Searching for New Physics with B_S Meson Decay

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Science in
Physics from the College of William and Mary in Virginia,

by

Noah G. Shanton

Christopher Monahan

Advisor: Prof. Christopher J. Monahan

[illegible]

Chancellor Professor of Physics David Armstrong

Williamsburg, Virginia

Contents

Acknowledgments

There are so many people who I would like to thank that it is inevitable that I will fail to mention a few, but I will do my best to acknowledge as many as possible. To start, I would like to thank Professor Monahan for allowing me to work with him on such short notice and for being an amazing mentor and guiding me through this entire process.

Next, I would like to thank Mom and Dad for always supporting me in my endeavors and encouraging my curiosity from a young age. Ever since you gave me a toy model of the solar system that would repeat the same facts about each planet over and over again I was hooked on space and science as a whole. Thank you for being there for me through the good and the bad, and always being available for me to talk. Also, thanks to my brother Connor for keeping me sane and forcing me to have more fun my senior year.

To my roommates Jared, Luke, and Matt, thank you for dealing with my consistent insanity while trying to create a cohesive project. Special shout-out to Matt for attempting to help me with my code. To Rhiannon: thank you for letting me complain about coding, practice giving these presentations and giving me a non-physics perspective, as well as helping me write these acknowledgements.

Finally, a huge thanks to all of my peers in the physics department, and especially the 30 or so who I have had every single physics class with going all the way back to 101H fall of freshman year. We all have grown so close since we came to college, and

I am so thankful that we are such a close-knit major. Now, specifically to Caitlin, Mika, and Seth: the long days and late nights working through the endless problem sets and studying for every midterm and exam together are some of the memories from my time here that I will look back on most fondly. Were they the most fun? No. But they sure were memorable, and I wouldn't trade them for the world. I know I wouldn't have survived this major without the help and support of you all, as well as the rest of our little physics community.

List of Figures

Abstract

In our universe, there exists unequal amounts of matter and antimatter; the origin of this asymmetry requires Charge-Parity (CP) violation. However, the standard model of particle physics, which describes the electromagnetic, strong, and weak fundamental forces and classifies elementary particles, does not account for enough CP violation to explain this asymmetry. To find new sources of CP violation, we want to search for forces or particles that are not captured by the standard model. One such method is to test the unitarity of the Cabibbo-Kobayashi-Maskawa (CKM) matrix, which describes how quarks mix under the weak force. If the CKM matrix can be shown to not be unitary by comparing experimental measurements and theoretical predictions of the CKM matrix elements, then this implies the existence of a fourth generation of quarks. Quarks bind together in groups of two to form mesons, which are governed by the strong nuclear force, and require lattice QCD to study theoretically with precision. I fit Monte Carlo simulated lattice QCD data for the form factors of the $B_s \to K$ and $B_s \to D_s$ decays, and found that the $B_s \to K$ decay required three terms to get the best possible fit, while the $B_s \to D_s$ decay required two terms. Unfortunately, the "best" χ^2 value of any of the fits was 87, pointing to inaccuracies in the fits.

Chapter 1

Introduction

In short, the goal for this project is to find new physics by showing that there is an additional generation of quarks not captured by the current standard model of particle physics. Before we can do so, however, we need to take a step back and take a deeper look at what makes up our universe. It is reasonable to expect that an equal amount of matter and antimatter was created in our universe during the Big Bang. However, if this were true, the matter and antimatter would have mostly annihilated, leaving a radiation-dominated universe. Given that I am typing this thesis on a computer, which is made up of matter, there appears to be more matter than antimatter.

There are three conditions, known as Sakharov conditions, which must be met in order for this asymmetry between matter and antimatter to occur [1]. One of these Sakharov conditions is Charge-Parity (CP) violation [1]. The standard model of particle physics, which describes the strong, weak, and electromagnetic fundamental forces, as well as classifies the elementary particles, does not account for enough CP violation for this asymmetry to occur [2]. So, we need to look for new sources of CP violation, such as particles or forces that are not captured by the standard model.

One method of searching for CP violation is examining the Cabibbo-Kobayashi-Maskawa (CKM) matrix. I will discuss the CKM matrix in depth in chapter 2, but for now, the CKM matrix describes how quarks mix under weak interactions. The weak force mixes up quarks, which can annihilate. In the standard model, the CKM matrix is 3x3 and unitary [3]. If it can be shown to not be unitary, this would imply that the matrix is actually 4x4, which in turn implies the existence of a 4th generation of quarks. Unfortunately, this would not tell us any information about the new generation, just that it exists.

Quark interactions in the standard model are governed by the strong nuclear force, as described by quantum chromodynamics (QCD) [2]. One method of evaluating quantum systems is perturbation theory. Perturbation theory is a systematic approach for obtaining approximate solutions. QCD is strongly coupled at low energy scales, meaning that the coupling coefficient diverges. Unfortunately, this means that perturbation theory is not convergent at low energy scales, and so cannot be used to solve the system analytically in this case. Instead, we need to use lattice QCD, which is a numerical approach to QCD, to evaluate the system. I will discuss lattice QCD in detail in chapter 2.

Circling back to the CKM matrix, each matrix element represents the relative probability that a quark decays from one flavour to another. So, to test the unitarity of the matrix, we can take measurements of certain decays, and compare these experimental measurements to the theoretical predictions based on the standard model. I explain this process in chapter 3.

In chapter 4, I discuss the results of comparing a path integral estimation to an exact calculation from quantum mechanics. Also, I implement a Metropolis Monte Carlo algorithm in order to find the excitation energy of a one dimensional harmonic oscillator, and perform a stability analysis on the data. Finally, I fit the form factor data for the $B_s \to K$ and $B_s \to D_s$ decays to equations 2.6 and 2.7. In chapter 5, I summarize the findings of this project, as well as look into potential opportunities for future work extending on this research.

Chapter 2

Theoretical Background

In this chapter, I outline four key ideas that are essential and need to be explained in more detail: the Cabibbo-Kobayashi-Maskawa (CKM) Matrix, lattice quantum chromodynamics (QCD), the vector and scalar form factors, and Monte Carlo Integration.

2.1 Cabibbo-Kobayashi-Maskawa (CKM) Matrix

The CKM matrix, also called the quark-mixing matrix, describes how quarks mix under weak interactions. Quarks are the eigenstates of the strong force, however, the eigenstates of the weak force are linear combinations of strong interaction eigenstates, rather than the strong eigenstates themselves. Under the standard model, the CKM matrix is 3x3 and unitary and can be written as follows:

$$V_{CKM} = \begin{pmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{pmatrix}, \tag{2.1}$$

where the subscript s refers to a strange quark, c refers to a charm quark, u refers to an up quark, d refers to a down quark, t refers to a top quark, and b refers to a bottom quark [3]. Each matrix element is the square root of the probability that a quark decays from the second flavour of quark in the subscript to the first [3]. So, for example, V_{ud} refers to the probability that a down quark decays into an up quark.

To test the unitarity of the CKM matrix, we need to have a large sample size of CKM matrix element measurements. However, we will not actually be measuring them directly. Instead, we can use the connection between QCD, the CKM matrix, and experimental data, along with Equations 2.2 and 2.3 to calculate the CKM matrix elements.

In this project, there are two decay rate equations that the CKM matrix is vital to. The first of these is

$$\frac{d\Gamma(B_s \to K l \nu_l)}{d\Omega} \propto |V_{ub}|^2 \left(c_+|f_+|^2 + c_0|f_0|^2\right), \tag{2.2}$$

where Γ is the decay rate, Ω is the solid angle, B_s is a B_s meson, K is a kaon, l is a lepton, ν_l is a neutrino, V_{ub} is an element of the CKM matrix, c_+ and c_- are kinematic factors, f_+ is a vector form factor, and f_0 is a scalar form factor [4]. A meson consists of two quarks bound tightly together by the weak force, as opposed to baryons, which consist of three quarks [2]. A kaon is a type of meson which contains either an up or down quark, along with strange quark; either the up/down quark or the strange quark is an anti-quark, but not both [5]. The B_s meson, on the other hand, is the strange-B meson, and contains a strange quark and an anti-bottom quark or a bottom quark and an anti-strange quark [5]. The form factors f_+ and f_0 parameterize the QCD contributions, and come from lattice QCD. The other equation is similar, but for the $B_s \to D_s$ decay:

$$\frac{d\Gamma(B_s \to D_s l \nu_l)}{d\Omega} \propto |V_{cb}|^2 \left(c_+|f_+|^2 + c_0|f_0|^2\right), \tag{2.3}$$

where the kaon is replaced by the D_s meson, which is a meson containing either a charm and anti-strange quark, or a strange quark and an anti-charm quark, and the CKM matrix element is changed to V_{cb} [5]. The reason why I do not specify the exact pairing of quarks and anti-quarks each of the B_s, K, and D_s mesons consist of is because in lattice QCD, there is no electromagnetism, and so we cannot distinguish quarks and anti-quarks based on charge.

