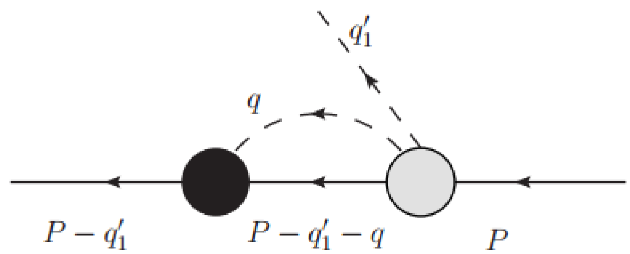

Figure 3.11: One of the diagram representations of $\mathcal{P}^{-}T_2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

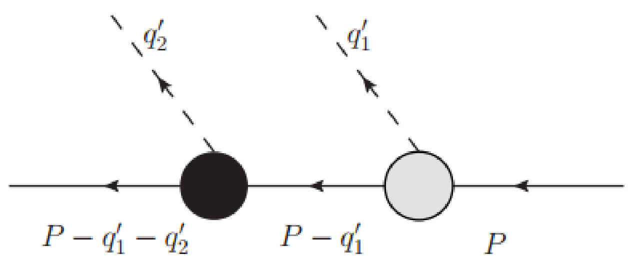

Figure 3.12: One of the diagram representations of $\mathcal{P}^{-}T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

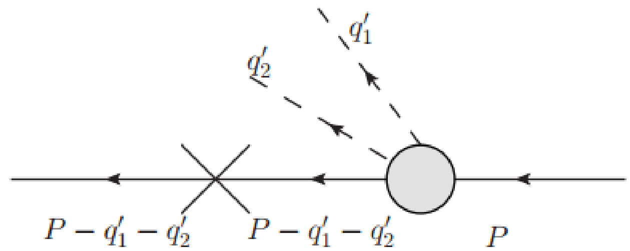

Figure 3.13: One of the diagram representations of $\mathcal{P}^{-}T_2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

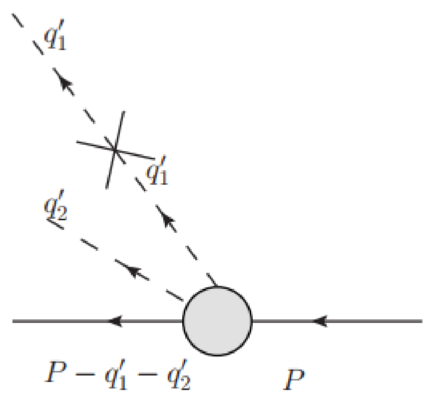

Figure 3.14: One of the diagram representations of $\mathcal{P}^{-}T_2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

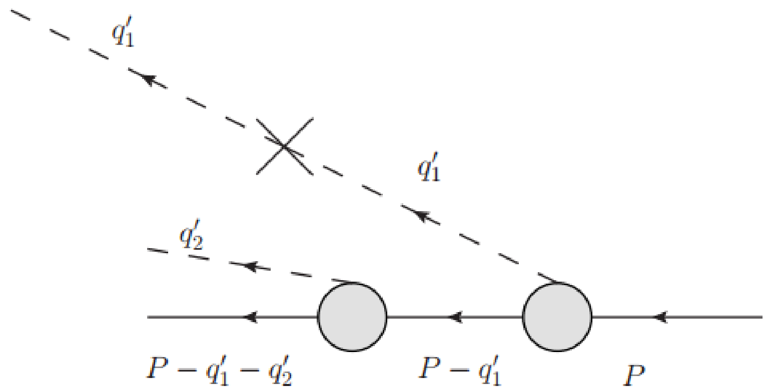

Figure 3.15: One of the diagram representations of $\mathcal{P}^- T_1^2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

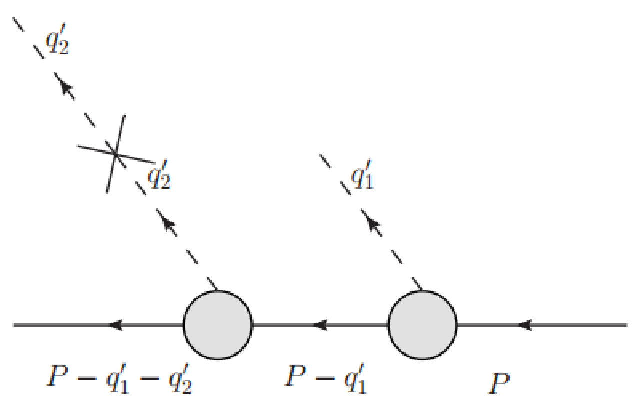

Figure 3.16: One of the diagram representations of $\mathcal{P}^- T_1^2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

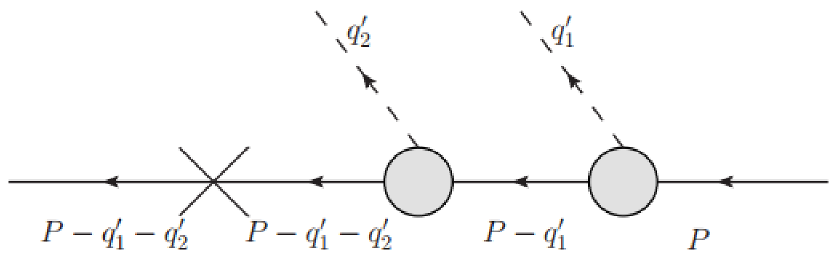

Figure 3.17: One of the diagram representations of $\mathcal{P}^- T_1^2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

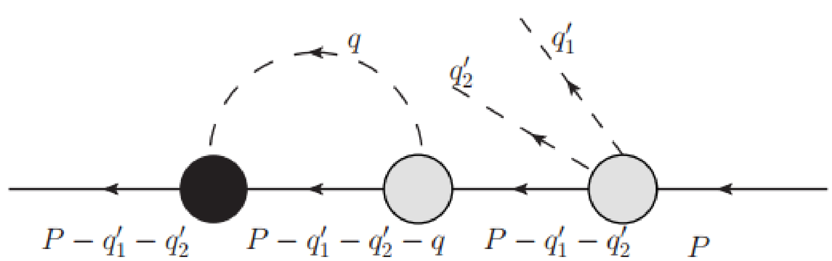

Figure 3.18: One of the diagram representations of $\mathcal{P}^- T_1 T_2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

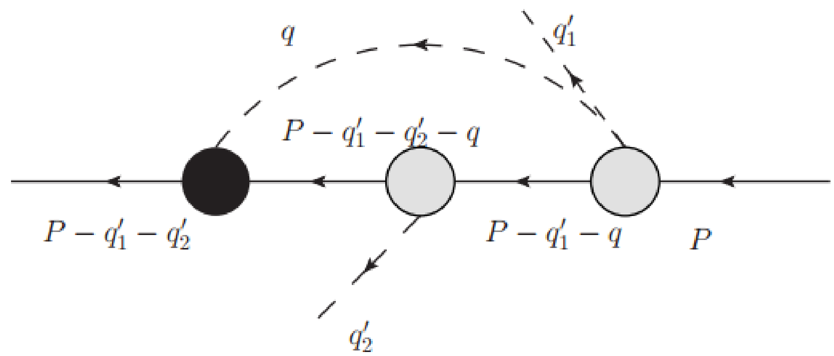

Figure 3.19: One of the diagram representations of $\mathcal{P}^{-}T_1T_2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

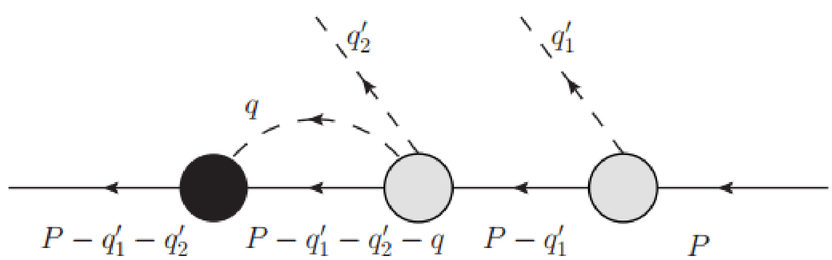

Figure 3.20: One of the diagram representations of $\mathcal{P}^{-}T_2T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

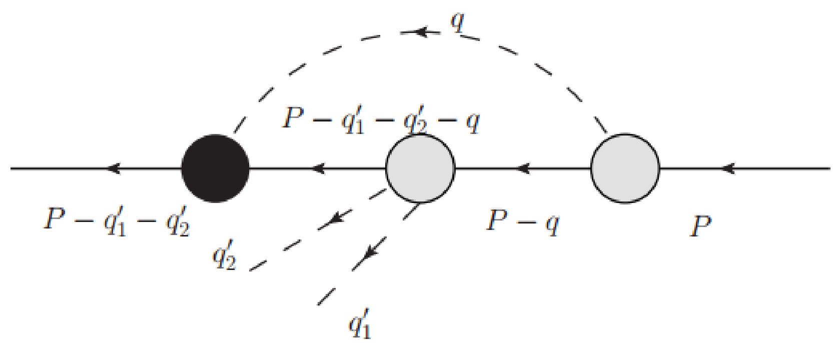

Figure 3.21: One of the diagram representations of $\mathcal{P}^{-}T_2T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

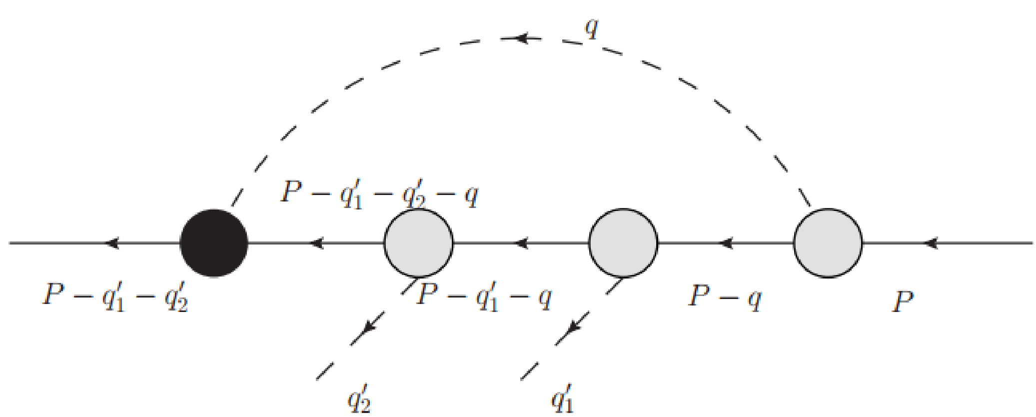

Figure 3.22: One of the diagram representations of $\mathcal{P}^{-}T_1^3$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

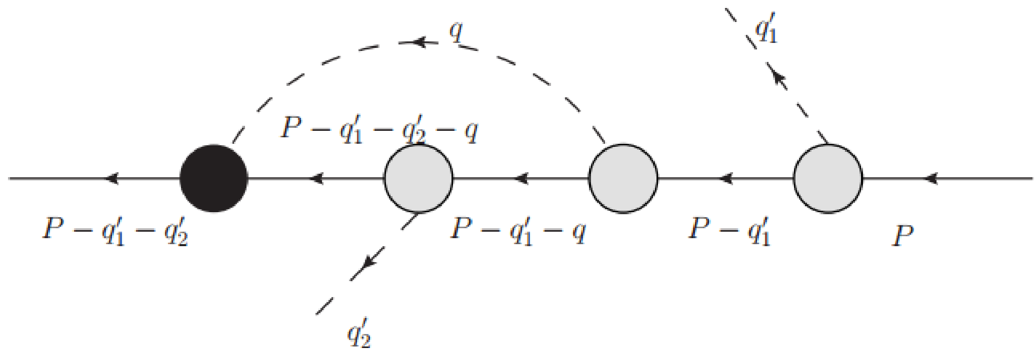

Figure 3.23: One of the diagram representations of $\mathcal{P}^{-}T_1^3$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

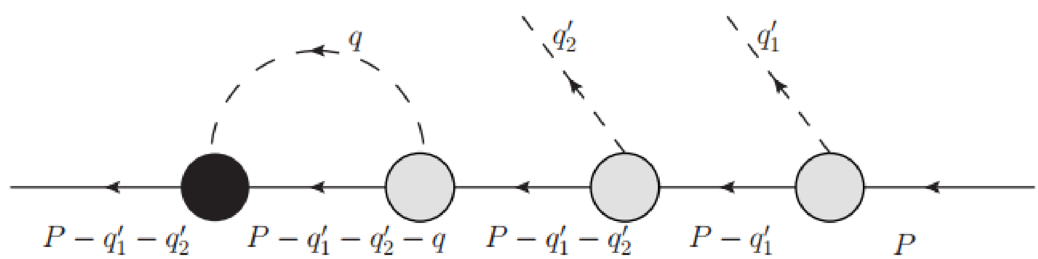

Figure 3.24: One of the diagram representations of $\mathcal{P}^{-}T_1^3$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

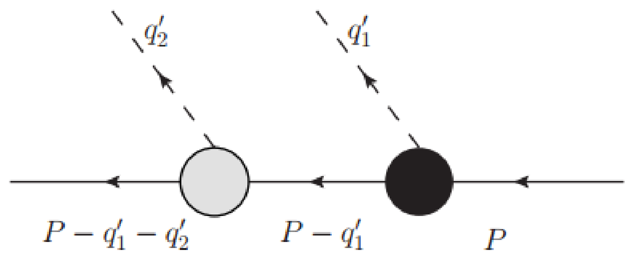

Figure 3.25: One of the diagram representations of $T_1\mathcal{P}^-$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

