FERMILAB-MASTERS-2006-07

SIMULATION OF D^0-$\bar{D}^0$ MIXING AND PRELIMINARY ANALYSIS OF CDF DATA

by

NAGESH P. KULKARNI

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2006

MAJOR: PHYSICS

Approved by:

_____

Advisor Date

ACKNOWLEDGMENTS

This project would not have been possible without the guidance and help from the many wonderful people in Wayne State University with whom I had a chance to work for the past two years.

I wish to thank Dr. Karchin, my advisor, for his excellent mentoring and direction throughout my research work. Working with Dr. Karchin, I have realized the importance of paying attention to the smallest of details and striving toward perfection in every task.

My sincere thanks to Dr. Mattson for providing me the data files and also for providing tons of information related to the experiment. I am grateful to Dr. Harr for the useful discussions that contributed in a big way to my research. I would also like to thank Mr. Alfredo Gutierrez for his work and help in the lab.

The constant encouragement and support from all the faculty members of the Physics Department was heartening. As space constraints prevent me from naming every one here, let me simply say that it was a privilege to work with each one of them and I look forward to continued association with this department.

Finally, thanks to my wife and daughter for their support and for making me start each day with a smile, and for endless cups of coffee ....

TABLE OF CONTENTS

1 ACKNOWLEDGMENTS **ii**

2 INTRODUCTION **1**

2.1 Motivation 1

2.2 D^0 Mesons 2

2.3 D^0 Decays and Search for D^0-$\bar{D}^0$ Mixing 3

2.4 Time Dependent Decay Rate 5

2.5 Past Results and Current Research 8

3 SIMULATION OF TIME DEPENDENT DECAY RATE RATIO **10**

3.1 Introduction 10

3.2 Simulation Technique 10

3.3 Confidence Level Contour 11

3.4 Result and Conclusion 13

4 DECAY TIME DISTRIBUTION **15**

4.1 Introduction 15

4.2 Data Taking and Online Filtering 15

4.3 Data Analysis 16

4.3.1 Standard Cuts 16

4.3.2 Preliminary Analysis 17

4.3.3 Removing Trigger Bias 17

4.4 Background Subtraction 18

4.5 Mean Lifetime 21

4.6 Conclusion 22

A DESCRIPTION OF LEAFS **23**

B ROOT/C++ Programs for Decay Time Analysis **24**

B.1 ROOT Program to Generate ScatterPlot 24

B.2 Program to Generate D^0 Mass for a ct range 24

B.3 Program for Decay Time Distribution 26

C ROOT/C++ programs for Simulation of $D^0 - \bar{D}^0$ Mixing **29**

C.1 Program to Generate Wrong-sign Distribution 29

C.2 Program to Generate Right-sign Distribution 30

C.3 Program to Generate WS/RS Ratio 31

C.4 Program to Generate 95% C.L. Ellipse 32

LIST OF FIGURES

2.1 Idealized representation of a $p\bar{p}$ interaction in the transverse plane. The z-axis is along the direction of the beam pipe and the dotted circle shows the inner radius of the beam pipe. Short-lived D mesons decay quickly before they can be observed, and their existence is inferred from their decay products by backtracking the trajectories of kaons and pions. 4

2.2 Cabibbo-favored D^0 decay. It is also called right sign decay. 4

2.3 Direct DCS D^0 decay. These decays are called wrong-sign decays. 5

2.4 The D^0 meson can also decay through mixing. The quark level Feynman diagram shows that D^0 first decays to $\bar{D}^0$ which in turn decays to $K^+\pi^-$. 5

2.5 95% CL regions from the BABAR experiment, reproduced from [5] 9

3.1 Simulated WS decay rate from equation (2.3). The time is in units of proper lifetime of D^0 mesons and the fit function is the same as equation (2.3) with x'^2 and y' as adjustable parameters. 11

3.2 Simulated RS decay rate using equation (2.4). The curve is the exponential fit. 12

3.3 The ratio of wrong-sign to right-sign decay rates, corresponding to $r(t)$ of equation (2.1). The fit function is a polynomial of second order. 12

3.4 95% C.L. with no CP violation. 13

4.1 $ct(D^0)$ vs mass(D^0) scatter plot. D^0 mass is on the x-axis and ct on the y-axis. The scatter plot shows the data points concentrated between the mass range of 1.84 GeV/c^2 and 1.88 GeV/c^2. The data points are dense in the ct range between 0.00 cm and 0.1 cm. 18

4.2 D^0 mass distribution for various ct intervals. (a) 0.00 cm $\leq ct \leq$ 0.04 cm. (b) 0.04 cm $\leq ct \leq$ 0.08 cm. (c) 0.08 cm $\leq ct \leq$ 0.12 cm. (d) 0.12 cm $\leq ct \leq$ 0.16 cm. (e) 0.16 cm $\leq ct \leq$ 0.20 cm. The vertical scale is events per 2 MeV/c^2. 19

4.3 Measured $ct(D^0)$ for the mass range 1.835 GeV/c^2 to 1.895 GeV/c^2. The odd shaped non-exponential distribution is due to trigger bias. 20
4.4 The ct distribution for the mass range 1.835 GeV/c^2 to 1.895 GeV/c^2, with trigger bias removed. The curve shows the exponential fit. 20
4.5 RS D^0 lifetime in cτ for the mass range 1.835 GeV/c^2 to 1.895 GeV/c^2, with background subtraction. 22

LIST OF TABLES

2.1 Important properties of D mesons (Particle Data Group average values). . . 2

2.2 Summary of the results from the two experiments with no CP violation. The 95% CL intervals are obtained using maximum likelihood fit. 8

4.1 Examples of online filtering. 16

4.2 Standard cuts. 17

A.1 Description of Leafs in DStar_01_1.root file. 23

CHAPTER 1

INTRODUCTION

1.1 Motivation

Certain elementary particles change their flavor as they travel. A well known example is neutrino oscillations, in which some of the electron neutrinos emitted from the sun transform into muon or tau neutrinos on their way, as a periodic function of time [1]. The phenomenon of oscillations is also associated with some other particles such as neutral K and B mesons. This interesting feature of elementary particles is a consequence of mixing of quantum states. The mixing of quantum states occurs in neutral K and B mesons because their mass eigenstates are not the same as flavor eigenstates. In fact, the mass eigenstates can be expanded as linear combinations of flavor eigenstates. For example, a B^0 meson and its antiparticle $\bar{B}^0$ can combine in a symmetric or anti-symmetric manner to give two new states with slight differences in their masses and lifetimes. In Dirac notation (assuming CP conservation),

$$|B_H\rangle = \tfrac{1}{\sqrt{2}}(|B^0\rangle - |\bar{B}^0\rangle) \text{ or } |B_L\rangle = \tfrac{1}{\sqrt{2}}(|B^0\rangle + |\bar{B}^0\rangle),$$

where $|B_H\rangle$ is the heavy state and $|B_L\rangle$ is the light state. Another choice is to define the mass eigenstates in terms of lifetimes, such as $|K_S\rangle$ (short lived) and $|K_L\rangle$(long lived). This choice is standard in neutral K mesons.