2.2 Lattice Quantum Chromodynamics

Before discussing lattice QCD, first we need to define what QCD is. Quantum chromodynamics (QCD) describes strong interactions between quarks [2]. The Lagrangian for QCD is

$$L_{QCD} = \overline{\psi}_q^i (i\gamma^\mu)(D_\mu)_{ij}\psi_q^j - m_q\overline{\psi}_q^i\psi_q^i - \frac{1}{4}F^a_{\mu\nu}F^{a\mu\nu}. \tag{2.4}$$

Here, $\overline{\psi}_q^i$ is a quark field with colour index i, $\psi_q = (\psi_{qR}, \psi_{qG}, \psi_{qB})^T$,$\gamma^\mu$ is a Dirac matrix, m_q is the mass of the quark, $F^a_{\mu\nu}$ is the gluon field strength tensor, which describes how gluons propagate and interact with themselves, for a gluon with color index a, and D_μ is the covariant derivative, which ensures the Lagrangian is invariant under gauge transformations, in QCD [6]. Lattice QCD is a method of analyzing quantum systems described by QCD, without using perturbation theory. Essentially, it takes a continuous theory, puts it in a discrete box, and then evaluates it numerically. Spacetime is continuous, and lattice QCD is defined by taking a small finite value of spacetime and make it discrete. This allows us to then make calculations using a computer, before getting rid of the lattice spacing and making it continuous again. It often utilizes Monte-Carlo simulations in order to generate a pseudo-random data set. The lattice itself can be considered both a physical thing and a math “tool”, depending on one’s point of view. From an experimental viewpoint, it is a math trick, but it is also similar to the rectangular lattice seen in atoms.

2.3 $f_+(\vec{p})$ and $f_0(\vec{p})$ Form Factors

The $f_+(\vec{p})$ and $f_0(\vec{p})$ form factors parameterize how particles interact with external forces and are determined by lattice QCD. They also have a dependence on one kinematic variable. Usually, this is either the momentum transfer, q^2 (technically this is the square of the momentum transfer, but to be more concise I chose to drop of the 'squared' when referring to q^2 for the remainder of this paper), or the energy of the mesonic decay, E_{X_s} [4]. The form factors are defined by the equation

$$\begin{aligned}\langle X_s(p_{X_s})|V^\mu|B_s(p_{X_s})\rangle = f_0^{(X_s)}(q^2)\frac{M_{B_s}^2 - M_{X_s}^2}{q^2}q^\mu \\ +f_+^{(X_s)}(q^2)\left[p_{B_s}^\mu + p_{X_s}^\mu - \frac{M_{B_s}^2 + M_{X_s}^2}{q^2}q^\mu\right],\end{aligned} \tag{2.5}$$

where $X_s\,(p_{X_s})$ is either the B_s or K meson with momentum p_{X_s}, V^μ is a vector current that connects a bottom quark with an up quark if $X_s = K$ or a charm quark if $X_s = D_s$, f_0 is the scalar form factor, and f_+ is the vector form factor. The mass of the B_s meson is represented by M_{B_s}, M_{X_s} is the mass of either the kaon or the D_s meson, q^2 is the momentum transfer, p^μ is the momentum four-vector, and q^μ is the momentum transfer four-vector [4]. However, the form factors can instead be expressed as an expansion in the variable z, which is defined in equation 2.8. For the scalar form factors, the equation is

$$f_0^{(X_s)}(q^2(z)) = \frac{1}{P_0^{X_s}}\sum_{j=0}^{J-1} a_j^{(0,X_s)} z^j, \tag{2.6}$$

while for the vector form factors the equation looks like

$$f_+^{(X_s)}(q^2(z)) = \frac{1}{P_+^{X_s}}\sum_{j=0}^{J-1} a_j^{(+,X_s)}\left[z^j - (-1)^{j-J}\frac{j}{J}z^J\right], \tag{2.7}$$

where $P_{0,+}^{X_s}$ is the Blaschke factor, a is an expansion coefficient, and z is given by the equation

$$z = \frac{\sqrt{t_+ - q^2} - \sqrt{t_+ - t_0}}{\sqrt{t_+ - q^2} + \sqrt{t_+ - t_0}}, \tag{2.8}$$

with $t_+ = (M_{B_s} + M_{X_s})^2$ and $t_0 = (M_{B_s} + M_{X_s})(\sqrt{M_{B_s}} - \sqrt{M_{X_s}})^2$ [4]. The Blaschke factors, given in the equation

$$P_{0,+}^{(X_s)}(q^2) = \left(1 - \frac{q^2}{\left(M_{0,+}^{(X_s)}\right)^2}\right), \tag{2.9}$$

account for the effects from unphysical masses [4]. The values for $M_{0,+}^{X_s}$ are listed in [4].

2.4 Monte Carlo Integration

One method of generating probability distributions for lattice QCD simulations is through Monte Carlo integration. It is possible to do this is to use `vegas`, which is a code package that deals with multidimensional integration [7]. In this project, I make use of the Metropolis Monte Carlo algorithm. This process starts with an arbitrary path and alters it by randomizing each of the values at each individual lattice site [7]. In order to randomize a value at a certain lattice site, first an algorithm must generate a random number that has a uniform probability distribution within a constant range [7]. Next, it must add the randomly generated number to the value at the lattice site and compute the change in the action that results (the action is $S_{lat}[x]$ as defined in equation 3.3). If the action decreases, then it keeps this new value and goes to the next site. If the action increases, then it generates a random number uniformly distributed between 0 and 1. If the exponential function of the negative of the difference from earlier is greater than this new number between 0 and 1, then it keeps this new value for the current lattice space, and moves on to the next one [7]. The end result of these steps is a random path different from the old path [7]. This process can be repeated to create as many random unique paths as desired.

In order to actually calculate an estimation of a function along a path, an algorithm must first initialize the path [7]. One way of doing this is to set the value at each lattice site to zero. Next, the path must be thermalized before being updated so it can compute and save the function [7]. This step is repeated numerous times. Then these saved values for the function are averaged. This average is the Monte Carlo estimate of the function [7].

Chapter 3

Numerical Methods

This project focuses on utilizing lattice QCD methods of numerically analyzing a system, which I described in more detail in section 2.2. Here, I detail the numerical methods I used to analyze the lattice QCD data.

To start, we need to look at where exactly the data comes from. The data comes from Monte Carlo integration within the `vegas` Python package. `Vegas` utilizes Gaussian variables in order to perform its functions. Gaussian variables are represented by a data type called gvar.GVar [8]. This data type stores both the mean and standard deviation of the Gaussian random variables that they represent [8]. In large part, the data is based on path integrals.

3.1 Path Integral Estimation

In order to better understand how both `vegas` and Monte Carlo integration works, I first used them to estimate the quantum mechanical propagator $\langle x|e^{-\hat{H}t}|x\rangle$ by evaluating a path integral. In this case, I used the Hamiltonian for a one-dimensional harmonic oscillator which, if we recall from PHYS 313, is

$$\hat{H} = \frac{1}{2m}\left[\hat{p}^2 + (m\omega x)^2\right], \tag{3.1}$$

where m is the mass of the particle, $\hat{p}$ is the momentum operator, ω is the angular frequency of oscillation, and x is the particle's position [9]. I then compared the results to the exact value, which is obtained from standard quantum mechanics. The path integral is

$$\langle x|e^{-\hat{H}t}|x\rangle \approx A \int_{-\infty}^{\infty} dx_1 \ldots dx_7 e^{-S_{lat}[x]}, \tag{3.2}$$

where A is the normalization constant and $S_{lat}[x]$ is defined as

$$S_{lat}[x] = \sum_{j=0}^{7} \left[\frac{m}{2a} \left(x_{j+1} - x_j \right)^2 + aV(x_j) \right], \tag{3.3}$$

with $m = 1$, $V(x_j) = \frac{x^2}{2}$, and $a = \frac{1}{2}$ [7]. The results are plotted in figure 4.1, and the code can be found in Appendix A.3. Next, the sampling itself is based on the Markov chain Monte Carlo class of algorithms. One of these algorithms is the Metropolis Monte Carlo algorithm.

3.2 Metropolis Monte Carlo

My next step in learning to better understand `vegas` was to implement a Metropolis Monte Carlo algorithm in order to find the excitation energy, ΔE_n, of a one-dimensional harmonic oscillator. To do so, I first calculated

$$G(t) = \frac{1}{N} \sum_j \langle\langle x(t_j + t) x(t_j) \rangle\rangle \tag{3.4}$$

for $t = 0, a, 2a, ..., (N-1)a$, with $N = 20$ lattice slices and a lattice spacing of $a = \frac{1}{2}$ [7]. Then, from these calculated G values, I calculated the excitation energy using the equation

$$\Delta E \equiv \log(G(t)/G(t+a))/a. \tag{3.5}$$

To calculate the error on ΔE, I implemented a jackknife error algorithm [10], which can be found in Appendix A.2. I will explain the jackknife error estimation method in more detail in section 3.3. I will also discuss the results of this approach to calculating the excitation energy in section 4.2.

Going further, I also calculated the excitation energy by fitting the G values I calculated to an exponential of the form $G(t) = ae^{-bt}$ using `lsqfit`. This is feasible because for large t,

$$G(t) \rightarrow | < E_0|\tilde{x}|E_1 > |^2 e^{-(E_1-E_0)t}, \tag{3.6}$$

as given in [7]. Thus, in the exponential fit, $b = \Delta E$ and a is a constant. The Python package `lsqfit` does least-squares fitting of noisy data by non-linear multi-dimensional functions [11]. It also utilizes Bayesian priors for the fit parameters and is heavily reliant on the `gvar` Python package I previously mentioned [11]. I will discuss the results of the fitting in section 4.2.