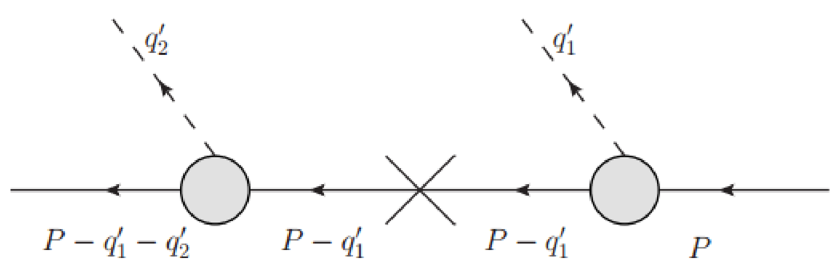

Figure 3.26: One of the diagram representations of $T_1\mathcal{P}^-T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

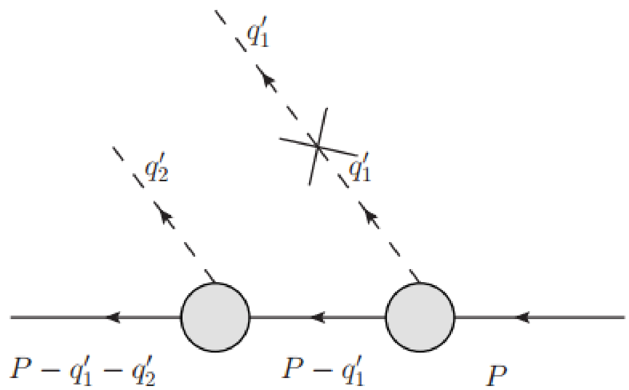

Figure 3.27: One of the diagram representations of $T_1\mathcal{P}^{-}T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

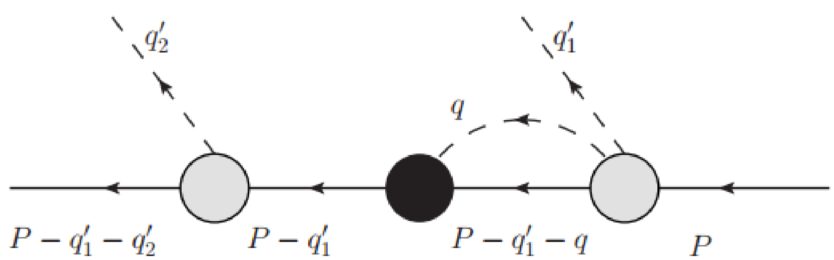

Figure 3.28: Diagram representation of $T_1\mathcal{P}^{-}T_2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^{-}$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

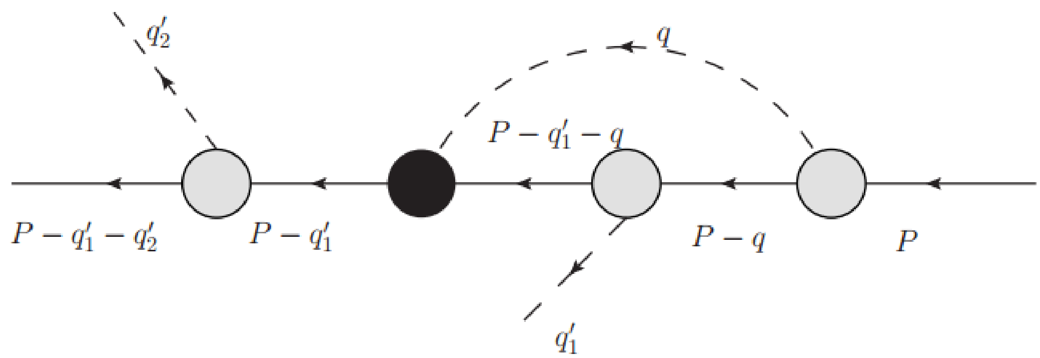

Figure 3.29: One of the diagram representations of $T_1\mathcal{P}^-T_1^2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

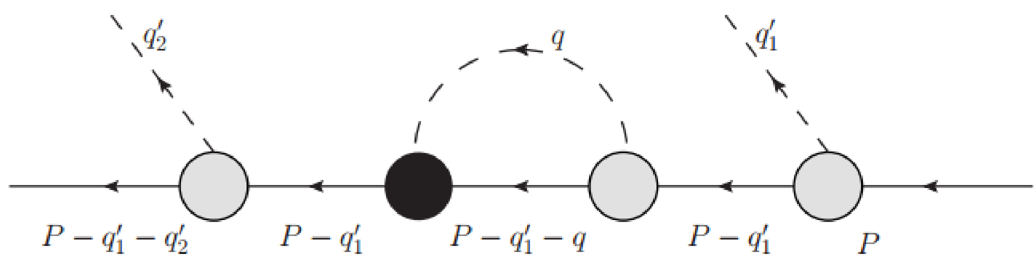

Figure 3.30: One of the diagram representations of $T_1\mathcal{P}^-T_1^2$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and black circles Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

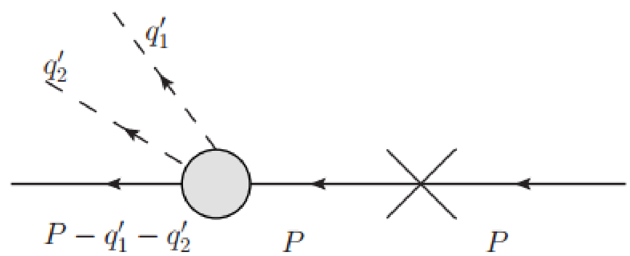

Figure 3.31: Diagram representation of $T_2\mathcal{P}^-$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

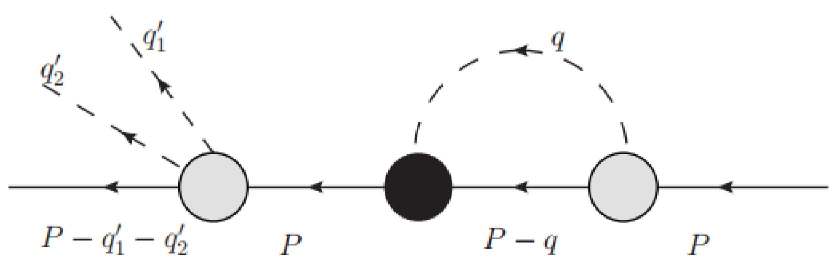

Figure 3.32: Diagram representation of $T_2\mathcal{P}^-T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

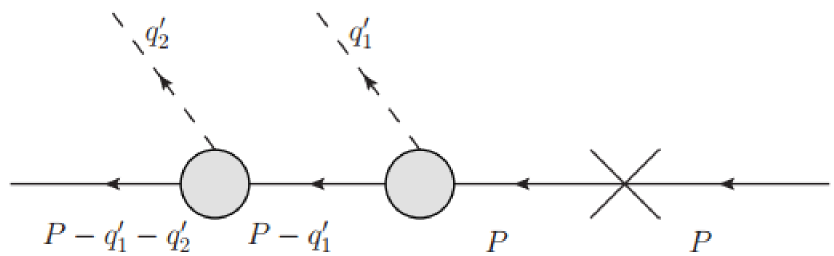

Figure 3.33: One of the diagram representations of $T_1^2\mathcal{P}^-$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and the cross being the interaction part of the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that loses momentum along the path.

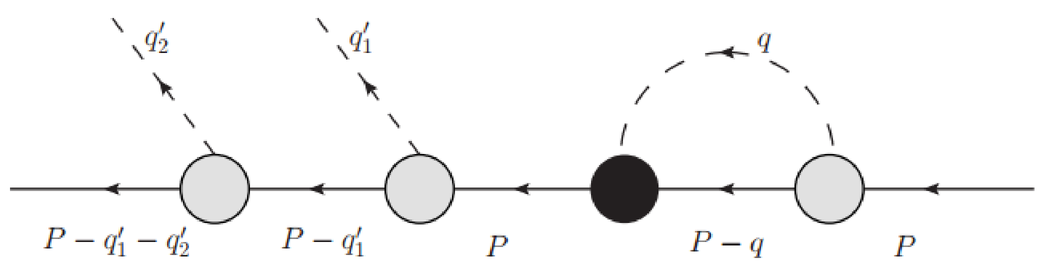

Figure 3.34: Diagram representation of $T_1^2\mathcal{P}^-T_1$ with the charged particle coming in from the right with total momentum P. The gray circles being the T operators and black circle being the Hamiltonian, $\mathcal{P}^-$. The dotted lines are the neutral particles with momentum q and the solid line is the charged particle that gains and loses momentum along the path.

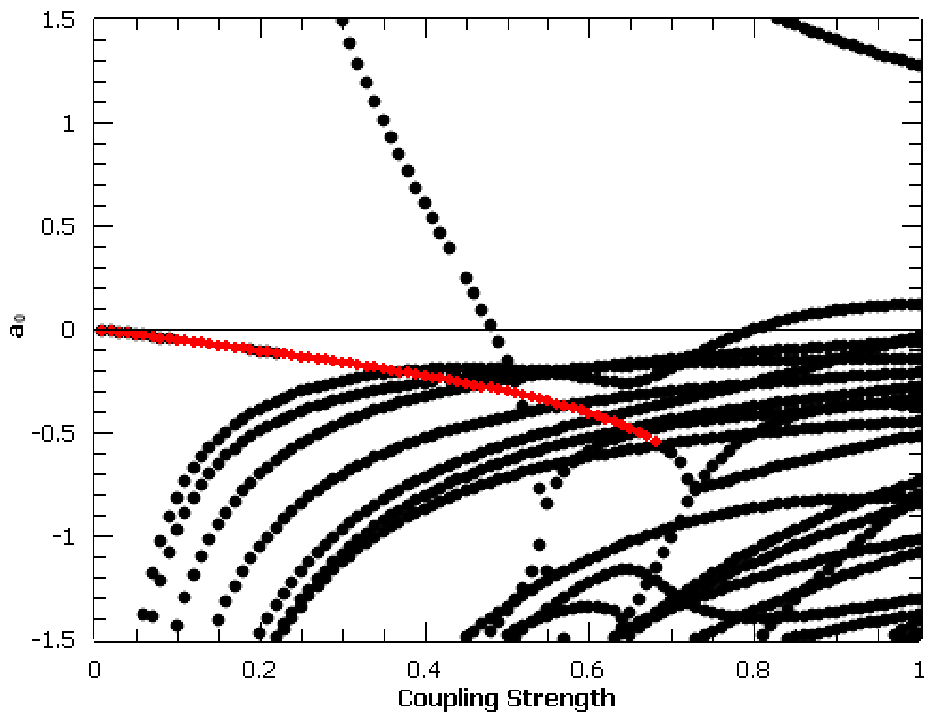

Figure 3.35: Plot of a_0 as a function of λ. a_0 was graphed for every solution but not all are shown due to the limited vertical range. The red points correspond to a physical solution.

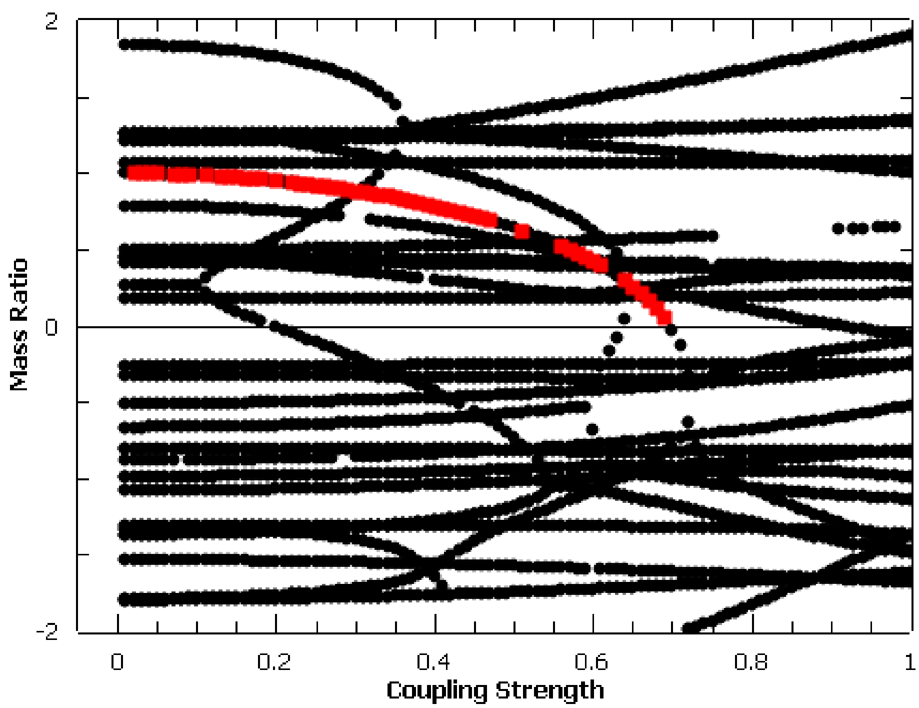

Figure 3.36: Plot of multiple solutions to find physical solution. The red solution corresponds to the only physical solution for this set up.