Mixing of quantum states gives rise to non-vanishing off diagonal terms in the Hamiltonian of the time dependent Schrödinger equation, which leads to B^0-$\bar{B}^0$ oscillations [2]. The oscillations in neutral K mesons also occur in a similar manner. In analogy with observed B^0-$\bar{B}^0$ and K^0-$\bar{K}^0$ oscillations, D^0-$\bar{D}^0$ oscillations also should occur. However, the oscillations of D^0 mesons are strongly suppressed by CKM factors.

D^0-$\bar{D}^0$ mixing is of particular interest for two reasons. Firstly, the standard model predicts D^0-$\bar{D}^0$ oscillations as a small effect, difficult to observe in current experimental

reach. If we observe this effect in the current experiments, it will be an indirect signal for new physics such as a fourth quark flavor or contributions from supersymmetry. The second reason is that mixing is also a crucial ingredient in the phenomenon of CP-violation and as CP is not conserved in new physics, observation of CP violation in D^0-$\bar{D}^0$ mixing would be an unambiguous sign of new physics [3, 4, 5], hence of great interest.

1.2 D^0 Mesons

In general, D^0, D^+, D^* and their antiparticles are called D mesons. In D mesons, a charm quark (c) combines with a quark of different flavor (down, up, strange) and so they are generally referred to as charm particles. For example, a D^0 meson is composed of a charm quark and an anti-up quark. Two spin 1/2 quarks can combine to give a spin triplet or singlet combination. D^0 particles have spin 0 due to the singlet combination whereas D^* particles have spin 1 due to the triplet combination. The D^*s are called spin excited mesons. D mesons have extremely short lifetimes. The mean lifetime of D^0 mesons is 410.3 $\pm$ 1.5 $\times 10^{-15}$ s. The D^* lifetime is given by the decay width Γ which is related to the mean lifetime τ by

$$\tau = \frac{\hbar}{\Gamma c^2}$$

where $\hbar$ is Planck's constant. Table 2.1 summarizes some of the important properties of D mesons.

Particle	Antiparticle	Mass (MeV/c^2)	Mean Lifetime τ $\times 10^{-15}$(s)	Mass Width Γ
$D^0(c\bar{u})$	$\bar{D}^0$ ($\bar{c}$u)	1864.1 $\pm$ 1.0	410.3 $\pm$ 1.5	
$D^+(c\bar{d})$	D^- ($\bar{c}$d)	1869.4 $\pm$ 0.5	1040 $\pm$ 7	
$D^{*0}(c\bar{u})$	$\bar{D}^{*0}$ ($\bar{c}$u)	2006.7 $\pm$ 0.5		$\leq$ 2.1 MeV
$D^{*+}(c\bar{d})$	D^{*-} ($\bar{c}$d)	2010.0 $\pm$ 0.5		$\leq$ 96 $\pm$ 4 $\pm$ 22 KeV

Table 1.1. Important properties of D mesons (Particle Data Group average values).

1.3 D^0 Decays and Search for D^0-$\bar{D}^0$ Mixing

The D mesons are too short-lived to be directly observed using the CDF II detector and their existence must be inferred from the observations of more stable particles to which they decay. To get an idea, the transverse distance between the points at which the D^0 mesons are produced and decay can be estimated from transverse momentum, τ and invariant mass. The CDF experiment at Fermilab requires a minimum transverse momentum p_T=5.5 GeV/c due to online filtering [6]. Assuming total momentum $p = p_T$, we get $\gamma = 3.115$ which in turn gives transverse distance = 0.0365 cm. At CDF II, the beam pipe diameter is 2.2 cm [7]. In the proton anti-proton collisions, D mesons are produced very close to the collision point and it can be seen from the above calculation that they decay inside the beam pipe. Therefore, the reaction is observed in terms of final decay products such as kaons and pions. The trajectories of these particles are traced back to their origin to obtain the point at which the D meson decays as shown in Figure 2.1. This idealized representation (effects of magnetic field are ignored and the drawing is not to scale) of the detector shows the point at which a D^0 particle decays. The invariant mass of fleeting D mesons is then calculated from the energies and momenta of kaons and pions using statistical methods.

D mesons decay through weak interactions in which W bosons are emitted or absorbed by constituent quarks. One of the hadronic decay modes has the end products of a kaon and a pion. Weak interactions also give rise to semileptonic processes in which the final decay products are neutrinos and other leptons along with hadrons. The weak interactions which lead to D^0-$\bar{D}^0$ mixing can be understood with the idea of "quark mixing" due to Cabibbo. According to Cabibbo's hypothesis, d and s quark states combine as

$$|d'\rangle = |d\rangle \cos\theta_{\mathrm{C}} + |\mathrm{s}\rangle \sin\theta_{\mathrm{C}} \text{ and } |s'\rangle = |s\rangle \cos\theta_{\mathrm{C}} - |\mathrm{d}\rangle \sin\theta_{\mathrm{C}}$$

where θ_C is the Cabibbo angle. The idea of d and s quark mixing extended to include the bottom quark results in the CKM matrix, allowing the mixing between d, s, and b quarks.

Due to Cabibbo quark mixing, the "cdW" vertex in the weak interactions is suppressed relative to the "csW" vertex by a factor of $\tan\theta_{\mathrm{C}}$. The decays which involve "cdW" vertices are called "Cabibbo-suppressed" decays. Similarly there are "Cabibbo-favored" decays which involve "csW" couplings.