I also employ Monte Carlo simulated data to find the form factors described in section 2.3. Looking at Equation 2.2, $B_s \rightarrow Kl\nu$ comes from experiment, $|V_{ub}|$ comes from the CKM matrix, and the form factors come from lattice QCD. If we know two of these, then we can calculate the other. In my case, I measure the scalar and vector form factors, f_0 and f_+ for both the $B_s \rightarrow K$ and $B_s \rightarrow D_s$ decays. In order to do this, we need precise theoretical predictions of CKM matrix elements from the standard model. We also need the $B_s \rightarrow K$ and $B_s \rightarrow D_s$ decays, which come from experiment. And the f_0 and f_+ form factors come from the Monte Carlo simulated data. I fit this form factor data to equations 2.6 and 2.7; these fits will be discussed in section 4.3.

3.3 Jackknife Resampling

The jackknife resampling method estimates the error in measurements by eliminating potential bias that arises in individual data points [10]. Essentially, a Jackknife algorithm takes a set of x values, removes the first *value*, then calculates the average of the remaining x values. It then calculates $f(x)$ using this average. Next, it removes the second x value, while keeping the first that was previously removed, and goes through the same calculations as before. This process is repeated until every x value has been removed once [10].

The jackknife averages are defined by the equation

$$x_i^J \equiv \frac{1}{N-1} \sum_{j \neq i} x_j, \tag{3.7}$$

where x_i^J are the jackknife averages, N is the number of x values in the original sample, and x_j is the sum of all the x values except for x_i [10]. Thus, the jackknife estimation of $f(X)$ calculated as

$$f(X) \simeq \overline{f(x^J)} \equiv \frac{1}{N} \sum_{i=1}^{N} f(x_i^J), \tag{3.8}$$

where $\overline{f(x^J)}$ is the average of the $f(x_i^J)$ values [10]. The uncertainty in this estimation of $f(X)$ can be calculated using the equation

$$\sigma_{f(\bar{x})} = \left(\sqrt{N-1}\right) \sigma_{f(x_J)}, \tag{3.9}$$

where $\sigma_{f(\bar{x})}$ is the uncertainty [10]. $\sigma_{f(x_J)}$ is defined by the equation

$$\sigma_{f^J}^2 = \overline{\left(f\left(x^J\right)\right)^2} - \left(\overline{f\left(x^J\right)}\right)^2. \tag{3.10}$$

In addition to applying jackknife resampling to estimate the error, I also utilized a couple of data analysis techniques. One is stability analysis and the other is known as folding.

3.4 Stability Analysis and Folding

Stability analysis, in reference to fitting a function, refers to limiting certain aspects of the data in order to see if these changes result in a better fit. Some of these aspects that can be played with include the range of the data included in the fit and the *a priori* estimates of the priors. I will discuss the results of the stability analysis for the $G(t) = ae^{-bt}$ exponential fit in section 4.2.

Folding, on the other hand, refers to averaging symmetric data points in an effort to reduce error and make the data more accurate by increasing the relevant sample size of the data. This technique applies only to time-symmetric data. To help visualize what exactly "symmetric data points" refers to, one can think of a parabola of the form $f(x) = x^2$. In this example, the data points at $x = 1$ and $x = -1$ are symmetric, as are $x = 2$ and $x = -2$, $x = 3$ and $x = -3$, and so on. As part of my analysis of the Metropolis Monte Carlo data I fit to $G(t) = ae^{-bt}$, I folded the $G(t)$ data values and plotted it alongside the fit function. I discuss these results in section 4.2.

Chapter 4

Results

In this section, I first detail the results of comparing a path integral to the exact calculation of the values. Next, I discuss the data and plots related to the excitation energy extrapolation. Finally, I explain the scalar and vector form factor data and plots of the data fit to equations 2.7 and 2.6.

4.1 Path Integral vs Exact Calculation

First, I calculated $\langle x|e^{-Ht}|x\rangle$ by evaluating a path integral and compared the resulting value to the value obtained from standard quantum mechanics [7]. The equation for the path integral can be found in equation 3.2. The results are plotted in figure 4.1, and my code for this exercise can be found in Appendix A.3. Looking at the plot, most of the data points calculated by the path integral lie along the curve of the exact calculation. There are error bars on the path integral data points, they are just very small.

4.2 Metropolis Monte Carlo Data

Next, I implemented a Metropolis Monte Carlo algorithm for a one-dimensional harmonic oscillator in order to find the excitation energy, ΔE_n. To do so, I first used equation 3.4 to calculate $G(t)$ values for for $t = 0, a, 2a, ...(N-1)a$, with $N = 20$

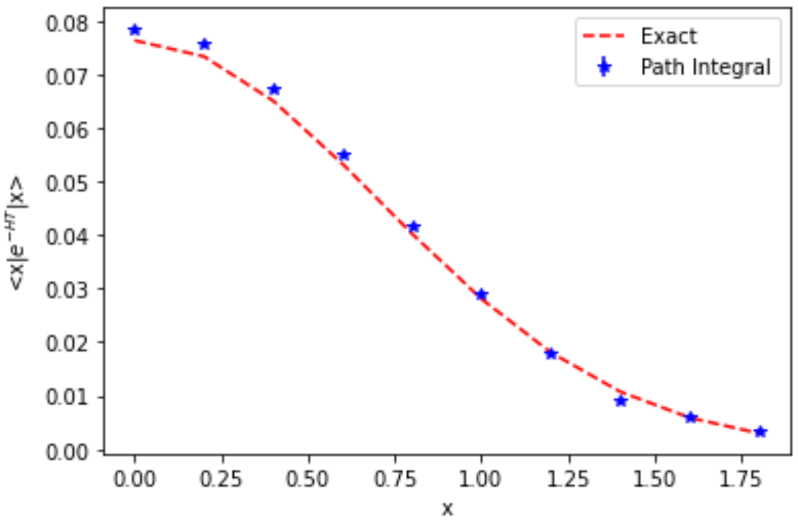

Figure 4.1: Plot comparing the calculation of $\langle x|e^{-Ht}|x\rangle$ using exact methods (multiplying by a known constant) and using a path integral estimation.

lattice slices and lattice spacing of $a = \frac{1}{2}$ [7]. Then, from these calculated G values, I calculated the excitation energy using equation 3.5.

To calculate the error on ΔE, I implemented a jackknife error algorithm, as described in section 3.3 which can be found in Appendix A.2. The results of this approach to calculating the excitation energy can be found in figure 4.2, which show a decent approximation from $t = 0$ to $t = 1.0$, after which the data appears to be less accurate. There is also very little error up to $t = 2.0$, at which point the error begins to increase dramatically with each consecutive data point. Interestingly, the only data point which appears to not reach the expected value of $\Delta E = 1.0$ within the error limits is the data point at $t = 1.5$.

In addition to using equations to calculate the excitation energy, I also used the Python package `lsqfit` to fit my calculated $G(t)$ data to an exponential equation matching the form of equation 3.6. To attempt to get a more accurate fit, I first folded my data of G values. I then recalculated the error on each data point. Next, I

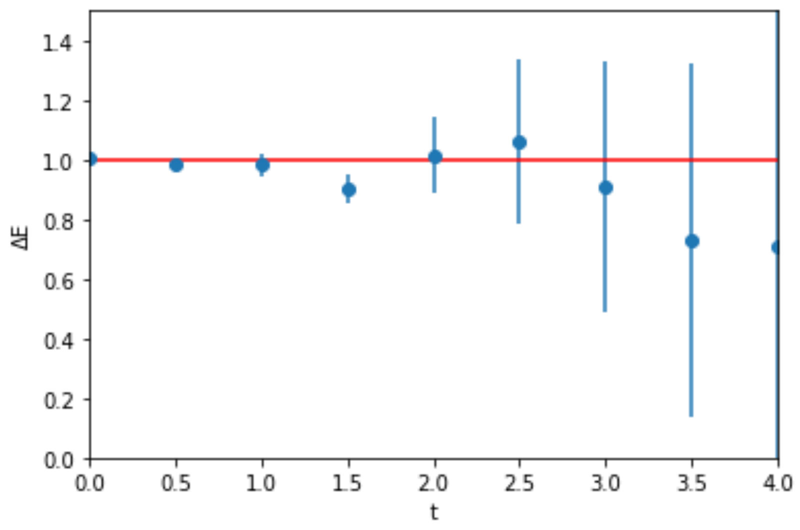

Figure 4.2: Plot of the excitation energy at varying times t. The excitation energy was calculated using the equation $\Delta E \equiv \log(G(t)/G(t+a)/a$. The horizontal red line at $\Delta E = 1.0$ is the expected value. The vertical blue lines are the error estimations on each point. These errors were calculated using the jackknife method.

did the fit using `lsqfit`. I then performed a stability analysis in which I altered the ranges of t included in various fits. The results of this stability analysis can be found in figure 4.3. Looking at the plot, including the full range of t appears to result in a value closest to the expected value, and the error increases as the range of t gets more restricted. One reason for this is that limiting the range of t decreases the sample size. Another reason is that I cut out the initial values of t, which were more precise. Both of these explain why the error increased in the way that it did. Furthermore, my folded data for $G(t)$ can be found in figure 4.4, along with a fit for $G(t) = ae^{-bt}$, with $a = 0.49$ and $b = 0.99$.