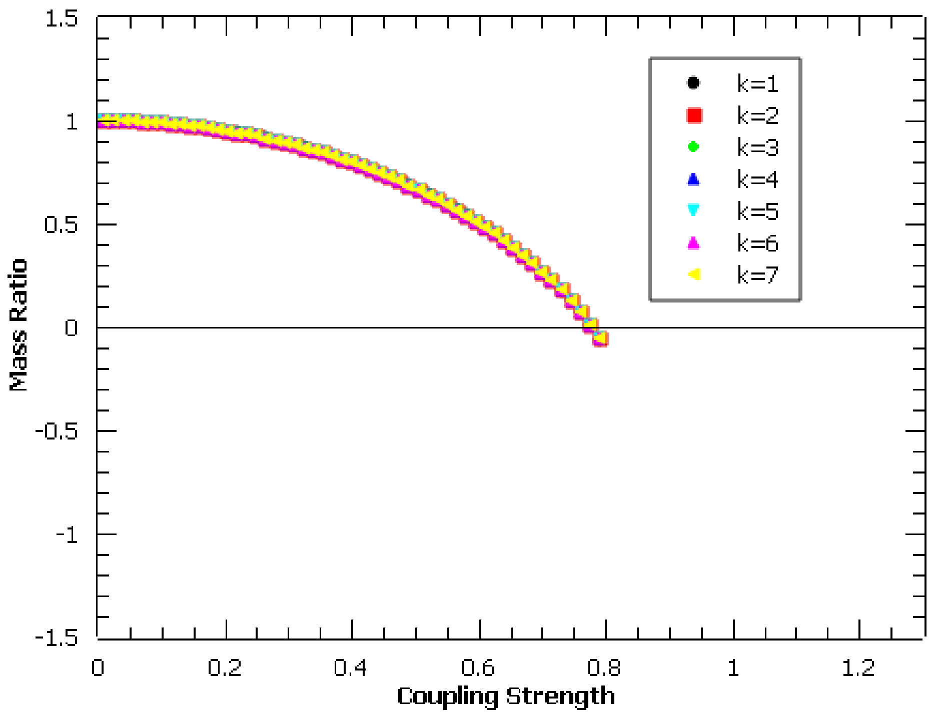

Figure 3.37: Plot of first order solutions using LFCC method for $\tilde{m} = 1$. The solutions converged very quickly. k is the basis polynomial order

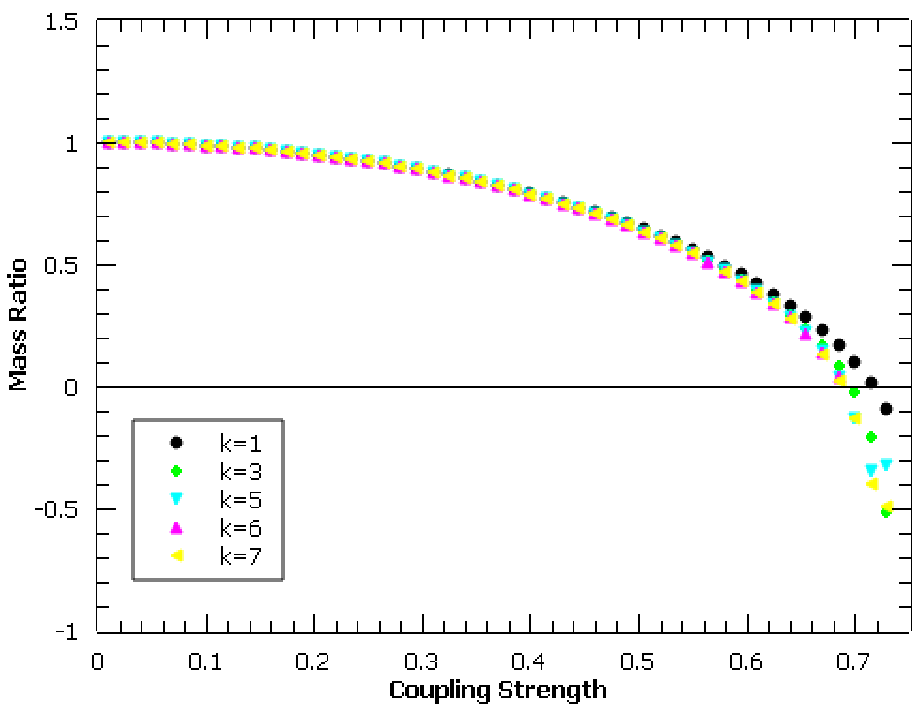

Figure 3.38: Plot of second order solutions using LFCC method for $\tilde{m} = 1$. k is the basis polynomial order. The solutions converged very slowly.

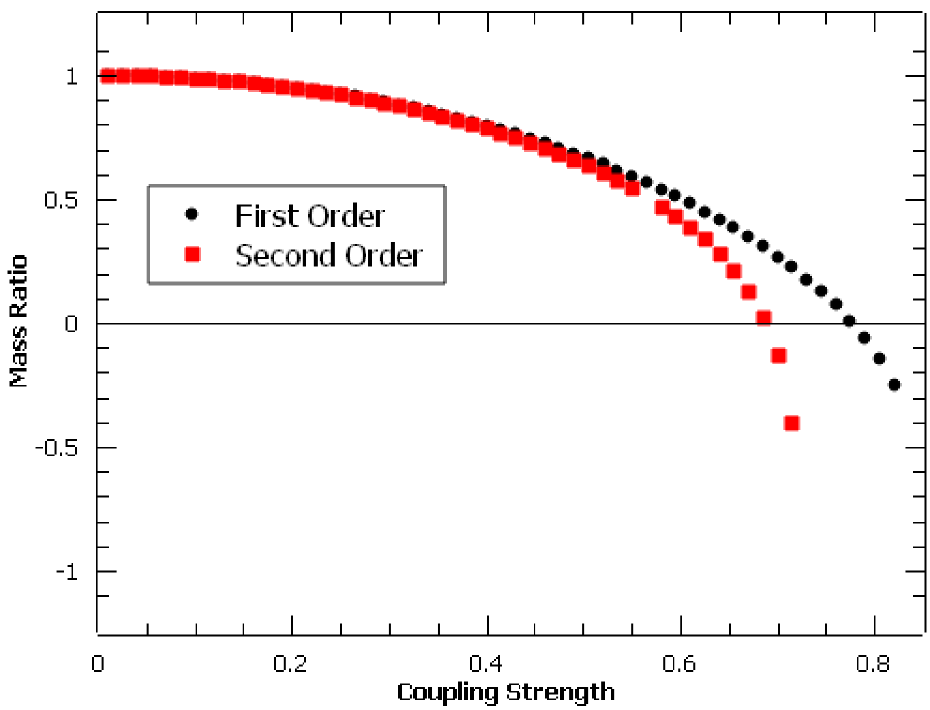

Figure 3.39: Plot of first and second order solutions using LFCC method for $\tilde{m} = 1$. The graph shows that adding second order equations changes the overall mass ratio.

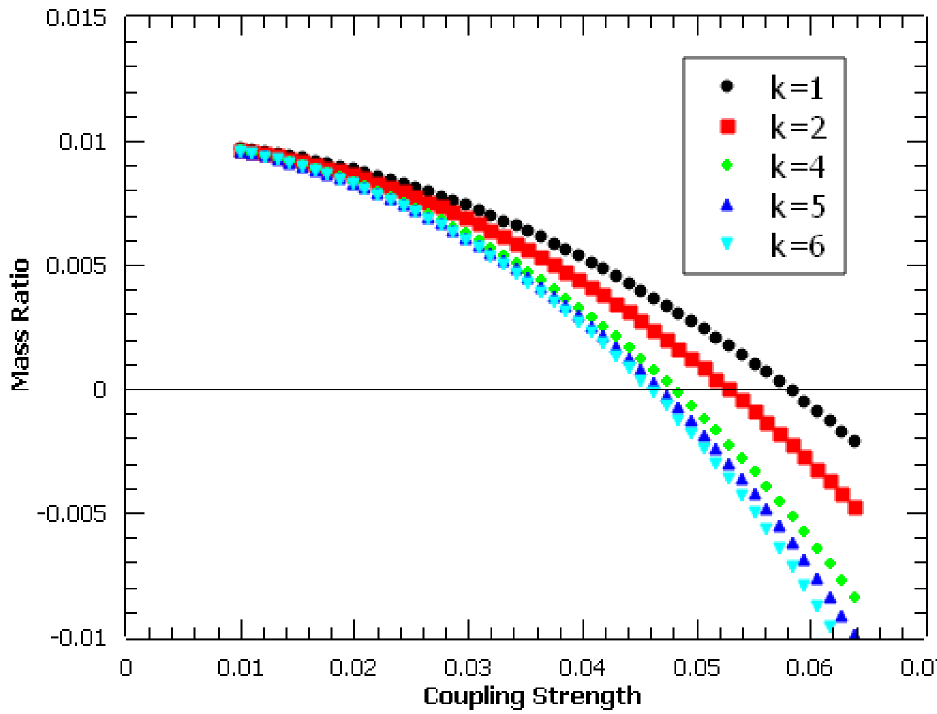

Figure 3.40: Plot of first order solutions using LFCC method for $\tilde{m} = 0.1$. k is the basis polynomial order. The solutions converged very slowly and are likely not fully converged.

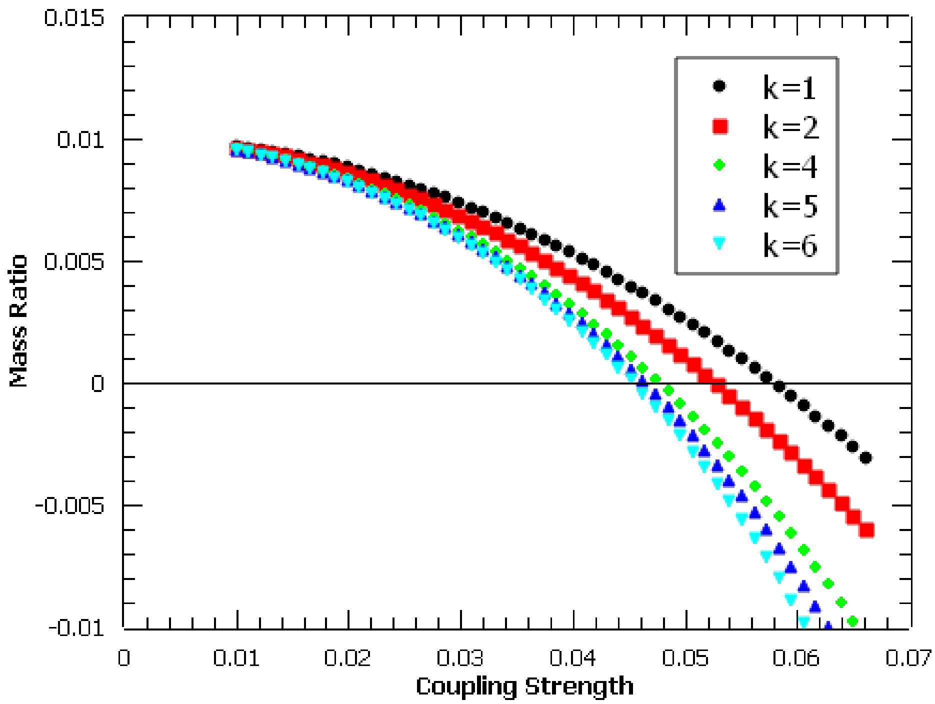

Figure 3.41: Plot of second order solutions using LFCC method for $\tilde{m} = 0.1$. k is the basis polynomial order. The solutions converged very slowly and are likely not fully converged.

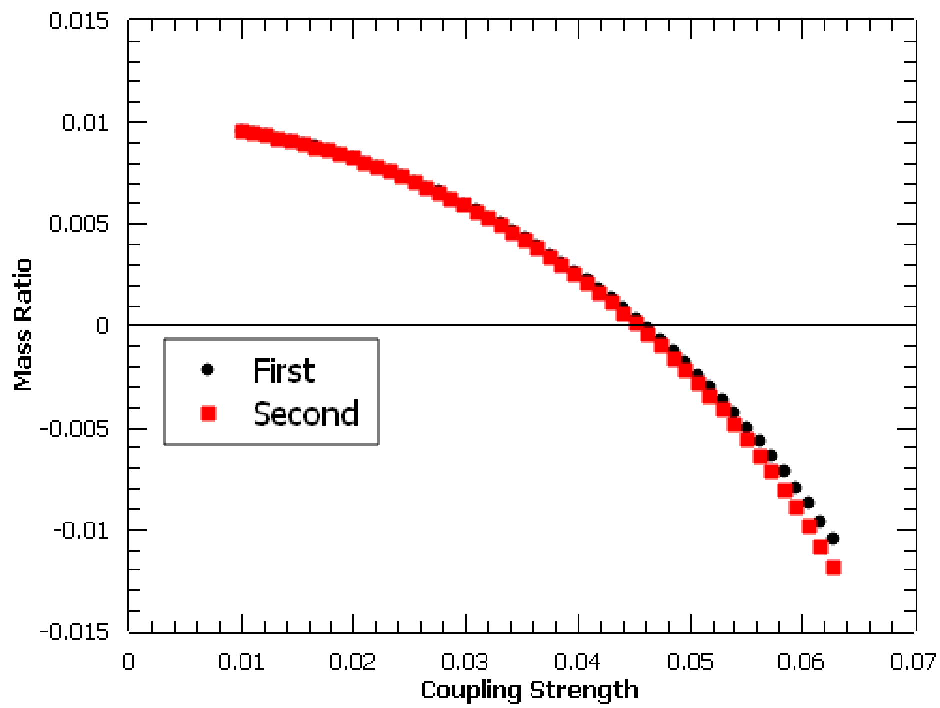

Figure 3.42: Plot of first and second order solutions using LFCC method for $\tilde{m} = 0.1$. The graph shows that adding the second order equations did not change the overall mass ratio a noticeable amount.