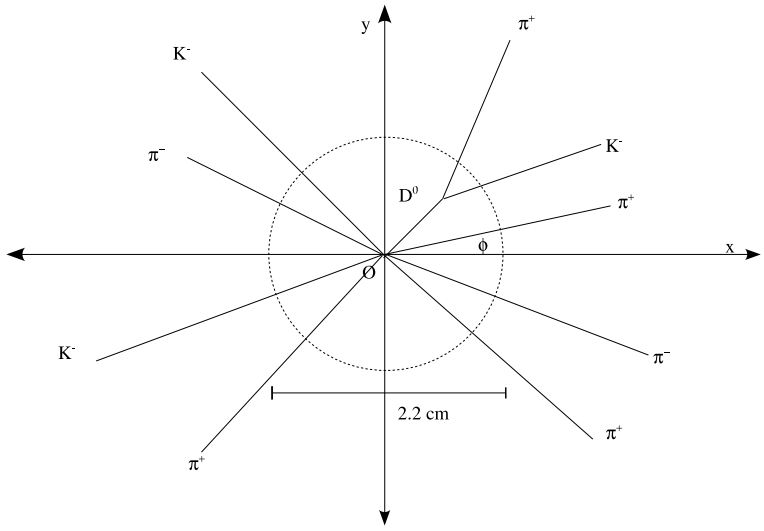

Figure 1.1. Idealized representation of a $p\bar{p}$ interaction in the transverse plane. The z-axis is along the direction of the beam pipe and the dotted circle shows the inner radius of the beam pipe. Short-lived D mesons decay quickly before they can be observed, and their existence is inferred from their decay products by backtracking the trajectories of kaons and pions.

The Cabibbo-favored decay is explained at quark level in the following Feynman diagram (Figure 2.2). The reaction involves a "csW" vertex. The final decay products, $\bar{u}$ and s quarks, combine to give a K^- and u and $\bar{d}$ quarks give a π^+. The Cabibbo-favored decay is also called right-sign (RS) decay. The probability of occurrence of $D^0 \to K^-\pi^+$ relative to all decay modes is approximately 5 percent.

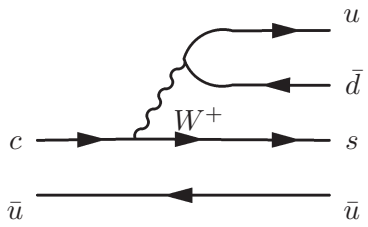

Figure 1.2. Cabibbo-favored D^0 decay. It is also called right sign decay.

The weak interaction for $D^0 \to K^+\pi^-$ involves both the "cdW" and "usW" vertices. This is a doubly Cabibbo suppressed decay (DCS). It is explained in the following Feynman diagram (Figure 2.3). The final state products, $\bar{s}$ and u quarks, give K^+ and d and $\bar{u}$ quarks give π^-. The DCS decay is also called wrong-sign (WS) decay. The probability of

occurrence of $D^0 \to K^+\pi^-$ decay is approximately 0.004 percent.

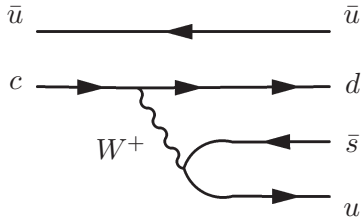

Figure 1.3. Direct DCS D^0 decay. These decays are called wrong-sign decays.

The D^0 can also give K^+ π^- final states through mixing. The following figure (Figure 2.4) shows the Feynman diagram for DCS decay through mixing.

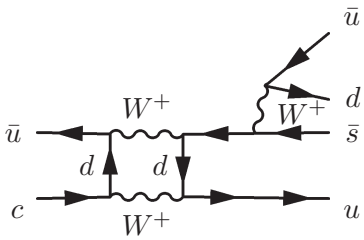

Figure 1.4. The D^0 meson can also decay through mixing. The quark level Feynman diagram shows that D^0 first decays to $\bar{D}^0$ which in turn decays to $K^+\pi^-$.

Experimentally, D^0-$\bar{D}^0$ mixing can be observed in multiple ways. One way is to study $D^0 \to K\pi$ decay. It is also possible to study semileptonic decays. In this work, we focus on $D^0 \to K\pi$ decay.

The usual method to identify CF and DCS decays is to tag the charged pion in the decay sequence $D^{*+} \to D^0\pi^+$ or $D^{*-} \to \bar{D}^0\pi^-$. In the CF decay, the pions from D^0 and D^* are of the same sign and in DCS decay, they have opposite signs. The mixing is studied as a ratio of DCS to CF decays. The search is complicated by contributions from direct DCS decay. Since the two sequences, $D^0 \to \bar{D}^0 \to K^+\pi^-$ and $D^0 \to K^+\pi^-$ have the same final state, one needs to separate these two contributions using the time-dependent decay rates. The time dependent decay rate is discussed in the next section.

1.4 Time Dependent Decay Rate

In this section, we apply basic concepts of quantum mechanics to understand the D^0-$\bar{D}^0$ mixing qualitatively. Here we give a brief review of the formalism; a complete treatment of

D^0-$\bar{D}^0$ mixing is discussed in the PDG review [1].

In the following discussion, we shall measure time in the rest frame of the D meson. Assuming CP is conserved[1], at $t = 0$, i.e. when the particle is produced, the initial mass eigenstate can be written as a linear combination as below;

$$|D_H\rangle = \frac{1}{\sqrt{2}}(|D^0\rangle - |\bar{D}^0\rangle)$$
$$|D_L\rangle = \frac{1}{\sqrt{2}}(|D^0\rangle + |\bar{D}^0\rangle).$$

At this point, $|D_H\rangle$ and $|D_L\rangle$ are simply labels. The inverse of the above equations will give $|D^0\rangle$ as a linear combination of $|D_H\rangle$ and $|D_L\rangle$. The states evolve according to the Schrödinger equation. Then, at a later time t,

$$|D^0\rangle(t) = \tfrac{1}{\sqrt{2}}(|D_H\rangle e^{-im_H t} + |\bar{D}_L\rangle e^{-im_L t})$$

where m_H and m_L are the rest masses of D_H and D_L particles. To ensure the exponential decay, we must multiply the absolute probability $||D^0\rangle(t)|^2$ by $e^{-t/\tau}$, where τ is the mean lifetime. This is equivalent to multiplying each mass eigenstate by the factor $e^{-\Gamma_{H,L}t/2}$, where Γ_H and Γ_L are the decay widths of $|D_H\rangle$ and $|D_L\rangle$ respectively. The equation then becomes

$$|D^0\rangle(t) = \tfrac{1}{\sqrt{2}}(|D_H\rangle e^{-im_H t} e^{-\Gamma_H t/2} + |\bar{D}_L\rangle e^{-im_L t} e^{-\Gamma_L t/2})$$

We define $a_\alpha(t) = e^{-im_\alpha t} e^{-\Gamma_\alpha t/2}$, where α stands for the subscripts H or L, and write

$$|D^0\rangle(t) = \tfrac{1}{\sqrt{2}}[a_H(t)|D_H\rangle + a_L(t)|\bar{D}_L\rangle]$$

Let the initial flavor state be $|D^0\rangle$. To find the time-dependent probabilities of D^0 and $\bar{D}^0$ at time t, we define,

$$A(t) = \tfrac{1}{2}[a_H(t) + a_L(t)] \quad \text{and} \quad \bar{A}(t) = \tfrac{1}{2}[a_H(t) - a_L(t)],$$

so that

$$|D^0\rangle(t) = A(t)|D^0\rangle + \bar{A}(t)|\bar{D}^0\rangle \quad \text{and} \quad |\bar{D}^0\rangle(t) = A(t)|D^0\rangle - \bar{A}(t)|\bar{D}^0\rangle$$

[1] We assume CP conservation through out this work.