4.3 Form Factor Data

Next, we use the various methods and tools mentioned in Chapter 3 to generate a fit to approximate the form factors associated with the $B_s \to Kl\nu$ and $B_s \to D_s l\nu$

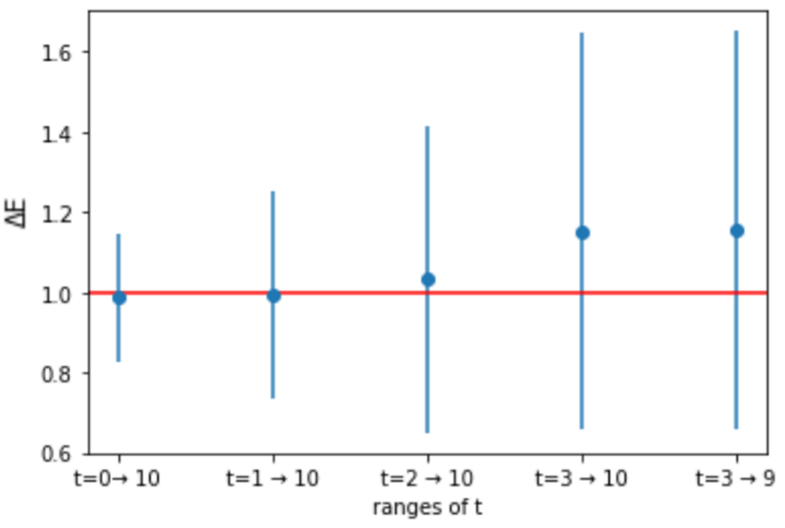

Figure 4.3: This plot is a stability analysis for finding $\Delta E(t)$ done by limiting the range of t that I included in the fit. The horizontal line at $\Delta E(t) = 1.0$ is the expected value for $\Delta E(t)$, and the vertical bars are the error estimations on the $\Delta E(t)$.

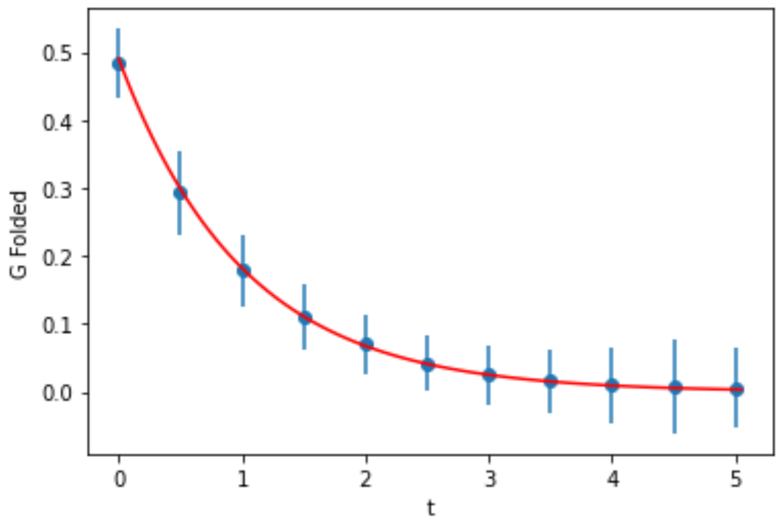

Figure 4.4: This is a plot of my folded data of $G(t)$ (the blue data points) fit to the equation $G(t) = ae^{-bt}$, with $a = 0.49$ and $b = 0.99$, with a χ^2 per degree of freedom of 0.039.

decays. The fit for the scalar form factors are be in the form of equation 2.6, and the vector form factors are be in the form of equation 2.7.

The first step of generating the fit was calculating z using equation 2.8. Although, to do so, first the momentum transfer q^2 must be calculated. The momentum transfer is defined as $q^\mu = p^\mu_{B_s} - p^\mu_{X_s}$ [4]. Thus, by doing some simplifying we can use the equation $q^2 = M^2_{B_s} + M^2_{X_s} - 2M_{B_s}E_{X_s}$ to calculate q^2, where E_{X_s} is the energy of either the kaon or the D_s meson.

Now, we fit the data. For each type of decay, I input the energy values as the independent variable and the form factors given in tables IV and V of [4] as the dependent variable. Instead of only fitting each to a single function, I did fits to a constant function, a linear function, a quadratic function, and a cubic function. For example, for the scalar form factors, this looks like $a_0 + a_1 z + a_2 z^2 + a_3 z^3$.

4.3.1 K decay data

To start, let's examine the data for the scalar form factors for the $B_s \to K$ decay. The first fit I did was the most simple, just $f_0 = a_0$, where $a_0 = 0.57721 \pm 0.00060$.

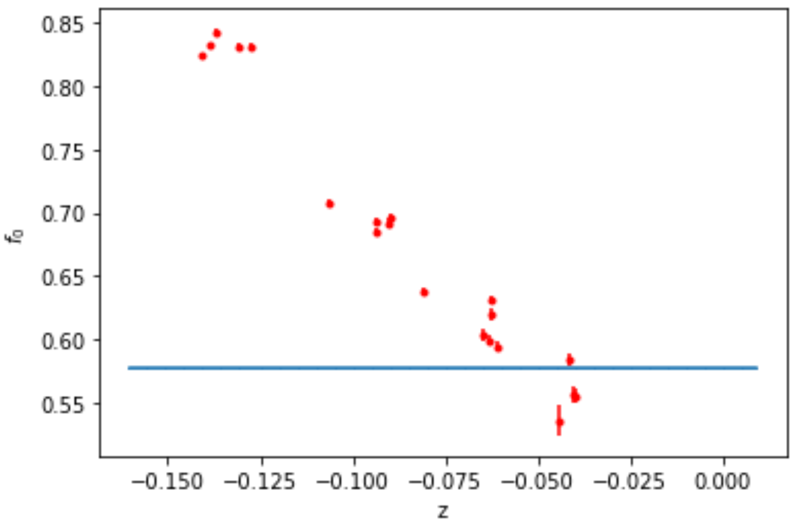

Figure 4.5: This is a plot of the expected scalar form factors for the $B_s \to K$ decay (the red dots) compared with the curve fit to the equation $f_0 = a_0$, where $a_0 = 0.57721 \pm 0.00060$. The fit has a χ^2 per degree of freedom of 850.

Even by just taking a quick glance at figure 4.5, it is quickly obvious that this fit is terrible. It simply does not include enough terms to accurately fit to the given data. However, we would expect the blue fit line to be closer to the average of the form factor data (slightly above 0.65, as opposed to between 0.55 and 0.60 as it is currently). This hints that there may be some issues with either the code or `lsqfit`, or both. These observations are supported by its χ^2 per degree of freedom of 850.

Next, I fit the data to $f_0 = a_0 + a_1 z$, where $a_0 = 0.3740 \pm 0.0020$ and $a_1 = -1.951 \pm 0.018$. Looking at figure 4.6, at first glance this fit appears to be better than the previous one. The fit line is closer to the data points, however, it still does not appear to be very good since none of the data points are actually along the trendline. This is again supported by the fact that this fit has a χ^2 per degree of freedom of 290: better than the previous fit, but still not great. Again, adding more terms will probably result in a better fit.

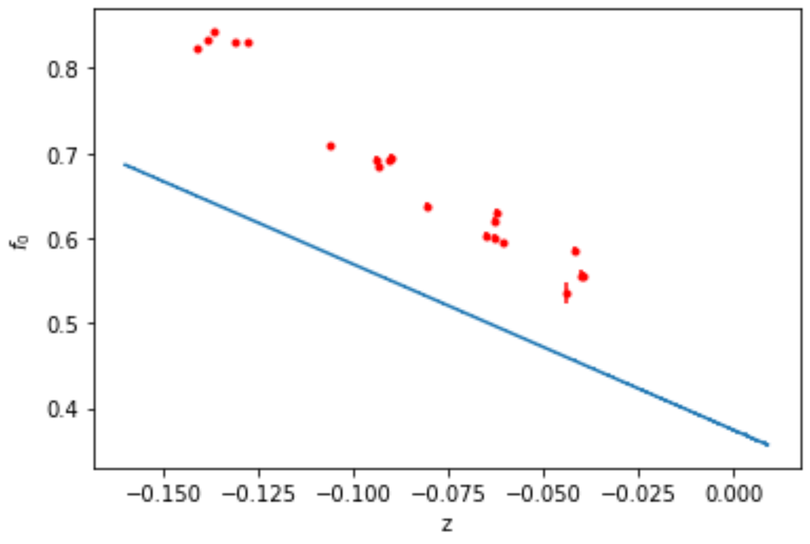

Figure 4.6: This is a plot of the expected scalar form factors for the $B_s \to K$ decay (the red dots) compared with the curve fit to the equation $f_0 = a_0 + a_1 z$, where $a_0 = 0.3740 \pm 0.0020$ and $a_1 = -1.951 \pm 0.018$. The fit has a χ^2 per degree of freedom of 290.