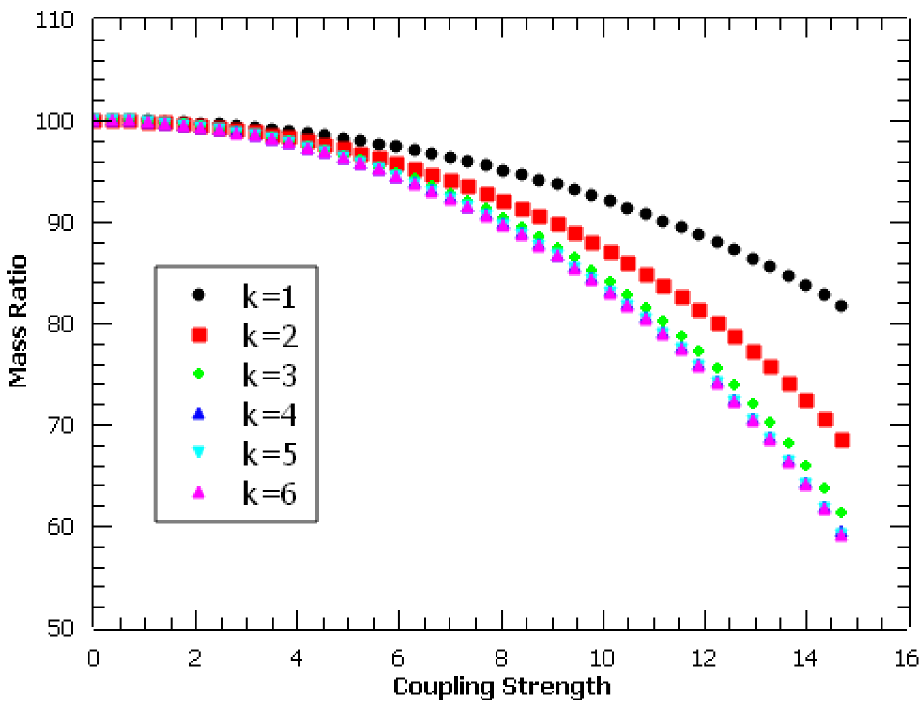

Figure 3.43: Plot of first order solutions using LFCC method for $\tilde{m} = 10$. k is the basis polynomial order.

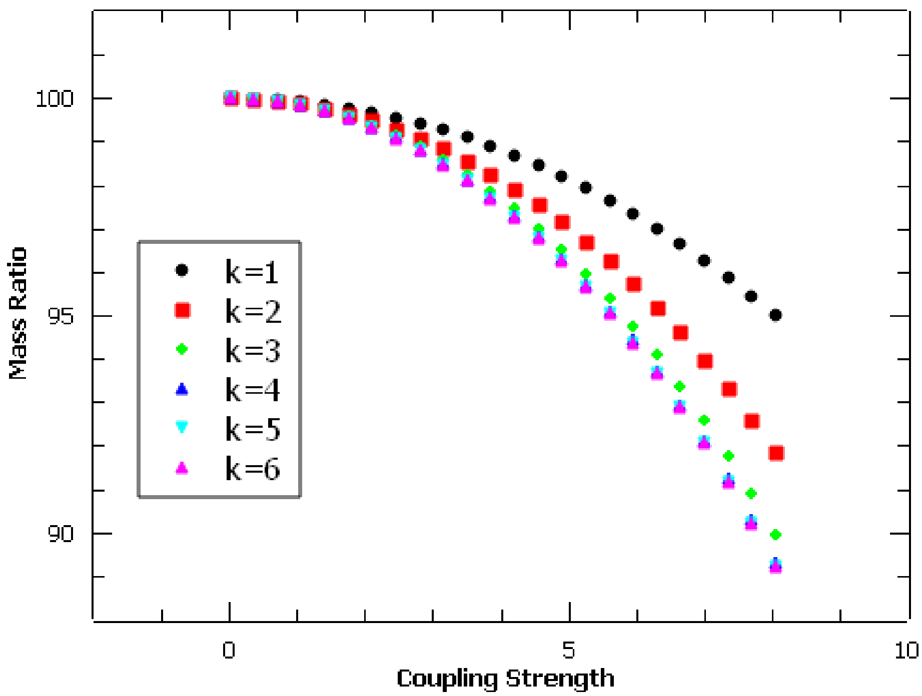

Figure 3.44: Plot of second order solutions using LFCC method for $\tilde{m} = 10$. k is the basis polynomial order.

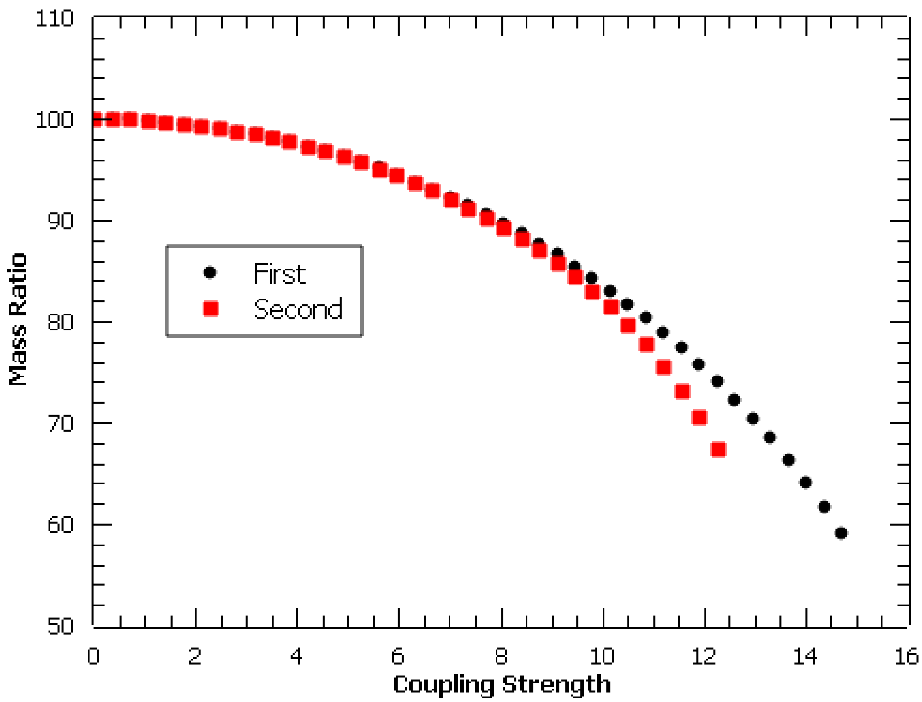

Figure 3.45: Plot of first and second order solutions using LFCC method for $\tilde{m} = 10$. The graph shows that adding the second order equations changes the overall mass ratio but due to limits in the code could not be explored fully.

Chapter 4

Comparison and Analysis

This chapter will compare the converged result for both the Fock state expansion and the LFCC method for each $\tilde{m}$. For $\tilde{m} = 1$, LFCC second order approximation and Fock state expansion result in very similar results shown in Figure 4.1. Both methods had different basis polynomials for single variable basis polynomials. The results may even be the same due to discrepancies in the code. This suggests both results being close to an exact solution. $\tilde{m} = 0.1$ has first and second order approximations both being similar to the Fock state expansion shown in Figure 4.2 but with slight differences making it difficult to tell. There is no evidence of where the exact result should be in this research. Higher values of $\tilde{m}$ caused problems throughout the research. Even with these troubles, Figure 4.3 shows a very similar result to Figure 4.1. This leads us to believe the results are not as bad as originally thought.

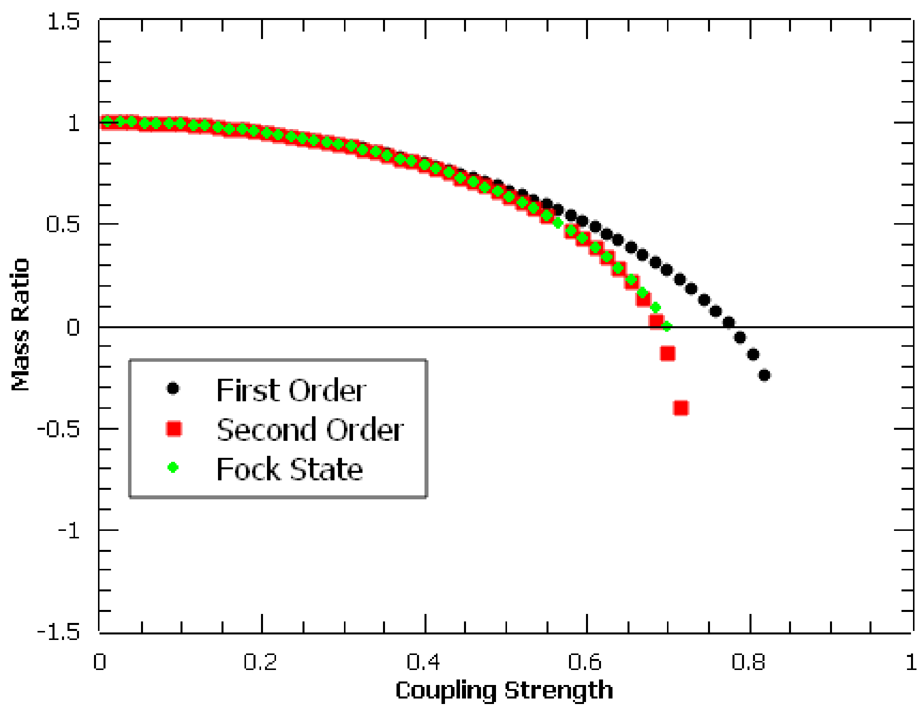

Figure 4.1: Convergence comparison for $\tilde{m} = 1$ between Fock state expansion and LFCC. The second order approximation and the Fock state expansion give a very similar result.

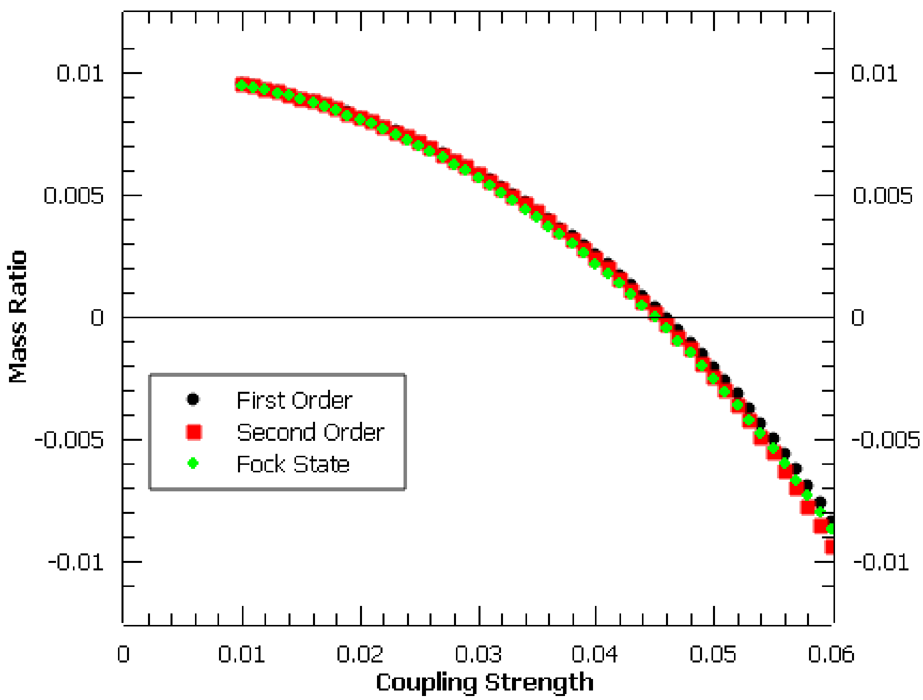

Figure 4.2: Convergence comparison for $\tilde{m} = 0.1$ between Fock state expansion and LFCC. The first and second order approximations give a very similar result. The Fock state expansion shows they are close to convergence.

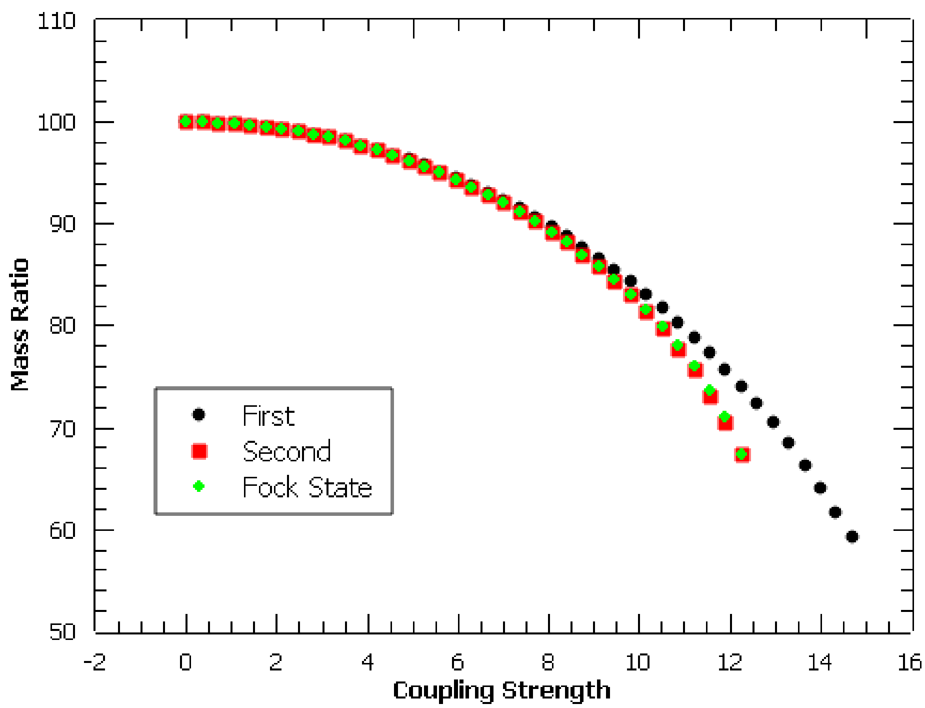

Figure 4.3: Convergence comparison for $\tilde{m} = 10$ between Fock state expansion and LFCC. The second order approximation and the Fock state expansion give a very similar result.