Then the probability of finding a particle in a state $|\bar{D}^0\rangle$ is

$$\begin{aligned} |\bar{A}(t)|^2 &= \frac{1}{4}[e^{-\Gamma_H t} + e^{-\Gamma_L t} - 2e^{-(\Gamma_H+\Gamma_L)t/2}\cos(\delta \mathrm{m\,t})] \\ &= \frac{1}{4}e^{-\Gamma_H t}[1 + e^{\delta\Gamma t} - 2e^{-\frac{1}{2}\delta\Gamma t}\cos(\delta \mathrm{m\,t})] \end{aligned}$$

where $\delta m = m_H - m_L$ and $\delta\Gamma = \Gamma_H - \Gamma_L$. It can be seen that a particle initially in state D^0 would transform into $\bar{D}^0$ at a later time t and its amplitude depends on mass difference δm. This is analogous to K^0-$\bar{K}^0$ or B^0-$\bar{B}^0$ mixing. However, unlike K and B mesons, the mass difference δm is very small in the case of D mesons and so the oscillations are very slow.

Two dimensionless quantities play important role in the theory and experiments,

$$x = \frac{\delta m}{\Gamma} \text{ and } \mathrm{y} = \frac{\delta\Gamma}{2\Gamma},$$

where Γ is the average width ($\frac{\Gamma_H+\Gamma_L}{2}$).

Following the formalism discussed in the PDG review [1, 2, 3], the time-dependent WS decay rate relative to the integrated RS decay rate is given by

$$\begin{aligned} r(t) &= \frac{1}{N^0_{RS}}\frac{dN_{WS}(t)}{dt} \\ &= e^{-\Gamma t}[R_D + \sqrt{R_D}y'\Gamma t + \frac{(x\prime^2 + y'^2)}{4}(\Gamma t)^2], \end{aligned} \tag{1.1}$$

where N^0_{RS} is the total number of RS decays integrated over all time. $N_{WS}(t)$ is the number of un-decayed wrong sign events at time t. x' and y' are related to x and y by

$$x' = x\cos\delta + \mathrm{y}\sin\delta, \quad \mathrm{y}' = -\mathrm{x}\sin\delta + \mathrm{y}\cos\delta.$$

δ is the strong phase difference between WS and RS amplitudes. In equation (1.1), the first term R_D is due to the DCS amplitude and is determined experimentally. The world average value of R_D is 0.362 ± 0.029 % [1]. The middle term is due to interference between the two processes and the last term due to mixing [3].

The time integrated WS to RS ratio is then given by

$$R = \int_0^\infty r(t)dt = R_D + \sqrt{R_D}y' + \frac{x'^2 + y'^2}{2}. \quad (1.2)$$

Hence, large values of x' and y' correspond to large mixing.

1.5 Past Results and Current Research

Various experiments have measured R_D, x′ and y′ of equation (1.2) to obtain 95% confidence level intervals on these parameters. The results of two experiments, the BABAR detector at the PEP-II e^+e^- collider and the Belle detector at the KEKB e^+e^- collider, are summarized in Table 2.2. These experiments construct the D^0 signal from kaons and pions and use the pion from D^{*+} to identify WS decays. The general strategy to determine the mixing parameters is to fit the RS and WS data simultaneously. These experiments do this analysis both by assuming CP conservation and allowing for CP violation. The 95% CL contours for the BABAR experiment are shown in Figure 2.5.

Experiment	WS decays	Parameter	95% CL interval ($\times 10^{-3}$)
BELLE	845 ± 40	x'^2	$x'^2 \leq 0.81$
90 fb^{-1}		y'	-8.2 $\leq y' \leq$ 16
		R_D	2.7 $\leq R_D \leq$ 4.0
BABAR	430	x'^2	$x'^2 \leq 2.0$
57.1 fb^{-1}		y'	-27 $\leq y' \leq$ 22
		R_D	2.4 $\leq R_D \leq$ 4.9

Table 1.2. Summary of the results from the two experiments with no CP violation. The 95% CL intervals are obtained using maximum likelihood fit.

A large number of WS decays with less background can give better results. The Belle collaboration (with 430 WS decays) and the BABAR collaboration (with 845 WS decays) have reported the best measurements of mixing parameters so far. The CDF collaboration has reported 2100 WS decays at the integrated luminosity of 0.35 fb^{-1} in $p\bar{p}$ collisions with center of mass energy $\sqrt{s}$ = 1.96 TeV. To incorporate the background, we calculate an equivalent wrong-sign signal using $N^0_{WS} = [N_S/\sqrt{N_S + N_B}]^2$, where N_S is the WS signal (=2100) and N_B is the background. From the D^0 signal plotted as a function of mass

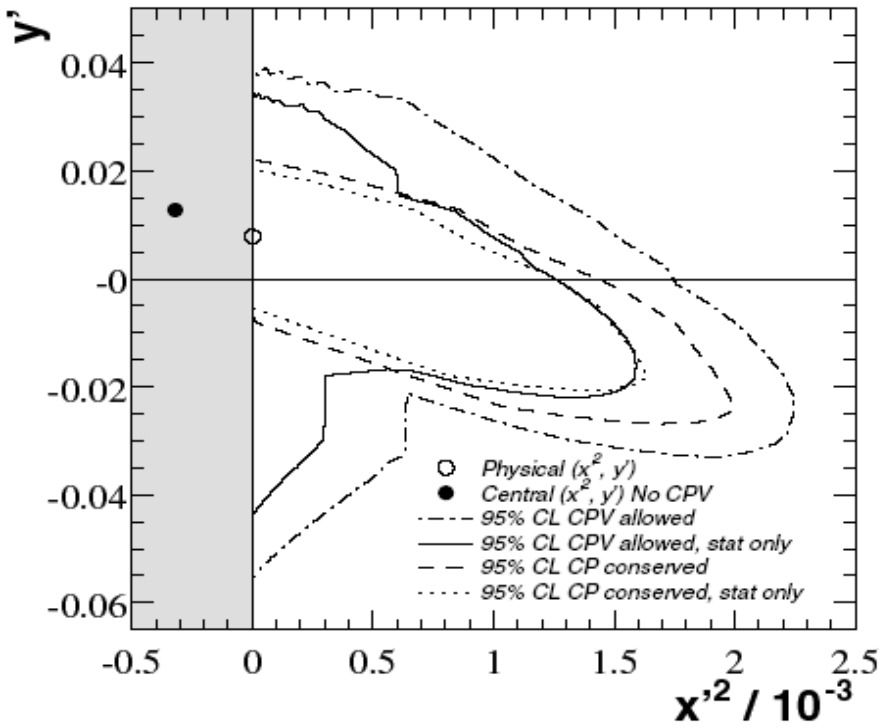

Figure 1.5. 95% CL regions from the BABAR experiment, reproduced from [5] .

difference $\Delta m = m(K^+\pi^-\pi^+) - m(K^+\pi^-) - m(\pi^+)$[6], we take $N_B = 6950$. This gives us the equivalent number of wrong-sign events equal to 650.