The next fit I did was to $f_0 = a_0 + a_1 z + a_2 z^2$, where $a_0 = 0.4549 \pm 0.0053$, $a_1 = -0.10 \pm 0.11$, and $a_2 = 9.44 \pm 0.57$. Looking at figure 4.7, this fit does not appear to be much better than the previous one. This is supported by the fit's χ^2 per degree of freedom of 280. Since the last two χ^2 per degree of freedom values were similar, I am not sure if adding more terms will change much, especially with the probable issues with the fitting code.

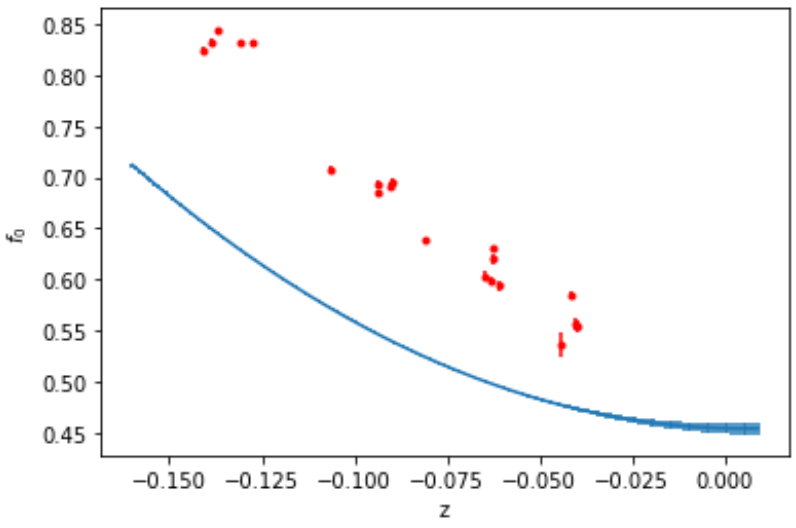

Figure 4.7: This is a plot of the expected scalar form factors for the $B_s \to K$ decay (the red dots) compared with the curve fit to the equation $f_0 = a_0 + a_1 z + a_2 z^2$, where $a_0 = 0.4549 \pm 0.0053$, $a_1 = -0.10 \pm 0.11$, and $a_2 = 9.44 \pm 0.57$. The fit has a χ^2 per degree of freedom of 280.

The last fit I did for the scalar form factors associated with the $B_s \to K$ decay was to the equation $z = a_0 + a_1 z + a_2 z^2 + a_3 z^3$, where 0.4523 ± 0.0055, $a_1 = -0.14 \pm 0.12$, $a_2 = 9.40 \pm 0.62$, and $a_3 = 1.32 \pm 0.98$. Taking a look at figure 4.8, this fit is nearly indistinguishable from the previous quadratic fit. This is supported by the fact that the parameters a_0, a_1, and a_2 are so similar between the two fits, and that both fits have a χ^2 per degree of freedom of 280.

Next come the vector form factors. Unlike the scalar form factors, the vector form

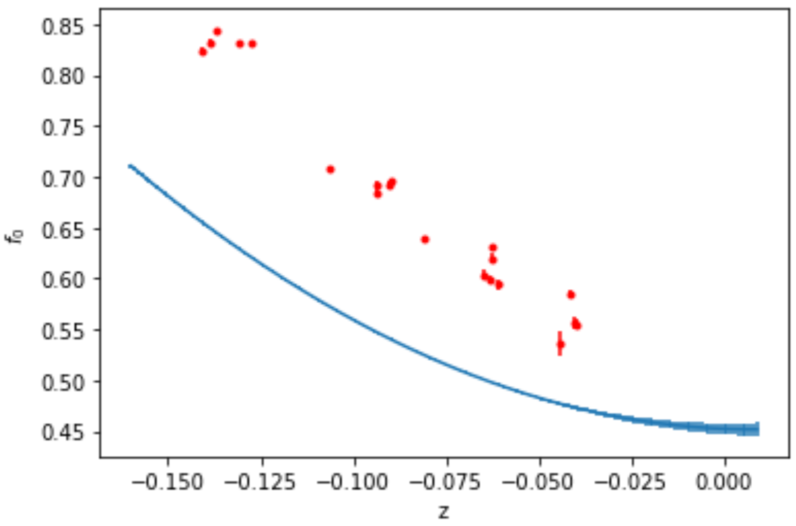

Figure 4.8: This is a plot of the expected scalar form factors for the $B_s \to K$ decay (the red dots) compared with the curve fit to the equation $f_0 = a_0 + a_1 z + a_2 z^2 + a_3 z^3$, where $a_0 = 0.4523 \pm 0.0055$, $a_1 = -0.14 \pm 0.12$, $a_2 = 9.40 \pm 0.62$, and $a_3 = 1.32 \pm 0.98$. The fit has a χ^2 per degree of freedom of 280.

factors are fit to equation 2.7. The first fit I did to just a scalar term was similar to the one for the scalar form factors, and so there is no useful information to glean from it. Thus, I skip to the next fit I did, to the equation $f_+ = a_0 + a_1 \left(z + \frac{1}{2}z^2\right)$, where $a_0 = 0.5511 \pm 0.0094$ and $a_1 = -9.64 \pm 0.13$; this fit has a χ^2 per degree of freedom of 89. Interestingly is the "best" χ^2 per degree of freedom of any fit so far, although it still is not very good. This fit can be seen in figure 4.9

I next fit the data to $f_+ = a_0 + a_1 \left(z - \frac{1}{3}z^3\right) + a_2 \left(z^2 + \frac{2}{3}z^3\right)$, where $a_0 = 0.610 \pm 0.010$, $a_1 = -7.78 \pm 0.19$, and $a_2 = 9.4 \pm 1.1$. This fit has a χ^2 per degree of freedom of 87. Since this value improved, I did another fit up to the a_3 term.

The final fit for the $B_s \to K$ decay is the fit to the equation $f_+ = a_0 + a_1 \left(z + \frac{1}{4}z^4\right) + a_2 \left(z^2 - \frac{2}{4}z^4\right) + a_3 \left(z^3 + \frac{3}{4}z^4\right)$, with $a_0 = 0.623 \pm 0.010$, $a_1 = -7.42 \pm 0.19$, $a_2 = 11.1 \pm 1.1$, and $a_3 = 1.0 \pm 1.0$, with a χ^2 per degree of freedom of 87. This is the same value as the previous fit, so including any more terms probably will not result

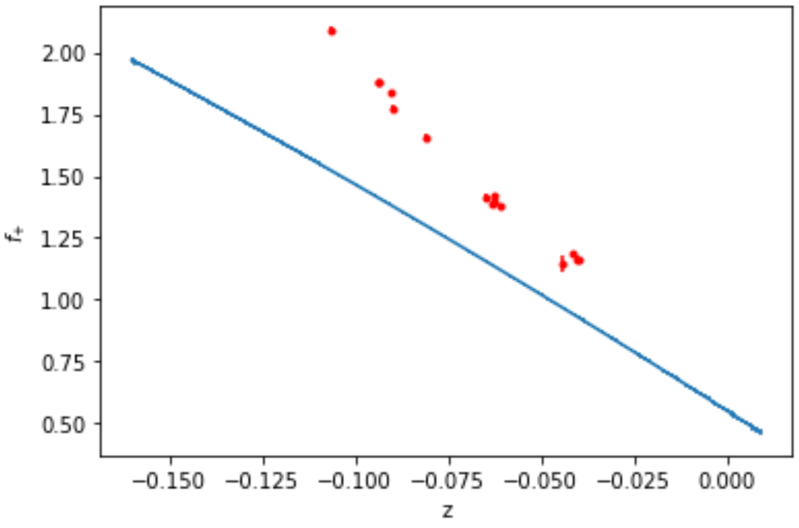

Figure 4.9: This is a plot of the expected vector form factors for the $B_s \to K$ decay (the red dots) compared with the curve fit to the equation $f_+ = a_0 + a_1\left(z + \frac{1}{2}z^2\right)$, where $a_0 = 0.5511 \pm 0.0094$ and $a_1 = -9.64 \pm 0.13$. The fit has a χ^2 per degree of freedom of 89.