Chapter 5

Summary

We studied convergence using light-front coupled-cluster (LFCC) method in scalar Yukawa theory. First, we laid some ground work by using a Fock state expansion to determine mass eigenstates. This required solution of a generalized eigenvalue problem, obtained with symmetric basis polynomials. Then the LFCC method was used to first order to define a system of equations for comparison. Later, a second order approximation was used to study convergence of the LFCC method.

The Fock state expansion first required light-front coordiantes, defined in section 2.1, to re-write momentum in terms of momentum fractions. This made the resulting wave function invariant under a Lorentz transformation. This lead to Eq. 2.7. The system was assumed to be limited to one dimension. The Fock-state wave function $\psi_n(x_i)$ was expanded in terms of symmetric polynomials defined in section 2.3. These approximations lead to the generalized eigenvalue problem Eq. 2.15. Many variables needed to be tested for convergence including $\tilde{m}$ defined as the charged particles mass divided by the neutral particles, the order of the basis polynomials for $\psi_i(x_1)$, and the number of sectors allowed. Results showed small values of $\tilde{m}$ mostly depended on the one-particle sector and converged very quickly as a result. Larger values of $\tilde{m}$ did the opposite. They converged very slowly and depended on higher particle sectors. The code designed for this research had limitations that made it difficult to draw any conclusions for $\tilde{m} = 10$. The MatLab code is listed in the Appendix.

The LFCC method takes advantage of re-writing the momentum state using an operator defined in Eq. 3.3. The approximation here is then made to this operator.

First order approximation only allows one particle to be created at a time and results in a system of equations defined by Figures 3.1-9 with vertex rules discussed in section 3.2. We desired these equations to match the Fock state expansions from Chap. 2. This caused a re-write of t_n defined by Eq. 3.12. We then needed to write $\tilde{t}_n$ in terms of basis polynomials defined by Jacobi polynomials in Eq. 3.17-18 to have specific orthogonality conditions defined in Eq. 3.23.

The system of equations were simplified using basic substitutions and integral order changes. This allowed the equations to be defined by matrices acted on vectors made up of the coefficients from the basis polynomials. This system was then solved using a basic system solver in Mathematica. The same procedure was used for the second order approximation but now one and two particles could be created. This added another equation to our system and many more terms. Figure 3.35 showed us how to find our physical solution by realizing the coefficients had to go to zero as our coupling strength went to zero. This allowed us to determine which solution we wanted from all the solutions showed in Figure 3.36. The mass ratio given through first order and second order did not change much by increasing basis polynomial order. This made looking for convergence rather simple. Adding second order equations did change the results from the first order approximation for larger $\tilde{m}$ values. The Mathematica code is listed in the appendix.

Comparing first and second order approximations with the Fock state expansion showed the second order and Fock state expansions were very similar. This indicates that the LFCC method converges quickly, at least for the quenched scalar Yukawa model, with a second-order approximation sufficient to match the full solution for all mass values considered. The LFCC method does this with only two functions $t_1(x)$ and $t_2(x_1, x_2)$ at its disposal, as opposed to the multiple wave functions $\psi_n(x_1, \ldots, x_n)$ of the Fock-state expansion.

This work can be extended by defining the T operator in different ways. One such way would be to allow the operator to act on the neutral particles as well. This would not change any physics and would allow comparisons to be made to this work. Further research can be done by adding another charged particle as well. This would change the Fock state expansion as well. This work showed the validity of the LFCC method and showed it may be used as an alternate to other methods.

Bibliography

[1] S. J. Brodsky, H-C. Pauli, S. Pinsky Quantum Chromodynamics and Other Field Theories on the Light Cone, Phys. Rep. **301**, 299-486 (1998)

[2] Y. Li, V. A. Karmanov, P. Maris, and J. P. Vary, Ab Initio Approach to the Non-Perturbative Scalar Yukawa Model, Phys. Lett. B **748**, 278 (2015)

[3] D. J. Griffiths, Introduction to Quantum Mechanics. 2nd Ed. Prentice Hall, (2005)

[4] M.E. Peskin, and D.V. Schroeder, An Introduction To Quantum Field Theory. Frontiers in Physics, (1995)

[5] V. A. Karmanov, Y. Li, A. V. Smirnov and J. P. Vary, Nonperturbative solution of scalar Yukawa model in two- and three-body Fock space truncations, Phys. Rev. D **94**, 096008 (2016)

[6] J. R. Hiller, Nonperturbative light-front Hamiltonian methods, Prog. Part. Nucl. Phys. **90**, 75 (2016)

[7] B. Elliott, S. S. Chabysheva and J. R. Hiller, Application of the light-front coupled-cluster method to ϕ^4 theory in two dimensions, Phys. Rev. D **90**, 056003 (2014)

[8] J. B. Swenson and J. R. Hiller, Numerical signatures of vacuum instability in a one-dimensional Wick-Cutkosky model on the light cone, Phys. Rev. D **48**, 1774 (1993).

[9] Dirac PAM (1949) Forms of Relativistic Dynamics, Rev. Mod. Phys. **21** 392-399

[10] Iyanaga, S. and Kawada, Y. (Eds.). Jacobi Polynomials, Appendix A, Table 20.V in Encyclopedic Dictionary of Mathematics. Cambridge, MA: MIT Press, p. 1480, (1980)