In this work, we estimate the accuracy for measuring x' and y' using the CDF detector, under some simplifying assumptions. We simulate the ratio of WS to RS decay rates using equations (1.1) and (1.2), with WS decays as estimated above and no CP violation. We construct the 95% confidence interval contour for x'^2 and y' from the fit. In the next chapter we discuss the simulation.

CHAPTER 2

SIMULATION OF TIME DEPENDENT DECAY RATE RATIO

2.1 Introduction

In this chapter, we simulate the ratio of wrong-sign decay rate to right-sign decay rate. The time dependent WS decay rate relative to the integrated RS decay rate is given by equation (1.1). For a small time interval Δt,

$$N(\Delta t) \simeq -\frac{dN(t)}{dt}\Delta t.$$

With this and expressing the time in units of τ, we write

$$N_{WS}(t) = N^0_{RS}e^t(R_D + \sqrt{R_D}y't + \frac{x'^2 + y'^2}{4}t^2)\Delta t. \tag{2.1}$$

From equations (1.2) and (2.2) we write the WS and RS decay rates in terms of the total number of WS decays (N^0_{WS}) as follows.

$$N_{WS}(t) = \frac{N^0_{WS}}{R_D + \sqrt{R_D}y' + \frac{x'^2+y'^2}{2}}e^{-t}(R_D + \sqrt{R_D}y't + \frac{x'^2 + y'^2}{4}t^2)\Delta t \tag{2.2}$$

and

$$N_{RS}(t) = \frac{N^0_{WS}}{R_D + \sqrt{R_D}y' + \frac{x'^2+y'^2}{2}}e^{-t}\Delta t. \tag{2.3}$$

The equations (2.3) and (2.4) are used to simulate the time dependent ratio of WS to RS decay rates.

2.2 Simulation Technique

We use the random number generator in the ROOT system to generate the distributions. The wrong-sign distribution is generated using equation (2.4) in 50 bins and for the time

range 0 to 10. We take the total number of wrong sign events $N^0_{WS} = 650$, and set the values of x'^2 and y' from Figure 2.5 to 0.5×10^{-3} and 0.0 respectively. The fit function is the same as equation (2.3) with x'^2 and y' treated as adjustable parameters. Similarly, we generate the right-sign distribution using equation (2.4). These distributions are shown in Figures 3.1 and 3.2. We take the ratio of these two distributions to plot a third histogram. We fit this distribution with a polynomial of second order. Figure 3.3 shows this histogram. The box in the figure shows χ^2, the goodness-of-fit. The parameter 'ndf' in the box stands for "number of degrees of freedom" which is equal to the number of data points minus the number of adjustable parameters of the fit function. The second parameter in the statistics box 'Prob' stands for probability and is related to the χ^2 distribution.

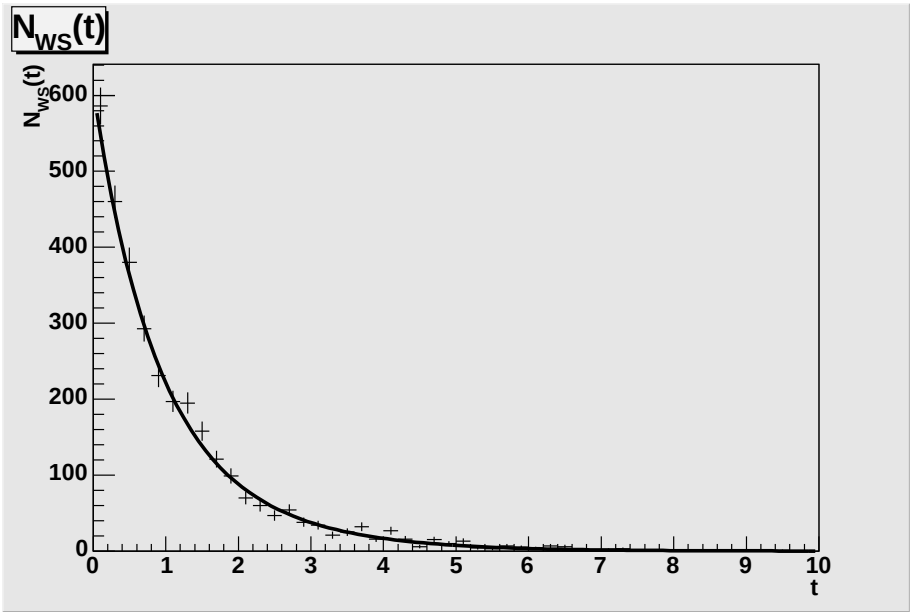

Figure 2.1. Simulated WS decay rate from equation (2.3). The time is in units of proper lifetime of D^0 mesons and the fit function is the same as equation (2.3) with x'^2 and y' as adjustable parameters.

2.3 Confidence Level Contour

In the polynomial fit, the function $p_0 + p_1 t + p_2 t^2$ is applied to the simulated $r(t)$ distribution. The first parameter p_0 corresponds to R_D and the error in p_0 gives the error in R_D. The second parameter p_1 corresponds to $\sqrt{R_D}y'$. The error in y' can be obtained