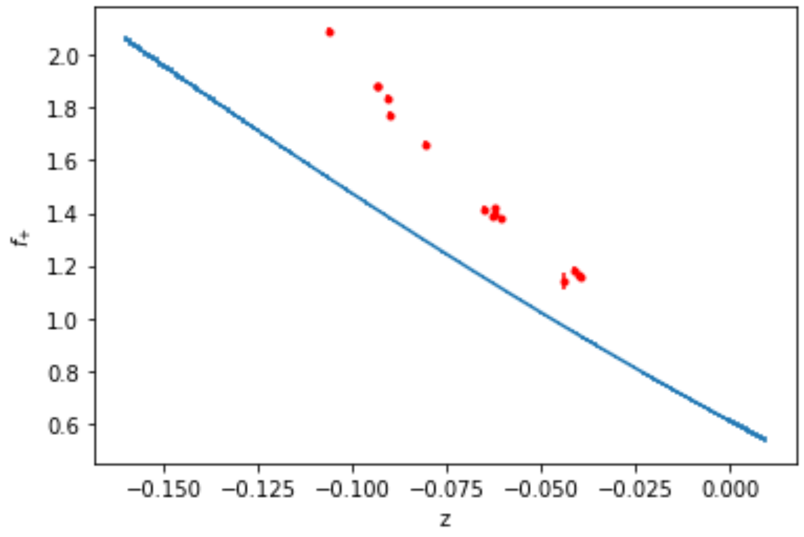

Figure 4.10: This is a plot of the expected vector form factors for the $B_s \to K$ decay (the red dots) compared with the curve fit to the equation $f_+ = a_0 + a_1\left(z - \frac{1}{3}z^3\right) + a_2\left(z^2 + \frac{2}{3}z^3\right)$, where $a_0 = 0.610 \pm 0.010$, $a_1 = -7.78 \pm 0.19$, and $a_2 = 9.4 \pm 1.1$. The fit has a χ^2 per degree of freedom of 87.

in a better fit.

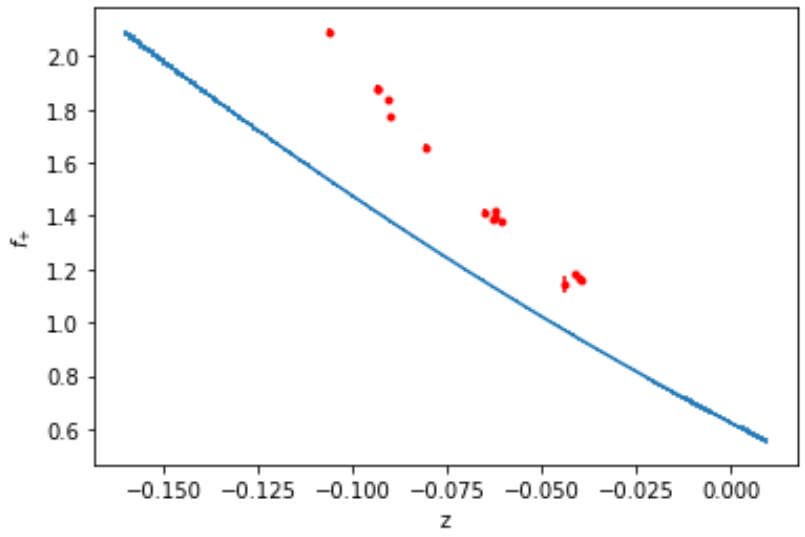

Figure 4.11: This is a plot of the expected vector form factors for the $B_s \to K$ decay (the red dots) compared with the curve fit to the equation $f_+ = a_0 + a_1\left(z + \frac{1}{4}z^4\right) + a_2\left(z^2 - \frac{2}{4}z^4\right) + a_3\left(z^3 + \frac{3}{4}z^4\right)$, where $a_0 = 0.623 \pm 0.010$, $a_1 = -7.42 \pm 0.19$, $a_2 = 11.1 \pm 1.1$, and $a_3 = 1.0 \pm 1.0$. The fit has a χ^2 per degree of freedom of 87.

4.3.2 D_s decay data

Similar to the $B_s \to K$ decay, I began with fitting the data to the function $f_0 = b_0$, where b_0 is a constant (I chose to use b as the parameters for the D_s fits and a for the K fits). However, it was a similarly atrocious fit, and so I will skip to the second fit I did: the fit to the equation $f_0 = b_0 + b_1 z$. Here, $b_0 = 0.7392 \pm 0.0018$ and $b_1 = -1.951 \pm 0.018$, and the fit has a χ^2 per degree of freedom of 440. Interestingly, this fit has a much higher χ^2 per degree of freedom value than the linear fit of the $B_s \to K$ decay did. By adding more terms to the fit, I would assume that the χ^2 per degree of freedom value would decrease similarly to how it did for the $B_s \to K$ decay fits.

The next equation I fit the data to is $f_0 = b_0 + b_1 z + b_2 z^2$, where $b_0 = 0.7403 \pm 0.0019$,

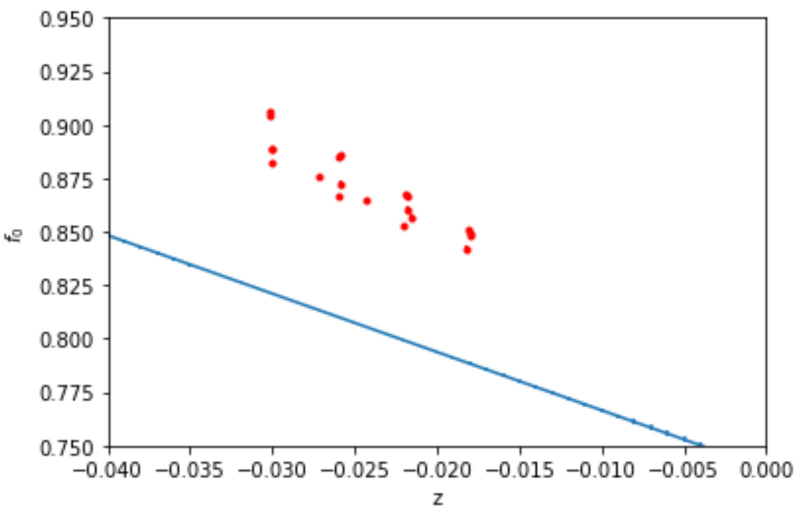

Figure 4.12: This is a plot of the expected scalar form factors for the $B_s \to D_s$ decay (the red dots) compared with the curve fit to the equation $f_0 = b_0 + b_1 z$, where $b_0 = 0.7392 \pm 0.0018$ and $b_1 = -2.727 \pm 0.069$. The fit has a χ^2 per degree of freedom of 440.

$b_1 = -2.638 \pm 0.085$, and $b_2 = 1.8 \pm 1.0$. This fit, surprisingly, also has a χ^2 per degree of freedom of 440. Although, looking at the form factor data, perhaps this should not be surprising. The red dots on the graph are tightly packed together and much less spread out then they were for the $B_s \to K$ decay. As a result, it is much harder to determine the full behavior of the data outside of the range $-0.30 < z < -0.015$. Since the χ^2 per degree of freedom value remained the same when adding a quadratic term, I would not expect adding a cubic term to add anything either.

The fit to the equation $f_0 = b_0 + b_1 z + b_2 z^2 + b_3 z^3$, with $b_0 = 0.7405 \pm 0.0019$, $b_1 = -2.628 \pm 0.087$, $b_2 = 1.7 \pm 1.0$, and $b_3 = 0.9 \pm 1.0$, can be found in figure 4.14. This fit, like the two previous ones, has a χ^2 per degree of freedom of 440. So, this points to the scalar form factors associated with the $B_s \to D_s$ decay only requiring a linear fit to get the best possible fit.

Moving on to the vector form factors, which are fit to equation 2.7. I again skip the fit to a constant for the same reasons as previously listed, and instead fit the data

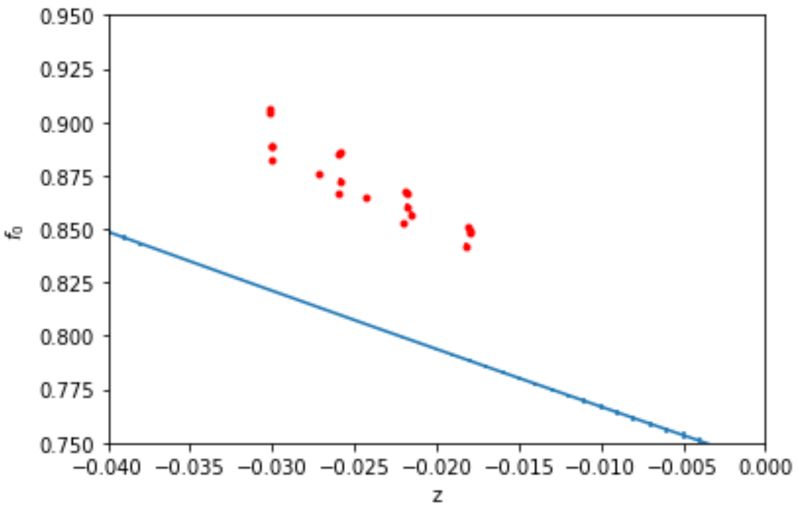

Figure 4.13: This is a plot of the expected scalar form factors for the $B_s \to D_s$ decay (the red dots) compared with the curve fit to the equation $f_0 = b_0 + b_1 z + b_2 z^2$, where $b_0 = 0.7403 \pm 0.0019$, $b_1 = -2.638 \pm 0.085$, and $b_2 = 1.8 \pm 1.0$. The fit has a χ^2 per degree of freedom of 440.