Appendix A

Code

A.1 Fock State Expansion Using MatLab

```
(* ::Package:: *)

% FockGenMultNorm2.m is used to study Yukawa Theory by creating
% the required matrices and solving the generalized eigenvalue
% problem M*D*Psi = C*Psi. This code creates the Matrices C
% and D and varies the coupling strength to higher states to
% see the effects on M, the eigenvalue and total mass to
% neutral mass ratio. The matrices are made by looking at
% the equations for Psi_n that are generated from the Fock
% expansion of the eigenstates.
%
% Degree of approximation for each sector in order starting
% with Psi_ 0 and number of particles
N = [0 6 6 5 5 3 2 2 2 2 2];
n = length(N)-1;
mt = 4;
dStor = cell(n, 1);

% Initializing matrices
```

```
D = [];
DiagC = [];
Result = [];
Result1 = [];

% Max value for coupling strength and number of increments
% to loop over
maxg = 10;
inc = 100;

% Loop that creates the matrix D in M*D*Psi = C*Psi where
% M is theeigenvalue
for s = 0:n
    d = BaseNorm(N(s+1),s);
    dStor{s+1} = d;
    D = blkdiag(D,d);
end

% C matrix is made up of multiple matrix cnn' where n
% and n' refer to Psi_n' in the Psi_n equation

% Uses the function OnD() to make the matrix for Psi_ 0
% with Psi_ 0
c00 = OnDNorm(N(1),0,mt);

% Defines the matrix associated with both one particle
% states
c11 = OnDNorm(N(2),1,mt);

% Stores both matrices in a block diagonal matrix. This
% helps with the final C matrix.
DiagC = blkdiag(DiagC,c00,c11);
```

```
% Defines the off diagonal matrices for C
c10 = OffDNorm(N(1),N(2),1);
c01 = transpose(c10);

% Puts the previous matrices together. If we were looking
% at up to only the 1 particle state, C would be completed.
C1 = [c00 c01];
C2 = [c10 c11];
C = [C1;C2];

% This helps to know what index the rest of the component
% matrices need to be in the final C matrix
c10s = size(c10);
c11s = size(c11);
tots = c10s(2);

% Simple if statement. If n=1 then the C matrix is
% completed and nothing more needs to be done to it.
%Any greater n value requires more component matrices.
if n == 1
else
    % Loop that makes the remaining component matrices c.
    %Everything is stored temporarily to save memory. This
    % follows the same pattern above.
    for s = 2:n
        % The block diagonal matrices of the form cnn are
        % made and stored.
        tempD = OnDNorm(N(s+1),s,mt);
        DiagC = blkdiag(DiagC,tempD);

        % The two off diagonal matrices are made and have
```

```
        % the form cnn-1
        tempL = OffDNorm(N(s),N(s+1),s);
        tempU = transpose(tempL);

        % Sizes are stored for indexing
        tempLs = size(tempL);

        % Matrices are combined to form C
        C3 = zeros(tempLs(1),tots);
        C4 = [C3 tempL tempD];
        C5 = [transpose(C3); tempU];
        C = [C C5;C4];

        tots = tots + tempLs(2);
    end
end

% Loop that runs over the desired values for the coupling
% strength
for i = 0:inc
    % Defines how g should increment.
    g = 0.01 + maxg/inc*i;

    % The matrix C was constucted in a way to not include
    % the coupling strength g in order to reduce
    % computation time. The only values that need g are
    % the other diagonal block matrices. The easiest way
    % to accomplish including g is to subtract the block
    % diagonal matrices, multiple the entire temporary C
    % by g, then add back in the blockdiagonal elements.
    tempC = C - DiagC;
    NewC = g*tempC;
```

```
    tempC2 = NewC + DiagC;

    % Obtains the eigenvalues, M, from the generalized
    % eigenvalue problem M*D*Psi = C*Psi and stores
    % them along with g.
    [Vector,Value] = eig(tempC2,D);
    Values = size(Value);

    for s = 1:Values(1);
        Value1(s) = Value(s,s);
    end

    [minV,minP] = min(Value1);
    Result = [Result;g minV];

    Vector1 = Vector(:,minP);
    Result1 = [Result1; RelP(g,dStor,Vector1,N)];
end

[VectorD,ValueD] = eig(D);
for s = 1:Values(1);
    Value1D(s) = ValueD(s,s);
end

% An if statement to check for negative eigenvalues.
%Physically the eigenvalues should not be negative.
%This could be caused by roundoff error
if min(Value1D) < 0
    disp('Warning: Overlap has negative eigenvalue')
end

% Prints a graph of the eigenvalues M as a function of g.
```

```
scatter(Result(:,1),Result(:,2));

function [Result] = BaseNorm(deg,n)
%Inputs: deg, The max polynomial degree, and n, number
%of particles.
%
%Outputs: Result, a square matrix.
%
%This function creates the dnn matrix for the final D
%matrix. These matrices are block diagonal in D. This
%function makes matrices representing
%Integrate[x1*x2*...*xn*(1-x1-x2-...-xn)*QNk^(n)*QNk^(n)
%dx1dx2...dxn.

%Calls Norm() to help normalize the basis polynomials
Nor = Norm(deg,n);

%If there are no neutral particles, the function returns
%a 1
if n == 0
    Result = 1;
else

    %Matrix that represents the poly x1x2...xn
    t1 = ones(1,n);
    %Matrix that represents the poly (1+x1+x2+...+xn) these
    %will be turned to minuses during integration
    t2 = [zeros(1,n); unique(perms([1 zeros(1,n-1)]),'rows')];

    %For loop that calles genSet that creats a matrix
    %that represents all possible polynomials needed.
    PRep = [];
```

```
for s = 0:deg
    PRep = [PRep; genSet(s,n,s)];
end

%Declares some needed variables for the later loop.
r = size(PRep);
Result = [];
f4 = [];
for i = 1:r(1)
    for j = 1:r(1)
        %Uses PRep to create the base polynomials
        f2 = unique(perms(PRep(i,:)), 'rows');
        f3 = unique(perms(PRep(j,:)),'rows');

        %Multiplies the four polynomials together
        f4 = PolM(f2,f3);
        t3 = PolM(t1,t2);
        t4 = PolM(f4,t3);

        %This loop, integrates the final resulting
        %polynimial represented by the matrix t4
        t4s = size(t4);
        EndInt = 0;
        for s = 1:t4s(1)
            Numer = 1;
            %This loop determines the final numerator
            %for the integrated value
            for t = 1:t4s(2)
                Numer = Numer*factorial(t4(s,t));
            end

            %This if statement determines if the term
```

```
                %from integration needed to be subtracted
                %or added to the final result.
                if sum(t4(s,:)) == sum(f4(1,:)) + n
                    EndInt = EndInt
                    + Numer/factorial(sum(t4(s,:)) + n);
                else
                    EndInt = EndInt
                    - Numer/factorial(sum(t4(s,:)) + n);
                end
            end

            %Records the result into a matrix called Result
            Result(i,j) =
            EndInt/(Nor(i,i)^(1/2)*Nor(j,j)^(1/2));
        end
    end
end

end

function [Result] = Norm(deg,n)
%Inputs: deg, The max polynomial degree, and n, number
%of particles.
%
%Outputs: Result, a square matrix with only diagonal
%elements that are the inverse square of the
%normalization constants for the basis polynomials
%
%This function helps normalize the basis polynomials
%by arrangeing them in order on the diagonal of a
%matrix. This allows them to be used easily with
%functions OffNorm, OnDNorm, and BaseNorm.
```

```
%If statement for if no particles are present, Norm just
%prints out 1.
if n == 0
    Result = 1;
else
    %Represents polynomial x1*x2*...*xn
    t1 = ones(1,n);

    %Represents polynomial (1-x1-x2-...-xn)
    t2 = [zeros(1,n); unique(perms([1 zeros(1,n-1)]),'rows')];

    %Creates a matrix that represents all the possible
    %degree configurations
    PRep = [];
    for s = 0:deg
        PRep = [PRep; genSet(s,n,s)];
    end

    %Initializes some values later needed for the loop
    r = size(PRep);
    Result = [];
    f4 = [];
    %Loop that makes each element of the final matrix
    for i = 1:r(1)
        %Matrices that represent the corresponding
        %polynomial to the row in PRep
        f2 = unique(perms(PRep(i,:)), 'rows');

        %Multiplies the polynomials together one by one
        %and creates a matrix that represents the final
        %result
```

```
        f4 = PolM(f2,f2);
        t5 = PolM(t1,t2);
        t6 = PolM(f4,t5);

        %This loop simulates integrating the polynomial
        t6s = size(t6);
        EndInt = 0;
        for s = 1:t6s(1)
            %Small loop to define the numerator for the
            %integral result
            Numer = 1;
            for t = 1:t6s(2)
                Numer = Numer*factorial(t6(s,t));
            end

            %This if statement is used to determine the
            %sign on each
            %term in the polynomial
            if sum(t6(s,:)) == sum(f4(1,:)) + n
            EndInt = EndInt + Numer/factorial(sum(t6(s,:))
            + n);
            else
            EndInt = EndInt - Numer/factorial(sum(t6(s,:))
            + n);
            end
        end

        %Records the final result
        Result(i,i) = EndInt;
    end
end
```

```
end

function [Result] = RelP(g,dStor,Vector,N)
%Inputs: g, The coupling strength, Vector, a vector that
%contains the eigenvector from the generalized eigenvalue
%of C and D, and N, a vector of the highest degrees for
%each sector.
%
%Outputs: Result, a matrix of values.
%
%This function looks at the relative probabilities of the
%sectors provided to it. This function completes this by
%being given the eigenvector for the C and D problem.
%The relative probabilty is Integrate[\Psi_i^2/Psi_0^2].

%The number of particles is needed for this loop as well
%as the an index of where in the eigenvector the loop needs
%to be. That is where tots comes in.
n = length(N)-1;
tots = 0;
Result = [];

%Loop that goes through every result from
%FockGenMultNorm2.m.
for i = 1:n
    %Need to know how many eigenvalues there were.
    dSize = size(dStor{i+1});
    sepVec = zeros(dSize(1),1);

    %Stores the desired basis polynomial coefficients.
    for t = 1:dSize(1)
        sepVec(t) = Vector(1+tots+t);
```

```
    end

    %Creates \Psi_i^2.
    tempResult = transpose(sepVec)*dStor{i+1}*sepVec;

    %Divides \Psi_i^2 by \Psi_0.
    Result = [Result tempResult/Vector(1)^2];

    %Records which \Psi_i we are on.
    tots = tots + dSize(1);
end

%Returns Results.
Result = [g Result];

function [Result] = OnDNorm(deg,n,mt)
%Inputs: deg, The max polynomial degree, and n, number
%of particles.
%
%Outputs: Result, a square matrix.
%
%This function creates the cnn matrix for the final C
%matrix. These matrices are block diagonal in C. This
%function makes matrices representing
%Integrate[mt*mt*x1*x2*...*xn + x1*x2*...*xn-1
%*(1-x1-x2-...-xn) + x1*x2*...*xn-2*xn*(1-x1-x2-...-xn)
%+ ... + x2*...*xn*(1-x1-x2-...-xn)
%*QNk^(n)*QNk^(n) dx1dx2...dxn.

%If the number of particles is 0, returns only mt.
if n == 0
    Result = mt*mt;
```

```
else

    %Creates a matrix with inverse squares of the
    %normalization constants for the polynomials
    Nor = Norm(deg,n);

    %Represents the polynomial x1*x2*...*xn and
    %(1-x1-x2-...-xn)
    t1 = ones(1,n);
    t2 = [zeros(1,n); unique(perms([1 zeros(1,n-1)]),'rows')];

    %Represents the polynomial of x1*x2*...*xn-1 added
    %with all permutations of the missing variable.
    t3 = unique(perms([0 ones(1,n-1)]),'rows');

    %Creates a matrix that represents all the possible
    %degree configurations
    PRep = [];
    for s = 0:deg
        PRep = [PRep; genSet(s,n,s)];
    end

    %Variables needed for the loop
    r = size(PRep);
    Result = [];
    f4 = [];

    %Loop that makes all the elements for the final matrix.
    for i = 1:r(1)
        for j = 1:r(1)

            %Matrices that represent the corresponding
```

```
%polynomial to the row in PRep
f2 = unique(perms(PRep(i,:)), 'rows');
f3 = unique(perms(PRep(j,:)),'rows');

%Multiplies all the matrices together
f4 = PolM(f2,f3);
t5 = PolM(t3,t2);
t6 = PolM(f4,t5);
t7 = PolM(f4,t1);

%Variables needed for loops
t6s = size(t6);
EndInt = 0;

%The resulting polynomial has varying signs
%of +/- so, two loops are needed to get the
%final result.
for s = 1:t6s(1)
    %Numerator for the result from integration
    Numer = 1;
    for t = 1:t6s(2)
        Numer = Numer*factorial(t6(s,t));
    end

    %This if statement adds or subtracts the
    %results depending on the degree of the
    %polynomial term before integration
    if sum(t6(s,:)) == sum(f4(1,:)) + n - 1
    EndInt = EndInt
    + Numer/factorial(sum(t6(s,:)) + n);
    else
    EndInt = EndInt
```

```
                - Numer/factorial(sum(t6(s,:)) + n);
                end
            end

            %Same loop as earlier without the if statement.
            t7s = size(t7);
            for s = 1:t7s(1)
                Numer = 1;
                for t = 1:n
                    Numer = Numer*factorial(t7(s,t));
                end
                EndInt = EndInt
                + mt*mt*Numer/factorial(sum(t7(s,:)) + n);
            end

            Result(i,j) =
            EndInt/(Nor(i,i)^(1/2)*Nor(j,j)^(1/2));
        end
    end
end

end

function [Result] = OffDNorm(deg1,deg2,n)
%Inputs: deg1, The max polynomial degree of the n-1 number
%of particles, deg2, the max polynomial degree of the
%n number of particles.
%
%Outputs: Result, a rectangular matrix that represents the
%n particle state interacting with the n-1 particle state
%
%This function creates the off diagonal matrices for the
```

```
%final C matrix. This matrix has the form cnn-1. This also
%creates the cn-1n matrix since they are just transposes
%of each other. This code represents
%n^(1/2)Integral[x1*x2*...*xn-1Q_Nk^(n)Q_Nk^(n-1)
%*dx1dx2...dxn

%Creates the inverse squares of the normalization constants
%for n-1 and n particle states
Nor1 = Norm(deg1,n-1);
Nor2 = Norm(deg2,n);

%Represents the polynomial x1*x2*...*xn-1
t1 = [ones(1,n-1) 0];

%Creates a matrix that represents all the possible degree
%configurations
PRep1 = [];
PRep2 = [];
for s = 0:deg2
    PRep2 = [PRep2; genSet(s,n,s)];
end

for s = 0:deg1
    PRep1 = [PRep1; genSet(s,n-1,s)];
end

%Initializes some values later needed for the loop
r1 = size(PRep1);
r2 = size(PRep2);
Result = [];
f4 = [];
f5 = [];
```

```
%Loop that makes each element of the final matrix
for i = 1:r1(1)
    for j = 1:r2(1)
        %Matrices that represent the corresponding
        %polynomial to the row in PRep
        f2 = unique(perms(PRep2(j,:)), 'rows');
        g2 = unique(perms(PRep1(i,:)),'rows');
        g2s = size(g2);

        %g2 is not the same size as f2 due to the f2 having
        %an extra particle xn. The terms in the polynomial
        %need to be added in so f2 and g3 can be multiplied
        %together correctly.
        g3 = [g2 zeros(g2s(1),1)];

        %Multiplies polynomials together to get the final
        %polynomial.
        f4 = PolM(f2,g3);
        f5 = PolM(f4,t1);

        %Initializing parameters for the loop.
        t5s = size(f5);
        EndInt = 0;

        %Loop that integrates the polynomial.
        for s = 1:t5s(1)
            %Numerator for the result from integration
            Numer = 1;
            for t = 1:t5s(2)
                Numer = Numer*factorial(f5(s,t));
            end
            EndInt = EndInt + Numer/factorial(sum(f5(s,:))
```

```
            + n);
        end

        Result(j,i) = n^(1/2)*EndInt/(Nor1(i,i)^(1/2)
        *Nor2(j,j)^(1/2));
    end
end

end

function [sets] = genSet(N,n,max)
%Inputs: N and max are the highest order of the basis
%polynomials. n is the number of particles.
%
%Outputs: sets, a matrix where a row contains m_i where
%m_i is x_i^m_i for the basis polynomials.
%
%This function produces a vector containing the degrees
%of some of the terms in the basis polynomials. This matrix
%will have its rows permuted to created the rest of the
%terms. This function is recursive. For any value of N, all
%lower values must also be used for the expansion.

%If statement to stop the code from running forever.
if N<1
    %if N is 0 then all variables in the term have degree
    %zero.
    sets = zeros(1,n);
else
    %Initializes the matrix
    sets = [];
    %The total degree for each term can only be N. This
```

```
    %loop makes sure to this occurs.
    for i = ceil(N/n):N
        %If statement needed for calling the function again.

        if i <= max
            x = genSet(N-i,n-1,i);
            %If statment used for when x calles genSet with
            %N<0.
            if isempty(x)
                sets = [sets; i];
            else
                %Loop that adds x to the result and stores
                %the result in sets.
                k = size(x);
                for j = 1:k(1)
                    temp = [i x(j,:)];
                    sets = [sets; temp];
                end
            end
        end
    end
end
```