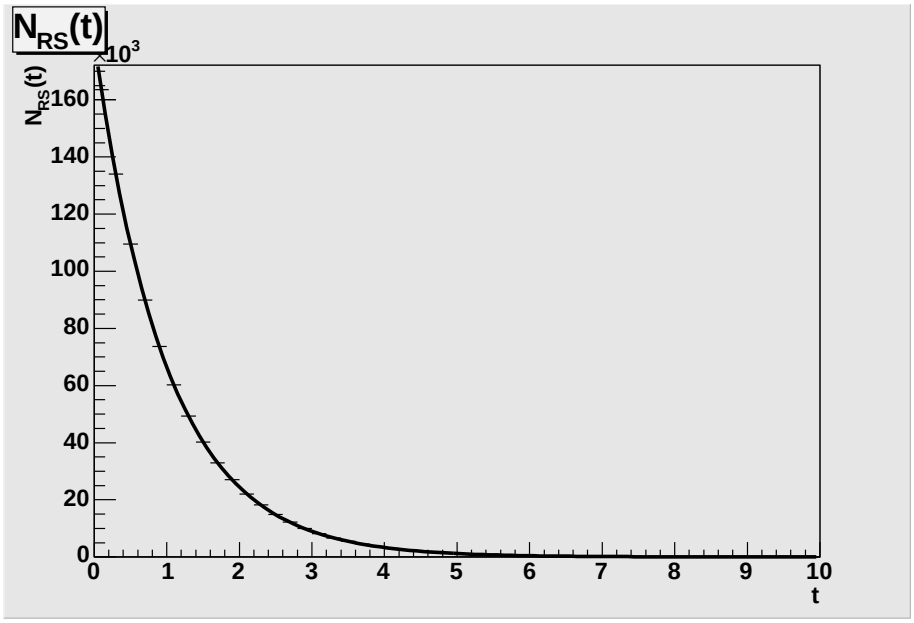

Figure 2.2. Simulated RS decay rate using equation (2.4). The curve is the exponential fit.

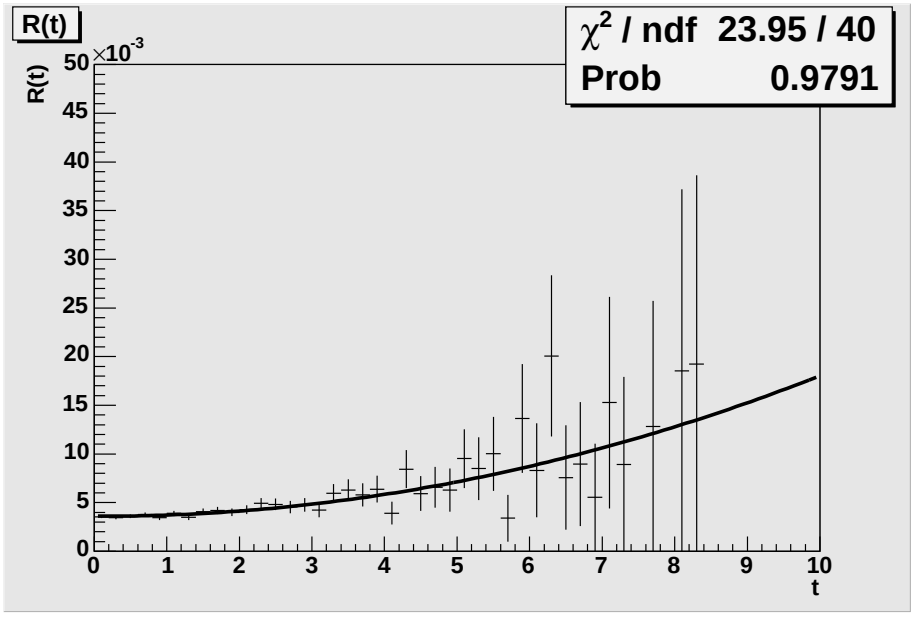

Figure 2.3. The ratio of wrong-sign to right-sign decay rates, corresponding to $r(t)$ of equation (2.1). The fit function is a polynomial of second order.

by differentiating the equation $y' = p_1/\sqrt{R_D}$ with respect to p_1. We get $\delta y' = \delta p_1/\sqrt{R_D}$. Similarly, the third parameter p_2 is equal to $(x'^2 + y'^2)/4$. Differentiating the equation for x'^2 with respect to p_2 and y' gives the error in x'^2, $\delta(x'^2) = \sqrt{16\delta p_2^2 + 4y'^2\delta y'^2}$. We obtain the equation of an ellipse using terms from the 3×3 error matrix, corresponding to a 95% confidence interval, as

$$\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x \sigma_y} = \frac{(1-\rho^2)(1.64^2)}{2},$$

where x corresponds to x'^2, y corresponds to y', and σ_x and σ_y are the errors in x'^2 and y', respectively. ρ is called the correlation coefficient and is given by $\rho = \text{covariance(x,y)}/\sigma_x\sigma_y$ [9]. The factor $1.64^2/2$ gives the 95% confidence interval [1]. Figure 3.4 shows the ellipse plotted in the $x'^2 - y'$ plane. The ROOT programs for this analysis are given in Appendix C.

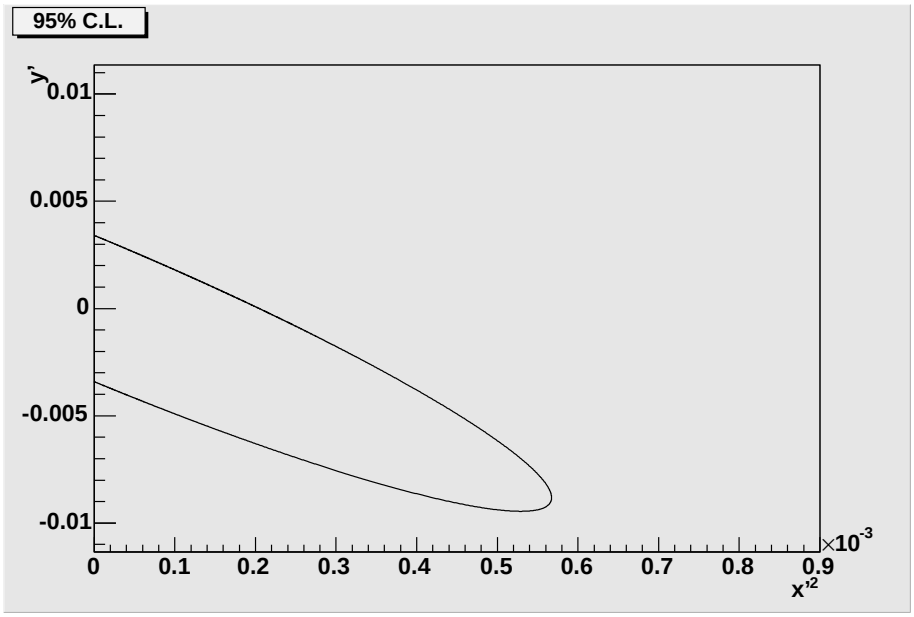

Figure 2.4. 95% C.L. with no CP violation.

2.4 Result and Conclusion

In summary, we estimated the accuracy of the CDF II detector for measuring the mixing parameters. We used WS decay events observed in the CDF Run II experiment at $\sqrt{s} = 1.96$ TeV and 0.35 fb^{-1}, equal to 2100. To incorporate the background, we took the total number

of wrong-sign events equal to 650. We simulated the ratio of wrong-sign decay rate to right-sign decay rate and fit the distribution to a polynomial of second order. From the error matrix, we obtained the 95% C.L. ellipse.