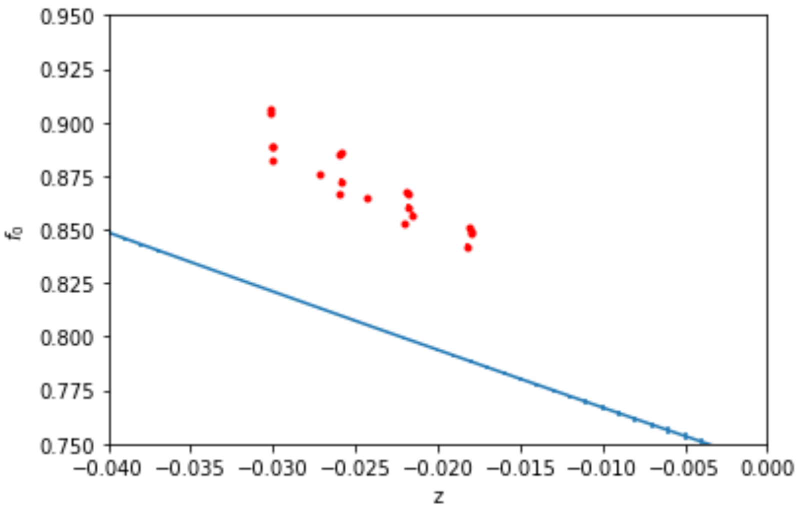

Figure 4.14: This is a plot of the expected scalar form factors for the $B_s \to D_s$ decay (the red dots) compared with the curve fit to the equation $f_0 = b_0 + b_1 z + b_2 z^2 + b_3 z^3$, where $b_0 = 0.7405 \pm 0.0019$, $b_1 = -2.628 \pm 0.087$, $b_2 = 1.7 \pm 1.0$, and $b_3 = 0.9 \pm 1.0$. The fit has a χ^2 per degree of freedom of 440.

to $f_+ = b_0 + b_1 \left(z + \frac{1}{2}z^2\right)$, where $b_0 = 0.8553 \pm 0.0047$ and $b_1 = -6.81 \pm 0.21$ with a χ^2 per degree of freedom value of 140. This fit can be found in figure 4.15.

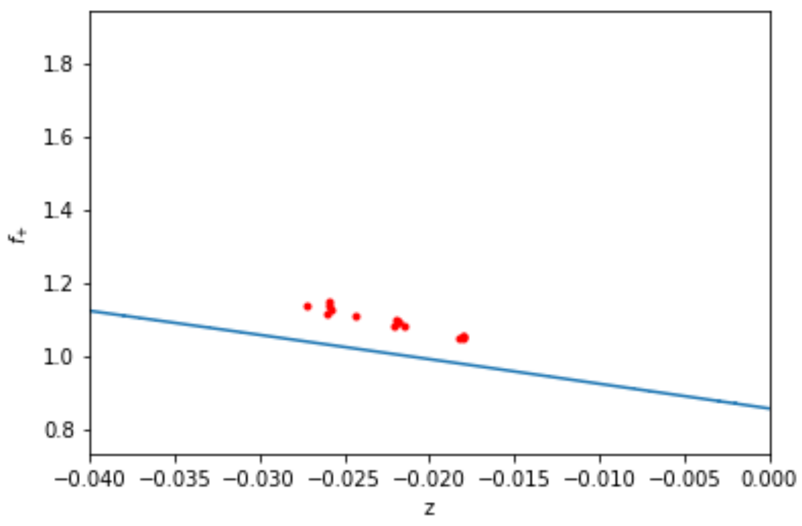

Figure 4.15: This is a plot of the expected vector form factors for the $B_s \to D_s$ decay (the red dots) compared with the curve fit to the equation $f_+ = b_0 + b_1\left(z + \frac{1}{2}z^2\right)$, where $b_0 = 0.8553 \pm 0.0047$ and $b_1 = -6.81 \pm 0.21$. The fit has a χ^2 per degree of freedom of 140.

Next I fit the vector form factor data to the equation $f_+ = b_0 + b_1\left(z - \frac{1}{3}z^3\right) + b_2\left(z^2 + \frac{2}{3}z^3\right)$, where $b_0 = 0.8571 \pm 0.0046$, $a_1 = -6.63 \pm 0.21$, and $b_2 = 1.1 \pm 1.0$. This fit has a χ^2 per degree of freedom of 140. This value did not decrease, and so I would not expect fitting up to a b_3 term to either. The plot of this fit can be seen in figure 4.16. Since the χ^2 per degree of freedom value did not decrease, I would not expect fitting to a b_0 term would do so either, however, I still include the fit next.

The last form factor data fit we are going to look at is the fit to the equation $f_+ = b_0 + b_1\left(z + \frac{1}{4}z^4\right) + b_2\left(z^2 - \frac{2}{4}z^4\right) + b_3\left(z^3 + \frac{3}{4}z^4\right)$, where $b_0 = 0.8572 \pm 0.0046$, $b_1 = -6.63 \pm 0.21$, $b_2 = 1.1 \pm 1.0$, and $b_3 = 1.0 \pm 1.0$, with a χ^2 per degree of freedom of 140. The fit results can be found in figure 4.17.

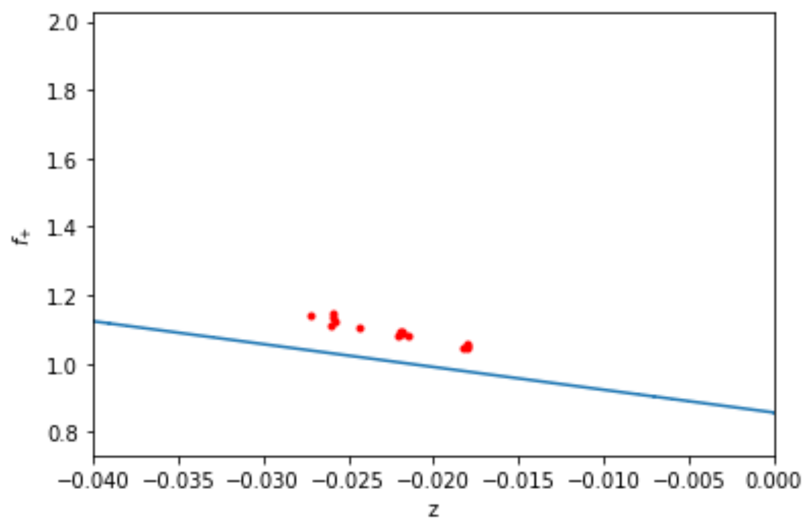

Figure 4.16: This is a plot of the expected vector form factors for the $B_s \to D_s$ decay (the red dots) compared with the curve fit to the equation $f_+ = b_0 + b_1 \left(z - \frac{1}{3}z^3\right) + b_2 \left(z^2 + \frac{2}{3}z^3\right)$, where $b_0 = 0.8571 \pm 0.0046$, $a_1 = -6.63 \pm 0.21$, and $b_2 = 1.1 \pm 1.0$. The fit has a χ^2 per degree of freedom of 140.

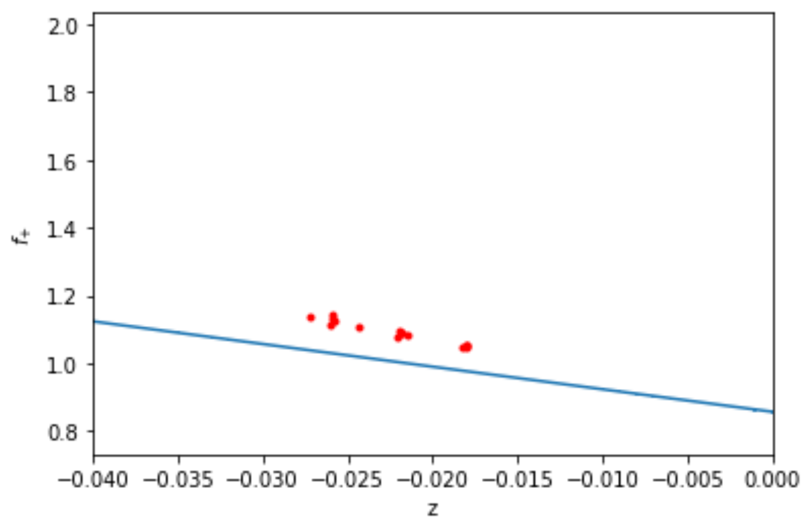

Figure 4.17: This is a plot of the expected vector form factors for the $B_s \to D_s$ decay (the red dots) compared with the curve fit to the equation $f_+ = b_0 + b_1 \left(z + \frac{1}{4}z^4\right) + b_2 \left(z^2 - \frac{2}{4}z^4\right) + b_3 \left(z^3 + \frac{3}{4}z^4\right)$, where $b_0 = 0.8572 \pm 0.0046$, $b_1 = -6.63 \pm 0.21$, $b_2 = 1.1 \pm 1.0$, and $b_3 = 1.0 \pm 1.0$. The fit has a χ^2 per degree of freedom of 140.

Chapter 5

Conclusions

In this project, I fit the form factor data given in [4] to equations of the form of equations 2.6 and 2.7 in an attempt to extract values for the ratio $|V_{ub}|/|V_{cb}|$. Along the way, I learned how to use the `vegas` and `lsqfit` Python packages, encountered numerous Python errors (and solved most of them!), and finally got some results, although I did not quite get to the point of extracting a value for $|V_{ub}|/|V_{cb}|$. Although these results were not great, they were still useful in providing some insight into the form factor equations, as well as how to improve on this project in the future.