A.2 First and Second Order in LFCC Using Mathematica

```
(* ::Package:: *)

(*PollyFun initializes what the polynomials needed for
LightMatrixMaker.wl*)

(*genSet is a recursive function that defines the exponents on
 the polynomials used in the second order approximation*)
```

```
genSet[deg_,n_,max_] := Module[{sets,i,k,j,x},

(*If statement that stops the function from calling itself
 forever*)
If[deg<1,
    sets = {ConstantArray[0,n]};
        For[r = 1,r<n+1,r++]
,
    sets = {}; (*Initializes some variables and starts the loop*)
    For [i = Ceiling[(deg/n)], i<deg+1,i++,
        If [i <= max,
            x = genSet[deg-i,n-1,i];
            If [x== {},
                                tempsize = Dimensions[sets];
                sets = Insert[sets, i,tempsize[[1]]+1];
            ,
                k = Dimensions[x];
                For [j = 1, j<k[[1]]+1,j++,
                    temp = Flatten[{i,x[[j]]}];
                                tempsize = Dimensions[sets];
                    sets = Insert[sets, temp,tempsize[[1]]+1];
                ]
            ]
        ]
    ]
];

sets
];

(*Gets the mth term of the deg degree polynomial for two
 variables Used in second order approximation*)
```

```
polly2[x_,y_,deg_,m_] := Module[{s,PRep,PRepSize,PRep2,q,temp,
sizetemp,tempPol1,tempPol,polly2,i,k,j},
(*Initializes variables*)
polly2={};
PRep = {};
(*loop to get the exponents from genSet*)
For[ s = 0, s<deg+1,s++,
        PRep = Join[PRep, genSet[s,2,s]];
];

PRepSize = Dimensions[PRep];
PRep2 = {};
(*loop puts the exponents with their proper variables x and y*)
For[ q = 1, q<PRepSize[[1]]+1,q++,
        temp = Flatten[DeleteDuplicates[Permutations[PRep[[q]]]]];
        sizetemp = Dimensions[temp];
        tempPol1 = 0;
        For[ j = 0, j<sizetemp[[1]]/2,j++,
                tempPol = x^temp[[1+2*j]]*y^temp[[2+2*j]];
                tempPol1 = tempPol1 + tempPol;
        ];
        polly2 = Insert[polly2, tempPol1,-1];
];

polly2[[m]]
]

(*Same as previous function but for one variable*)
polly1[x_,deg_,m_] := Module[{tempPol,polly2,j},
polly2={};
For[ j = 0, j<deg+1,j++,
        tempPol = x^j;
```

```
        polly2 = Insert[polly2, tempPol, -1];
];

polly2[[m]]
]

(* ::Package:: *)

(*IntEs.wl is a code written to initialize functions
in Mathematica. The functions written here Integrate polynimials.
There are multiple functions since an integration with two
variables instead of three is slightly different.*)

(*Names the function and all local variables*)
Int3Est01[poly_,x_,y_,z_] := Module[{ExPoly,ListPoly,ListPolyLeng,
i,j,ExpTab,Coefsub,CoefTab,polySub,Result1,Result,ExtraTerm},

(*Expands the given polynomial*)
ExPoly = Expand[poly] + ExtraTerm;

(*Makes each term into a list*)
ListPoly2 = ExPoly /. Plus -> List;
ListPoly = DeleteCases[ListPoly2,ExtraTerm];
ListPolyLeng = Dimensions[ListPoly];

(*Makes a list of the exponents for each term*)
ExpTab = Table[{Exponent[ListPoly[[m]],x],Exponent[ListPoly[[m]]
,y],Exponent[ListPoly[[m]],z]},{m,1,ListPolyLeng[[1]]}];

(*Gathers the coefficients for each term into a list*)
Coefsub = CoefficientRules[ListPoly,{x,y,z}];
CoefTab = Table[ExpTab[[m]] /. Coefsub[[m]],{m,1,
```

```
ListPolyLeng[[1]]}];

(*Variable initialization*)
Result1 = ConstantArray[0,ListPolyLeng[[1]]];

(*Loop that acts as the integration.*)
For[i = 1, i < ListPolyLeng[[1]]+1,i++,
        polySub = 1;
        For[j = 1, j < 4,j++,
                polySub = polySub/(ExpTab[[i,j]]+1);
        ];
        Result1[[i]] = polySub;
];

(*Combines the coefficients from the terms and multiplies them
 by the integration outcome then sums them together.*)
Result = Total[Result1*CoefTab]

]

(*Does the same as the function above but with two vairables
instead of three*)
Int2Est01[poly_,x_,y_] := Module[{ExPoly,ListPoly,
ListPolyLeng,i,j,ExpTab,Coefsub,CoefTab,polySub,
Result1,Result,ExtraTerm},

(*Expands the given polynomial*)
ExPoly = Expand[poly] + ExtraTerm;

(*Makes each term into a list*)
ListPoly2 = ExPoly /. Plus -> List;
ListPoly = DeleteCases[ListPoly2,ExtraTerm];
```

```
ListPolyLeng = Dimensions[ListPoly];

(*Makes a list of the exponents for each term*)
ExpTab = Table[{Exponent[ListPoly[[m]], x], Exponent[ListPoly[[m]],
y]}, {m, 1, ListPolyLeng[[1]]}];

(*Gathers the coefficients for each term into a list*)
Coefsub = CoefficientRules[ListPoly, {x, y}];
CoefTab = Table[ExpTab[[m]] /. Coefsub[[m]], {m, 1,
ListPolyLeng[[1]]}];

(*Variable initialization*)
Result1 = ConstantArray[0, ListPolyLeng[[1]]];

(*Loop that acts as the integration.*)
For[i = 1, i < ListPolyLeng[[1]]+1, i++,
        polySub = 1;
        For[j = 1, j < 3, j++,
                polySub = polySub/(ExpTab[[i, j]]+1);
        ];
        Result1[[i]] = polySub;
];
(*Combines the coefficients from the terms and multiplies them
 by the integration outcome then sums them together.*)
Result = Total[Result1*CoefTab]

]

(*2 variables with different integration bounds*)
Int2Est[poly_, x_, y_] := Module[{ExPoly, ListPoly,
ListPolyLeng, i, j, ExpTab, Coefsub, CoefTab, polySub,
Result1, Result, ExtraTerm},
```

```
(*Expands the given polynomial*)
ExPoly = Expand[poly] + ExtraTerm;

(*Makes each term into a list*)
ListPoly2 = ExPoly /. Plus -> List;
ListPoly = DeleteCases[ListPoly2,ExtraTerm];
ListPolyLeng = Dimensions[ListPoly];

(*Makes a list of the exponents for each term*)
ExpTab = Table[{Exponent[ListPoly[[m]],x],
Exponent[ListPoly[[m]],y]},{m,1,
ListPolyLeng[[1]]}];

(*Gathers the coefficients for each term into a list*)
Coefsub = CoefficientRules[ListPoly,{x,y}];
CoefTab = Table[ExpTab[[m]] /. Coefsub[[m]],{m,1,
ListPolyLeng[[1]]}];

(*Variable initialization*)
Result1 = ConstantArray[0,ListPolyLeng[[1]]];

(*Loop that acts as the integration.*)
For[i = 1, i < ListPolyLeng[[1]]+1,i++,
        polySub = 1;
        polySub = Factorial[ExpTab[[i,2]]]*Factorial[
        ExpTab[[i,1]]]/Factorial[ExpTab[[i,1]]
        +ExpTab[[i,2]]+2];
        Result1[[i]] = polySub;
];
(*Combines the coefficients from the terms and multiplies them
 by the integration outcome then sums them together.*)
```

```
Result = Total[Result1*CoefTab]

]

(*One variable integration*)
Int1Est01[poly_,x_] := Module[{ExPoly,ListPoly,ListPolyLeng,i,j,
ExpTab,Coefsub,CoefTab,polySub,Result1,Result,ExtraTerm},

(*Expands the given polynomial*)
ExPoly = Expand[poly] + ExtraTerm;

(*Makes each term into a list*)
ListPoly2 = ExPoly /. Plus -> List;
ListPoly = DeleteCases[ListPoly2,ExtraTerm];
ListPolyLeng = Dimensions[ListPoly];

(*Makes a list of the exponents for each term*)
ExpTab = Table[{Exponent[ListPoly[[m]],x]},{m,1,
ListPolyLeng[[1]]}];

(*Gathers the coefficients for each term into a list*)
Coefsub = CoefficientRules[ListPoly,x];
CoefTab = Table[ExpTab[[m]] /. Coefsub[[m]],{m,1,
ListPolyLeng[[1]]}];

(*Variable initialization*)
Result1 = ConstantArray[0,ListPolyLeng[[1]]];

(*Loop that acts as the integration.*)
For[i = 1, i < ListPolyLeng[[1]]+1,i++,
        polySub = 1;
        polySub = polySub/(ExpTab[[i,1]]+1);
```

```
        Result1[[i]] = polySub;
];
(*Combines the coefficients from the terms and multiplies
 them by the integration outcome then sums them together.*)
Result = Total[Result1*CoefTab]

]

(* ::Package:: *)

(*LightMatrixMaker2.wl creates the matrices and vectors to be
 used to solve the final equation for the LFCC method*)

(*f[] is a function that defines the basis polynomials for single
 variable polynomials.*)
f[b_,x_] = JacobiP[b,1,1,2*x - 1] * (((b + 2)/(b + 1))
* (2*b + 3))^(1/2);
(*k dictates the degree of basis polynomials to be used*)
k = 2;
k2 = 2;
w = k+1;
w2temp = Dimensions[polly2[x,y,k2,All]];
w2 = w2temp[[1]];

(*These variables call the function polly2. This defines all the
needed two variable basis polynomials*)
px1z2 = polly2[x1,z2*(1-x1),k2,All]
px1z1 = polly2[x1,z1*(1-x1),k2,All]
px1yx1 = polly2[x1,y*(1-x1),k2,All]
px1x2 = polly2[x1,x2,k2,All]
px1y = polly2[x1,y,k2,All]
pz1z2 = polly2[z2,z1,k2,All]
```

```
pzy = polly2[z1*(1-y),z2*(1-y),k2,All]
pxzx = polly2[x1,z2*(1-x1)*(1-z1),k2,All]
pzyz = polly2[z1(1-y),z2*(1-y)*(1-z1),k2,All]
pzyzz = polly2[z1(1-y),z2*(1-z1(1-y)),k2,All]
pxzxy = polly2[x1,z2*(1-x1)(1-z1),k2,All]
pz1z2z2 = polly2[z1(1-z2),z2,k2,All]
pz1z2z1 = polly2[z1,z2(1-z1),k2,All]

(*Defines the matrices. m and n are reversed relative to the
 thesis and that there are other permutations. These
 permutations should not matter because they multiply
 symmetric products of coefficients, such as a_n a_l a_s.*)
PTloop = Table[Int1Est01[f[n,x],x], {n, 0, k}];
PTu = Table[Int1Est01[(1-x)*f[m,x]*f[n,x],x],{m, 0, k},
{n, 0, k}];
PTloopTsub = Table[Int1Est01[x*f[m,x]*f[n,x],x], {m, 0, k},
{n, 0, k}];
PTTloopsub = Table[Int2Est01[f[m,(1-y)*z]*(z*(1-y))*f[n,y]
*f[s,z],z,y], {m, 0, k},{n, 0, k},{s, 0, k}];
TP = Table[Int1Est01[x*x*f[m,x]*f[n,x],x],{m, 0, k},
{n, 0, k}];
TPTsub = Table[Int1Est01[x*(1-x)*f[m,x]*f[n,x],x]  ,
{m, 0, k},{n, 0, k}];
PT2loop = Table[Int2Est[f[m,x1]*x1*px1y[[n]],y,x1], {m,0,k},
{n, 1, w2}];

P1T = Table[Int2Est[f[n,x1]*x1*px1x2[[m]],x2,x1], {m,1,w2},
{n, 0, k}];
PxT2 = Table[Int2Est[x1*x2*px1x2[[m]]*px1x2[[n]],x2,x1],{
m,1,w2},{n, 1, w2}];
PuT2 = Table[Int2Est[(1-x1-x2)*x2*px1x2[[m]]*px1x2[[n]],
x2,x1],{m,1,w2},{n, 1, w2}];
```

```
PTTusub = Table[Int2Est01[z2*(1-x1)*(1-x1)*(1-z2)*px1z2[[m]]
*f[n,x1]*f[l,z2],z2,x1],{m,1,w2},{n, 0, k},{l,0,k}];
PTuTsub = Table[Int2Est01[x1*(1-x1)*(1-z2)*px1z2[[m]]
*f[n,x1]*f[l,z2],z2,x1],{m,1,w2},{n, 0, k},{l,0,k}];
PxTTsub = Table[Int2Est01[x1*z2*(1-x1)*px1z2[[m]]*f[n,z2]
*f[l,x1],z2,x1],{m,1,w2},{n, 0, k},{l, 0, k}];
PTT2loopsub = Table[Int3Est01[(1-z1)*x1*z2*(1-x1)^2
*pxzxy[[m]]*px1z1[[l]]*f[n,z2],z2,z1,x1],{m,1,w2},
{n, 0, k},{l, 1, w2}];
PT2loopTsub =Table[Int3Est01[(1-x1)*(1-z2)*z2*x1*px1z2[[m]]
*f[n,x1]*pz1z2z2[[l]],z1,z2,x1],{m,1,w2},{n, 0, k},
{l, 1, w2}];
PT2Tloopsub =Table[Int3Est01[z1*z2*(1-y)*(1-y)*(1-z1)*
(1-z1)*pzyz[[m]]*pz1z2z1[[l]]*f[n,y],y,z1,z2],{m,1,w2},
{n, 0, k},{l, 1, w2}];
PTTTloopsub = Table[Int3Est01[z1*(1-y)*z2*(1-z1)*(1-y)
*pzyz[[m]]*f[s,z2]*f[n,z1]*f[l,y],z2,z1,y], {m,1,w2},
{n,0,k},{l, 0, k},{s,0,k}];
PTTloopTsub =Table[Int3Est01[x1*z2*(1-x1)*(1-z1)*pxzx[[m]]
*f[n,z2]*f[l,x1]*f[s,z1],z2,z1,x1],{m,1,w2},{n, 0, k},
{l, 0, k},{s,0,k}];
PTloopTTsub1 =Table[Int2Est01[x1*z2*(1-x1)*px1z2[[m]]
*f[n,z2]*f[l,x1],x1,z2],{m,1,w2},{n, 0, k},{l, 0, k}];
TP1 = Table[Int2Est01[f[n,z2]*z2*(1-x1)*(1-z2)*px1z2[[m]],
z2,x1], {m,1,w2},{n, 0, k}];
TPxTsub = Table[Int2Est01[x1*z2*(1-x1)*(1-z2)*px1z2[[m]]
*f[n,z2]*f[l,x1],z2,x1],{m,1,w2},{n, 0, k},{l, 0, k}];
TPT2sub =Table[Int3Est01[x1*z2*(1-x1)*(1-x1)*(1-z2)
*px1z2[[m]]*px1yx1[[l]]*f[n,z2],y,z2,x1],{m,1,w2},
{n, 0, k},{l, 1, w2}];
TPTTloopsub =Table[Int3Est01[z1*z2*(1-y)*(1-(1-y)z2)^2(
1-z2)*pzyzz[[m]] f[s,z2]*f[n,z1]*f[l,y],y,z2,z1],{m,1,w2},
```

```
{n, 0, k},{l, 0, k},{s,0,k}];
TPTloopTsub = Table[Int3Est01[x1*z2*(1-x1)*(1-z2)
*px1z2[[m]]*f[n,z2]*f[l,x1],z1,z2,x1],{m,1,w2},
{n, 0, k},{l, 0, k}];
T2Psub = Table[Int2Est[x1*x2*(1-x1-x2)*px1x2[[m]]
*px1x2[[n]],x2,x1],{m,1,w2},{n, 1, w2}];
TTPsub = Table[Int2Est01[x1*z2*(1-x1)*(1-x1)*(1-z2)
*px1z2[[m]]*f[n,x1]*f[l,z2],z2,x1],{m,1,w2},{n, 0, k},
{l,0,k}];
TTPTsub = Table[Int2Est01[x1*z2*(1-x1)*(1-x1)*(1-z2)
*px1z2[[m]]*f[n,x1]*f[l,z2],z2,x1],{m,1,w2},{n, 0, k},{l,0,k}];



(*Some equations are similar enough where we only need to