We find $x'^2 \leq 0.6 \times 10^{-3}$ and $-0.0035 \leq y' \leq 0.0035$. The result is considerably more restrictive in x' as compared to the results from the BABAR collaboration and the Belle collaboration (Table 2.2). With continuous improvements in the CDF Run II luminosity and in calibration and analysis softwares, it is possible to obtain the world's best limits on the mixing parameters.

CHAPTER 3

DECAY TIME DISTRIBUTION

3.1 Introduction

In this chapter, we validate the CDF data by analyzing the lifetime distribution of D^0 mesons, which is necessary to measure mixing. The mean lifetime of D^0 mesons was precisely measured in the past experiments. The PDG world average is $\tau = 410.3 \pm 1.5 \times 10^{-15}$ seconds or $c\tau = 0.01230 \pm 0.000045$ cm, where c is the speed of light.

3.2 Data Taking and Online Filtering

The Collider Detector at Fermilab is a multi-purpose experiment to study proton anti-proton collisions at the Fermilab Tevatron collider with center of mass energy $\sqrt{s} = 1.96$ TeV [7, 10, 12]. In Run II, the Tevatron operates with 36 bunches of protons and anti-protons each, with a bunch crossing interval of 396 ns. In online filtering, the events are selected by applying constraints to certain quantities associated with particle tracks, such as four momentum (p^μ), azimuthal angle (ϕ), polar angle (θ), and y-intercept (y_0). In the cylindrical coordinate system of the CDF detector, the origin is at the center of the detector and the z-axis is along the direction of the beam. The azimuthal angle is the angle between the track and the x-axis in the transverse plane and the polar angle is the angle between the track and the z-axis. From the track parameters, other quantities are calculated. These include transverse momentum (p_T), impact parameter (d_0), invariant mass (m), and the opening angle between the two tracks in the transverse plane ($\Delta\phi$). The transverse momentum is the component of momentum vector in the transverse direction to the beam (the x-y plane). The impact parameter d_0 is the distance between the particle track (which is a helix due to the presence of magnetic field in the detector) and the origin at closest distance of approach and is given by the formula

$$d_0 = \frac{\hat{\mathbf{z}} \cdot (\mathbf{r} \times \mathbf{p_T})}{|\mathbf{p_T}|},$$

where $\mathbf{r}$ is the vector pointing from the origin to the helix at minimum distance of approach and $\hat{\mathbf{z}}$ is the unit vector along the z-axis. The invariant mass is defined by $m^2c^4 = E^2 - \mathbf{p}^2c^2$, where E and $\mathbf{p}$ are the total energy and momentum.

Hadronic decays of heavy flavor particles are acquired with online filters as shown in Table 4.1.

Quantity	Filters requirement
χ^2	≤ 25
d_0	0.1mm $\leq$ d$_0$ $\leq$ 1.0mm
p_T	$\geq$ 2.04 GeV
$\Delta\phi$	$0^o \leq \Delta\phi \leq 135^o$
Scalar sum of p_Ts of tracks	$\geq$ 5.5 GeV

Table 3.1. Examples of online filtering.

3.3 Data Analysis

The ROOT system is used for data analysis. It stores the data in a file format called a tree file, which has extension .root. The data is stored in the form of a tree structure, with branches and leafs [11]. Typically the variables are defined as leafs. For example, the transverse momentum of a D^* is stored in a leaf called DS_PT. Each leaf is an array of a certain data type such as integer, float, double, etc. Appendix A describes the ROOT file used for decay time distribution analysis.

3.3.1 Standard Cuts

The D^0 signal is reconstructed from kaons and pions. The initial cuts are applied to reduce the combinatorial background from improper combination of tracks and from mis-identified D^0 tracks. Table 4.2 lists the standard cuts optimized to increase the DCS significance for the mixing analysis [6]; the same cuts are used for this work. The term χ_{xy} in the table

is the goodness-of-fit, L_{xy} is the distance between the z-axis and the vertex (the point at which two or more tracks meet) and σ is the error in the L_{xy} measurement.

Parameter	Cut
χ_{xy} for D^*	≤ 10
L_{xy}/σ for D^*	≤ 15
L_{xy}/σ for D^0	≥ 5
Impact parameter for D^0	≤ 0.01 cm

Table 3.2. Standard cuts.

3.3.2 Preliminary Analysis

We first look at the mass vs ct scatter plot for D^0 particles (Figure 4.1). It is a two dimensional histogram with mass of the D^0 on the x-axis and ct on the y-axis. There are 100 bins on each axis. There is no data beyond the range of 1.7 GeV/c^2 to 2.0 GeV/c^2 due to offline filtering. Also, we see that the particle density decreases rapidly in the ct range from 0.0 cm to 0.25 cm, as the D^0 mesons obey the exponential decay law. For our analysis of the decay time distribution, we choose the mass range between 1.84 GeV/c^2 and 1.88 GeV/c^2 and the ct range between 0.0 cm and 0.20 cm, where we expect to find the maximum signal.

To investigate further, we divide the ct range 0.0 cm to 0.20 cm into five equal parts and plot the mass histogram for each of these regions. Figure 4.2 shows all these histograms.

3.3.3 Removing Trigger Bias

Figure 4.3 is the $ct(D^0)$ distribution with the standard cuts applied. This distribution does not represent exponential decay due to trigger bias. In order to remove the trigger bias (approximately) and retain only the exponential part, a minimum value of ct (=0.038 cm) is subtracted. Figure 4.4 shows the $ct(D^0)$ distribution with trigger bias removed. This histogram is plotted for the mass range of 1.865 ± 0.030 GeV/c^2. It has 25 bins of bin size 0.004 cm each. The standard exponential fit is applied. The error bars correspond to the square root of the number of entries in each bin.