To start, based on the fits I did, it to get best possible fit of the scalar and vector form factors for the $B_s \to K$ decay requires fitting to three terms. Conversely, for the $B_s \to D_s$ decay, the scalar and vector form factors only required two terms. Unfortunately, however, I am not sure how well this will hold up, since none of my fits had a 'good' χ^2 per degree of freedom value. The inaccuracies in the fits are in part due to using a simplified fit equation. The fit equation I used does not incorporate the chiral corrections, L^{X_s}, which account for discretization effects caused by the lattice. This is almost certainly causing some amount of error. I also was unable to include the Blaschke factor term in my fits; as a result, my fits do not account for the unphysical quark masses. Aside from including these corrections within the fit equation, another method of accounting for these corrections would be to only include data points with the same initial conditions, such as those with the same lattice spacing, in a given fit. However, the fact that the discrepancies between the form factor data and the fits point towards problems with the fitting code as the main source of error in the fits.

5.1 Future Work

There are a few potential avenues to explore regarding this research in the future. First and foremost, debugging the fitting code is essential to get any sort of decent fits. Next, doing the fits while accounting for the chiral corrections and including the Blaschke factor would be a good place to start to see how much these corrections affect the fits. Additionally, it would be interesting to see if grouping the data based on their initial conditions and then fitting each impacts the fits. Finally, getting to the point where the values for $|V_{ub}|/|V_{cb}|$ can be extracted is another extension of this project that could be done. We also now have experimental data from LHCb not available at the time of the previous paper, which can be utilized in future analysis [12].

Appendix A

Python Computer Programs

A.1 Code sample 1

The following is the Python code for the path integral vs exact calculation comparison.

```
/*******************************************************/

/*   Noah Shanton   */

import vegas
import numpy
import math
import matplotlib.pyplot as plt

N = 8
Xn=1
m=1
a=0.5
E=.5
T=4
t=numpy.arange(0.0, 2.0, 0.2)
def f(x):
    S = 0
    A=(m/(2*math.pi*a))**(N/2)
    x=numpy.append(x,Xn)
    x=numpy.insert(x,0,Xn)
    for j in range(N):
      S+=((m/(2*a))*(x[j+1]-x[j])**2+(a*x[j]**2)/2)
    return math.exp(-S)*A

s=0
V=numpy.zeros(len(t))
while s < len(t):
    t = numpy.arange(0.0, 2.0, 0.2)
    Xn=t[s]
    lim = 5
    intrange = [-lim,lim]
    intlist = [intrange for j in range(N-1)]
    integ = vegas.Integrator(intlist)
    result = integ(f, nitn=100, neval=10000)
    V[s]=result.mean
    s+=1

def g(t):
  return (math.exp(-(t**2)/2)/(math.pi**(0.25)))**2*math.exp(-E*T)

V2=numpy.vectorize(V)
g2=numpy.vectorize(g)

plt.plot(t, g2(t),"r--",label="Exact")
plt.plot(t, V,"b*",label="Path Integral")
plt.ylabel("<x|e^(-HT)|x>")
plt.xlabel("x")
plt.legend(loc="upper right")
plt.show()
```

A.2 Code sample 2

The following is an excerpt from my Python code involving the Metropolis Monte Carlo algorithm. It is the algorithm for a jackknife error estimation.

```
/***********************************************************/

/*   Noah Shanton   */

for sledge in range(N):
  for ash in range(N_cf):  #"transpose" list of G values because my brain likes this better
    G_inOrder[sledge][ash]=G[ash][sledge]

for bot in range(N):        # create a jachknifed list
  for lemon in range(N_cf):
    for pain in range(N_cf):
      if pain != lemon:
        G_J_v2[bot][lemon][pain]=G_inOrder[bot][pain]
      else:
        G_J_v2[bot][lemon][pain]=0

for kapkan in range(N): #calculate the average G value within each jachknife
  for glasses in range(N_cf):
    sum=0
    for glaz in range(N_cf):
      sum+=G_J_v2[kapkan][glasses][glaz]
    G_Avg_v2[kapkan][glasses]=sum/(N_cf-1)

values_squared=G_Avg_v2**2
```

```
for iq in range(N):         # calculate the error
  mozzie=0
  thermite=0
  for kali in range(N_cf):
    mozzie+= G_Avg_v2[iq][kali]
    thermite+= values_squared[iq][kali]
  G_Avg_sq[iq]=(mozzie/N_cf)**2
  G_sq_Avg[iq]=thermite/N_cf
G_err_v2= math.sqrt(N_cf-1) * (G_sq_Avg - G_Avg_sq)**0.5
```

A.3 Code sample 3

The following is the Python code for one of the form factor fits (namely, the scalar form factor data for the $B_s \to K$ decay fit to the a_2 term).

```
/***********************************************************/

/*   Noah Shanton   */

priorK02 = gv.BufferDict()         # a priori values for fit parameters

priorK02['a0']= gv.gvar(1.1,1.0)
priorK02['a1']=gv.gvar(-3,1.0)
priorK02['a2']=gv.gvar(9.5,1.0)

priorK02['M_Bs']=np.array([[gv.gvar(3.23019,0.00025)], [gv.gvar(3.26785,0.00033)],
[gv.gvar(3.23585,0.00038)], [gv.gvar(2.30906,0.00026)], [gv.gvar(2.30122,0.00026)]])


priorK02['M_K']=np.array([[gv.gvar(0.31195,0.00014)],[gv.gvar(0.32870,0.00017)],
[gv.gvar(0.35744,0.00021)],[gv.gvar(0.22861,0.00021)],[gv.gvar(0.24566,0.00013)]])

fitK02 = lsqfit.nonlinear_fit(data=(EnergyK0,form_factorsK0),prior=priorK02,fcn=fcnK02)

pK = fitK02.p                  # best-fit values for parameters
outputsK02 = gv.BufferDict()

outputsK02['a0'] = pK['a0']
outputsK02['a1']=pK['a1']
outputsK02['a2']=pK['a2']

outputsK02['M_Bs'] = pK['M_Bs']
outputsK02['M_K'] = pK['M_K']

inputsK = OrderedDict()
inputsK['f0'] = form_factorsK0     #y values
inputsK['prior'] =priorK02          #priors

print(fitK02)
```

```
fK02_values=np.zeros(len(x_axis))
fK02_error=np.zeros(len(x_axis))
maybe=0
for maybe in range(len(x_axis)):
  fK02_values[maybe]=(pK['a0']+pK['a1']*x_axis[maybe]+pK['a2']*x_axis[maybe]**2).mean
  fK02_error[maybe]=(pK['a0']+pK['a1']*x_axis[maybe]+pK['a2']*x_axis[maybe]**2).sdev

plt.errorbar(x_axis,fK02_values,yerr=fK02_error)
plt.errorbar(z_list_K,form_K0_values,yerr=form_K0_error,fmt='r.')
plt.ylabel('$f_{0}$')
plt.xlabel('z')
plt.show()
```

References

[1] Gouranga C. Nayak. *Matter-Antimatter Asymmetry Of The Universe and Baryon Formation From Non-Equilibrium Quarks and Gluons*. (unpublished). 2019. arXiv: 1909.05640 [physics.gen-ph].

[2] Samuel J. Ling, Jeff Sanny, and William Moebs. *University Physics*. (unpublished). Rice University, 2016. ISBN: 978-1-947172-22-7.

[3] T. N. Pham. *CKM Matrix Elements*. 2011. arXiv: 1110.6050 [hep-ph].

[4] Christopher J. Monahan et al. "Form factor ratios for $B_s \to Kl\nu$ and $B_s \to D_s l\nu$ semileptonic decays and $|V_{ub}/V_{cb}|$". In: *Physical Review D* 98.11 (Dec. 2018). ISSN: 2470-0029. DOI: 10.1103/physrevd.98.114509. URL: http://dx.doi.org/10.1103/PhysRevD.98.114509.

[5] P.A. Zyla et al. "Review of Particle Physics". In: *PTEP* 2020.8 (2020). and 2021 update, p. 083C01. DOI: 10.1093/ptep/ptaa104.

[6] P. Skands. "Introduction to QCD". In: *Searching for New Physics at Small and Large Scales* (Sept. 2013). Lectures at TASI 2012. DOI: 10.1142/9789814525220_0008. URL: http://dx.doi.org/10.1142/9789814525220_0008.

[7] G. Peter Lepage. "LATTICE QCD FOR NOVICES". In: *Proceedings of HUGS 98* (1998). (unpublished). arXiv: hep-lat/0506036v1.

[8] G. Peter Lepage. *gvar*. URL: https://github.com/gplepage/gvar.

[9] David J Griffiths and Darrell F Schroeter. *Introduction to Quantum Mechanics*. 3rd ed. Cambridge University Press, 2018. ISBN: 978-1-107-18963-8.

[10] Peter Young. "JackKnife and Bootstrap Resampling Methods in Statistical Analysis to Correct for Bias". (unpublished).

[11] G. Peter Lepage. *lsqfit*. URL: https://github.com/gplepage/lsqfit.

[12] R. Aaij et al. "First Observation of the Decay Bs0→K+ and a Measurement of —Vub—/—Vcb—". In: *Physical Review Letters* 126.8 (Feb. 2021). ISSN: 1079-7114. DOI: 10.1103/physrevlett.126.081804. URL: http://dx.doi.org/10.1103/PhysRevLett.126.081804.