 multiply by PTloop.*)

(*Initializes variables*)
PTloopTsub2 = Table[i + j + g, {i,0,k}, {j,0,k},{g,0,k}];
TPTsub2 = PTloopTsub2;

TPTloopTsub2 = Table[i + j + g + l, {l,0,w2-1}, {j,0,k},
{g,0,k},{i,0,k}];
T2PTsub = Table[i + j + g, {i,0,w2-1}, {j,0,k},{g,0,w2-1}];
PTloopT2sub = T2PTsub;
PTloopTTsub2 = TPTloopTsub2;
TTPTsub2 = TPTloopTsub2;

(*Defines those variables for each element in a loop*)
For [m=1, m<w+1, m++,
        For [n=1, n<w+1, n++,
                For [s=1, s<w+1 , s++,
```

```
                        PTloopTsub2[[m,n,s]] = PTloop[[s]]
                        *PTloopTsub[[m,n]];
                        TPTsub2[[m,n,s]] = PTloop[[s]]
                        *TPTsub[[m,n]];
                        For [l=1, l<w2+1 , l++,
                                TPTloopTsub2[[l,n,s,m]] =
                                PTloop[[m]]*TPTloopTsub[[l,n,s]];
                                PTloopTTsub2[[l,n,s,m]] =
                                PTloop[[m]]*PTloopTTsub1[[l,n,s]];
                                TTPTsub2[[l,n,s,m]] = PTloop[[m]]
                                *TTPTsub[[l,n,s]];
                        ]
                ]
        ]
]

(*Different shaped matrices for this loop*)
For [s=1, s<w2+1, s++,
        For [n=1, n<w2+1, n++,
                For [m=1, m<w+1 , m++,
                        T2PTsub[[n,m,s]] = PTloop[[m]]
                        *T2Psub[[n,s]];
                        PTloopT2sub[[n,m,s]] = PTloop[[m]]
                        *PxT2[[n,s]];
                ]
        ]
]

(*General list of variables then A is a list of the variables
correponding to the k value*)
AL = {a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14};
A = Take[AL, w];
```

```
AL2 = {c0 ,c1 ,c2 ,c3 ,c4 ,c5 ,c6 ,c7 ,c8 ,c9 ,c10 ,c11 ,c12 ,c13 ,c14 ,c15 ,
c16 ,c17 ,c18 ,c19 ,c20 ,c21 ,c22 ,c23 ,c24 ,c25 ,c26 ,c27 ,c28 ,c29 ,c30 };
A2 = Take[AL2, w2];

(*Matrices sometimes the wrong dimension such as kxkxk but need
 to be kx(k*k)*)
PTloopT = ArrayReshape[PTloopTsub2,{w,w*w}];
PTTloop = ArrayReshape[PTTloopsub,{w,w*w}];
TPT = ArrayReshape[TPTsub2,{w,w*w}];
TTP = ArrayReshape[TTPsub,{w2,w*w}];
T2PT = ArrayReshape[T2PTsub,{w2,w2*w}];
PTloopT2 = ArrayReshape[PTloopT2sub,{w2,w2*w}];
TPTloopT = ArrayReshape[TPTloopTsub2,{w2,w*w*w}];
PxTT = ArrayReshape[PxTTsub,{w2,w*w}];
TPxT = ArrayReshape[TPxTsub,{w2,w*w}];
PTloopTT = ArrayReshape[PTloopTTsub2,{w2,w*w*w}];
TTPT = ArrayReshape[TTPTsub2,{w2,w*w*w}];
PTTu = ArrayReshape[PTTusub,{w2,w*w}];
PTuT = ArrayReshape[PTuTsub,{w2,w*w}];

PTT2loop = ArrayReshape[PTT2loopsub,{w2,w2*w}];
PTTloopT = ArrayReshape[PTTloopTsub,{w2,w*w*w}];
TPT2 = ArrayReshape[TPT2sub,{w2,w2*w}];
TPTTloop = ArrayReshape[TPTTloopsub,{w2,w*w*w}];
PTTTloop = ArrayReshape[PTTTloopsub,{w2,w*w*w}];
PT2loopT = ArrayReshape[PT2loopTsub,{w2,w2*w}];
PT2Tloop = ArrayReshape[PT2Tloopsub,{w2,w2*w}];
(*Pre-allocates the size for the kxk vector*)
BLM = Table[1,{i,w},{j,w}];
BLM2 = Table[1,{i,w},{j,w},{n,w}];
BLMmix21 = Table[1,{i,w},{j,w2}];
```

```
(*Fills BLM with the correct variable combinations*)
For [n=1, n<w+1, n++,
        For [s=1, s<w+1 , s++,
                BLM[[n,s]] = AL[[n]]*AL[[s]];
                For [m=1, m<w+1 , m++,
                        BLM2[[n,s,m]] = AL[[n]]*AL[[s]]*AL[[m]];
                ]
        ]
]

(*Fills BLMmix21 with the correct variable combinations*)
For [s=1, s<w+1 , s++,
        For [n=1, n<w2+1, n++,
                For [m=1, m<w2+1 , m++,
                        BLMmix21[[s,m]] = AL[[s]]*AL2[[m]];
                ]
        ]
]

(*Makes a (k*k)x1 vector to be multiplied by the matrices*)
B = Flatten[BLM]
B2 = Flatten[BLM2]
Bmix21 = Flatten[BLMmix21]
A
A2

(* ::Package:: *)

(*LightFinalMult2.wl uses the Matrices and vectors from
 LightMatrixMaker2.wl to solve the system of equations
for the variables a1,a2 and so on then calcualtes the
```

```
value for (M/\mu)^2 with first and  second approximation.*)

(*Control parameters \tilde{m}. t is the current iteration
 with l being the max value for \lambda. inc controls how
many values of \lambda are tested. Then there are a bunch
of variables for storing results*)
mt = 1;
t = 1;
l = 1.5;
inc = 100;
Final = Table[u c  ,{u,inc+1},{c,2}];
Final2 = Final;
Final2Multi = Table[u c  ,{u,inc+1},{c,200}];
solvePlots = Table[u  ,{u,200}];
FinalOrder1 = Final;
Alength = Dimensions[A];
A2length = Dimensions[A2];

(*These variables will later be used to test the result given
 by FindRoot[]*)
Atest = A;
A2test = A2;
BLMtest =BLM;
BLM2test = BLM2;
BLMmix21test = BLMmix21;
Btest = B;
B2test = B2;
Bmix21Test = Bmix21;
tempVec = Table[u  ,{u,inc+1}];
Finaltest = MapThread[Append,{Final,tempVec}];
Finaltest = MapThread[Append,{Finaltest,tempVec}];
```

```
(*Takes the two coefficient vectors and combines them into a
single vector.*)
A3 = Join[A,A2];

(*Need to know how many coefficients there are along with a
way to store them all*)
Coeffs = Dimensions[A3];
Final2MultiCoef = Table[u c  ,{u,2},{c,Coeffs[[1]]+1}]

(*Initializes the initial guesses for FindRoot[]*)
Guess3 = A3;
Guess = A;

(*Loops that make the very first initial guess all 0*)
For  [r=1, r<w+1, r++,
        Guess[[r]] = {AL[[r]],0};
];

For  [r=1, r<w+w2+1, r++,
        Guess3[[r]] = {A3[[r]],0};
]

(*Check to make sure it worked*)
Print[Guess3];

(*This begins the loop to solve the system of equations for
multiple values for \lambda. In this code, \lambda is g*)
For  [g=0.01, g<l+l/inc, g = g + l/inc,

(*Creats the desired system of equations and stores them as
 a matrix*)
```

```
EquationOrder1 = g*PTloop + PTu.A + g*PTloopT.B/2
+ g*PTTloop.B/2 + mt^2*TP.A  - g*TPT.B;

Equation = g*PTloop + PTu.A + g*PTloopT.B/2 + g*PTTloop.B/2
 + mt^2*TP.A  - g*TPT.B  + 2*g*PT2loop.A2;

Equation2 = 2*g*P1T.A + 2*mt*mt*PxT2.A2 + 4*PuT2.A2 + PTTu.B
 + PTuT.B + mt*mt*PxTT.B + g*PTloopT2.Bmix21
+ 2*g*PTT2loop.Bmix21 + 2*g*PT2loopT.Bmix21 + g*PT2Tloop.Bmix21
 + g*PTTTloop.B2/3 + g*PTTloopT.B2/3 + g*PTloopTT.B2/3
- 2*g*TP1.A - 2*mt*mt*TPxT.B  - 2*PTTu.B - 4*g*TPT2.Bmix21
 - g*TPTTloop.B2 - g*TPTloopT.B2 - 2*mt*mt*T2Psub.A2
 - 2*g*T2PT.Bmix21 + mt*mt*TTP.B + g*TTPT.B2;

(*Find root needs an initial guess. First, pre-allocate size
 then makes the previous solution first order the initial
 guess for all variables*)

Equation3 = Join[Equation,Equation2];
ResultFROrder1 = A /. FindRoot[{EquationOrder1}, Guess];
For [r=1, r<w+1, r++,
        Guess3[[r]] = {A[[r]],ResultFROrder1[[r]]};
];
ResultFR = A3 /. FindRoot[{Equation3}, Guess3];
DL = Dimensions[ResultFR];
MatrixForm[ResultFR];

(*Solve[] used to see all the solutions. This is only used
for testing at low values for basis polynomial order*)
ResultL = A3 /. NSolve[{Equation3 == 0}, A3];
Print[MatrixForm[ResultL]];
```

```
(*Records how many solutions Solve[] produced*)
ResLength = Dimensions[ResultL];

(*Uses the result from FindRoot to find (M/\mu)^2*)
t1t = Sum[ResultFR[[v+1]]*f[v,y],{v,0,k}];
t1tOrder1 = Sum[ResultFROrder1[[v+1]]*f[v,y],{v,0,k}];
t1t2 = Sum[ResultL[[ResLength[[1]],v+1]]*f[v,y],{v,0,k}];

(*This loop saves all the results from Solve[] for testing*)
For [r=1, r<ResLength[[1]]+1, r++,
Final2MultiCoef = Append[Final2MultiCoef,Join[{g},
ResultL[[r]]]];
];

(*This loop saves all the values for (M/\mu)^2 Solve[]
produced for tesing*)
For [r=1, r<ResLength[[1]]+1, r++,
t1t2Multi = Sum[ResultL[[r,v+1]]*f[v,y],{v,0,k}];
PolyLMulti = Integrate[t1t2Multi,{y,0,1}];
Final2Multi[[t,r+1]] = mt*mt + g*PolyLMulti;
];

(*Stores all the results and prints them to check them
as the code works*)
Final2Multi[[t,1]] = g;
PolyFR = Integrate[t1t,{y,0,1}];
PolyFROrder1 = Integrate[t1tOrder1,{y,0,1}];
PolyL = Integrate[t1t2,{y,0,1}];
Print[ResultFR,g];
Print[ResultFROrder1,g];
Final[[t,2]] = mt*mt + g*PolyFR;
```

```
Final[[t,1]] = g;
FinalOrder1[[t,2]] = mt*mt + g*PolyFROrder1;
FinalOrder1[[t,1]] = g;

Finaltest[[t,2]] = Final[[t,2]];
Finaltest[[t,1]] = Final[[t,1]];
Finaltest[[t,4]] = Final[[t,2]] - FinalOrder1[[t,2]];
Final2[[t,2]] = mt*mt + g*PolyL;
Final2[[t,1]] = g;

(*The next four loops store the coefficient results from
FindRoot[] so they can be fed back into the equations to
 test if FindRoot[] did return an accurate result*)
For [r=1, r<Alength[[1]]+1, r++,
        Atest[[r]] = ResultFR[[r]];
];

For [r=1, r<A2length[[1]]+1, r++,
        A2test[[r]] = ResultFR[[Alength[[1]]+r]];
];

For [n=1, n<w+1, n++,
        For [s=1, s<w+1 , s++,
                BLMtest[[n,s]] = Atest[[n]]*Atest[[s]];
                For [m=1, m<w+1 , m++,
                        BLM2test[[n,s,m]] = Atest[[n]]*Atest[[s]]
                        *Atest[[m]];
                ]
        ]
];

For [s=1, s<w+1 , s++,
```

```
        For [n=1, n<w2+1, n++,
                For [m=1, m<w2+1 , m++,
                        BLMmix21test[[s,m]] = Atest[[s]]
                        *A2test[[m]];
                ]
        ]
];

(*Makes a (k*k)x1 vector to be multiplied by the
 matrices*)
Btest = Flatten[BLMtest];
B2test = Flatten[BLM2test];
Bmix21test = Flatten[BLMmix21test];

Equationtest = g*PTloop + PTu.Atest+ g*PTloopT.Btest/2
 + g*PTTloop.Btest/2 + mt^2*TP.Atest - g*TPT.Btest
 + 2*g*PT2loop.A2test;

Equation2test = 2*g*P1T.Atest + 2*mt*mt*PxT2.A2test
+ 4*PuT2.A2test + PTTu.Btest + PTuT.Btest + mt*mt*PxTT.Btest
+ g*PTloopT2.Bmix21test  + 2*g*PTT2loop.Bmix21test
 + 2*g*PT2loopT.Bmix21test + g*PT2Tloop.Bmix21test
+ g*PTTTloop.B2test/3 + g*PTTloopT.B2test/3
+ g*PTloopTT.B2test/3 - 2*g*TP1.Atest - 2*mt*mt*TPxT.Btest
 - 2*PTTu.Btest - 4*g*TPT2.Bmix21test- g*TPTTloop.B2test
- g*TPTloopT.B2test - 2*mt*mt*T2Psub.A2test
- 2*g*T2PT.Bmix21test + mt*mt*TTP.Btest + g*TTPT.B2test;

Finaltest[[t,3]]  = Total[ N[Join[Equationtest,
Equation2test]]];

t++;
```

```
]
(*Prints the results and makes plots*)
MatrixForm[Finaltest]
MatrixForm[Final2]

ListPlot[FinalOrder1]
ListPlot[Final]
ListPlot[Re[Final2]]

MatrixForm[Final2Multi]

For [r=1, r<ResLength[[1]]+1, r++,
solvePlots[[r]]=ListPlot[Re[Thread[{Final2Multi[[All,1]],
Final2Multi[[All,r+1]]}]]];
];

Final2MultiCoef = Delete[Final2MultiCoef,1];
Final2MultiCoef = Delete[Final2MultiCoef,1];

(* ::Package:: *)

(*CoefPlots.nb takes the coefficients given from Solve[] in
 LightFinalMulti2.wl and plots them. Then only one coefficient
is printed for study.*)

(*Initializes the list that will hold all the plots*)
CoefPlots = A3;

(*This loop runs for the number of coefficients which is
 determined in LightFinalMulti2*)
For [r=1, r<Coeffs[[1]]+1, r++,
CoefPlots[[r]]=ListPlot[Re[Thread[{Final2MultiCoef[[All,1]],
```

```
Final2MultiCoef[[All,r+1]]}]],PlotRange -> {{0, 1}, {-.5, .5}},
 AxesOrigin -> {0, 0}];
];

(*Prints the graph for a_0*)
CoefPlots[[1]]

(* ::Package:: *)

(*PlotSolve.wl takes the plots created from Solve[] and
FinalLightMulti2.wl and prints them to be studied.*)
solvePlots = Table[u ,{u,ResLength[[1]]+1}]

(*A loop that goes through every result and plots them.
 Then, stores the plots onto a single variable solvePlots*)
For [r=1, r<ResLength[[1]]+2, r++,
solvePlots[[r]]=ListPlot[Re[Thread[{Final2Multi[[All,1]],
Final2Multi[[All,r+1]]}]]];
];

(*First prints all the plots seperately*)
solvePlots
ResLength

(*Prints all the plots onto the same graph then does this
again on a specific range.*)
Show[solvePlots]
Show[solvePlots, PlotRange -> {{0, 20}, {-1, 100}},
AxesOrigin -> {0, 0}]
```