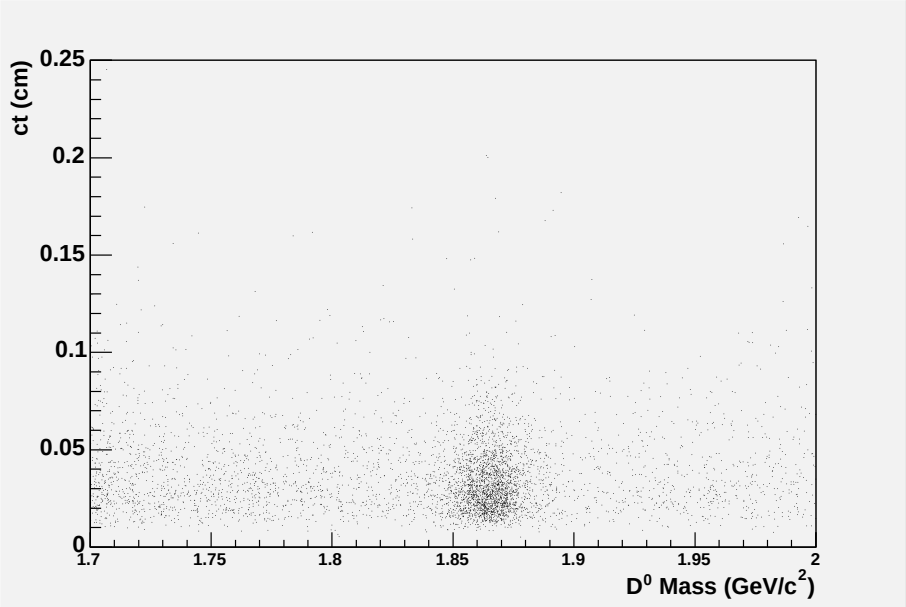

Figure 3.1. $ct(D^0)$ vs mass(D^0) scatter plot. D^0 mass is on the x-axis and ct on the y-axis. The scatter plot shows the data points concentrated between the mass range of 1.84 GeV/c^2 and 1.88 GeV/c^2. The data points are dense in the ct range between 0.00 cm and 0.1 cm.

3.4 Background Subtraction

In order to estimate the background in the mass distribution for D^0 mesons, we examine the histogram (a) in Figure 4.2. It can be divided into three parts. The central part of the histogram shows the peak at approximately 1.865 GeV/c^2, which is close to the mass of D^0 mesons and hence represents the D^0 signal. The data on the left and right sides of this interval is from the other sources of kaons and pions. Thus, the mass distribution contains signal plus background. We estimate the background in the central region as approximately equal to the average of the integrals of the left and right side distributions. The ROOT system can give the integral of a selected range of a distribution. Using this functionality, we calculate the background as follows. We divide the total distribution into three equal mass ranges: 1.775 GeV/c^2 to 1.835 GeV/c^2 (range 1), 1.835 GeV/c^2 to 1.895 GeV/c^2 (range 2) and 1.895 GeV/c^2 to 1.955 GeV/c^2 (range 3). Each range has 25 bins in it. We take the sum of integrals of range 1 and range 2 and divide it by 50, to get the average background per bin. The total background in the range 2 is then approximately equal to 25 times the average background per bin. The ROOT program to do this is given in Appendix B. We

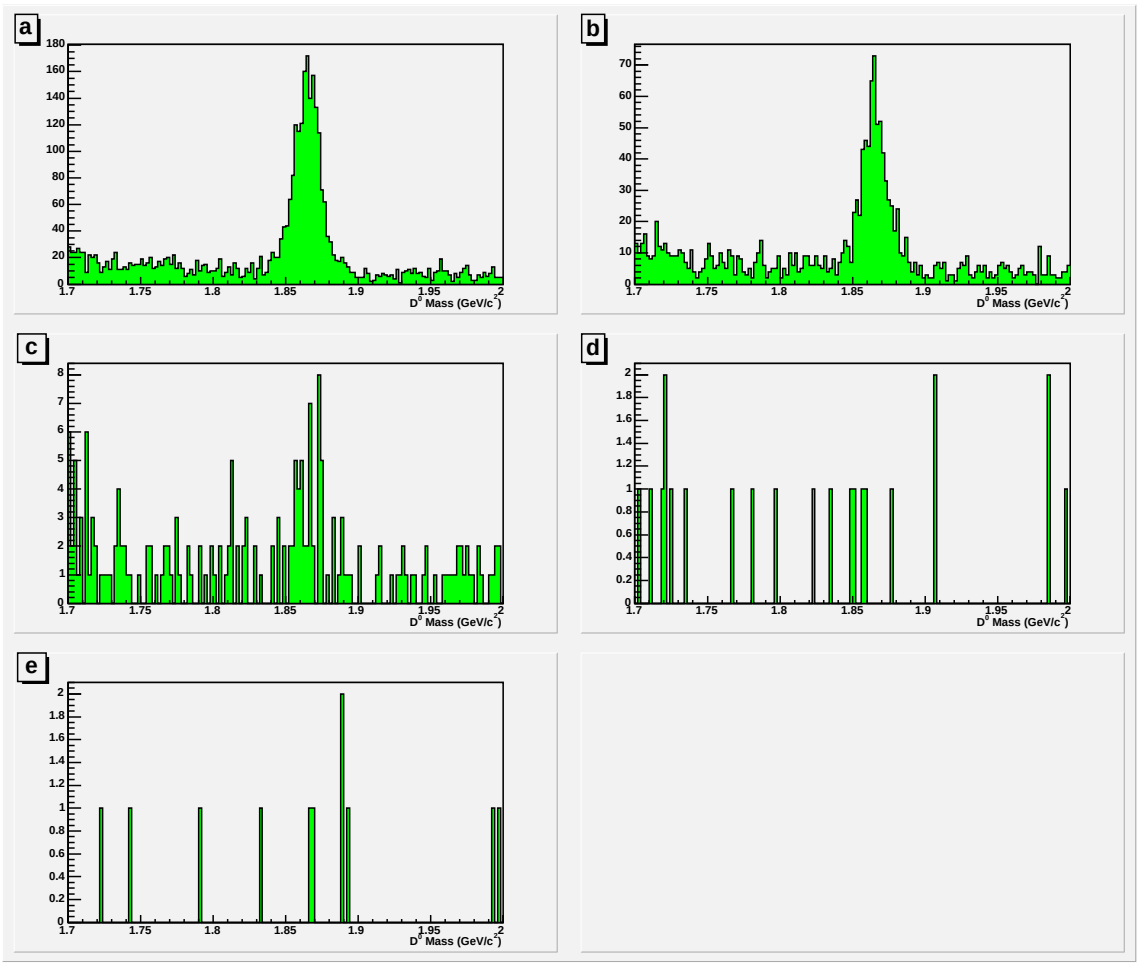

Figure 3.2. D^0 mass distribution for various ct intervals. (a) 0.00 cm $\leq ct \leq$ 0.04 cm. (b) 0.04 cm $\leq ct \leq$ 0.08 cm. (c) 0.08 cm $\leq ct \leq$ 0.12 cm. (d) 0.12 cm $\leq ct \leq$ 0.16 cm. (e) 0.16 cm $\leq ct \leq$ 0.20 cm. The vertical scale is events per 2 MeV/c^2.

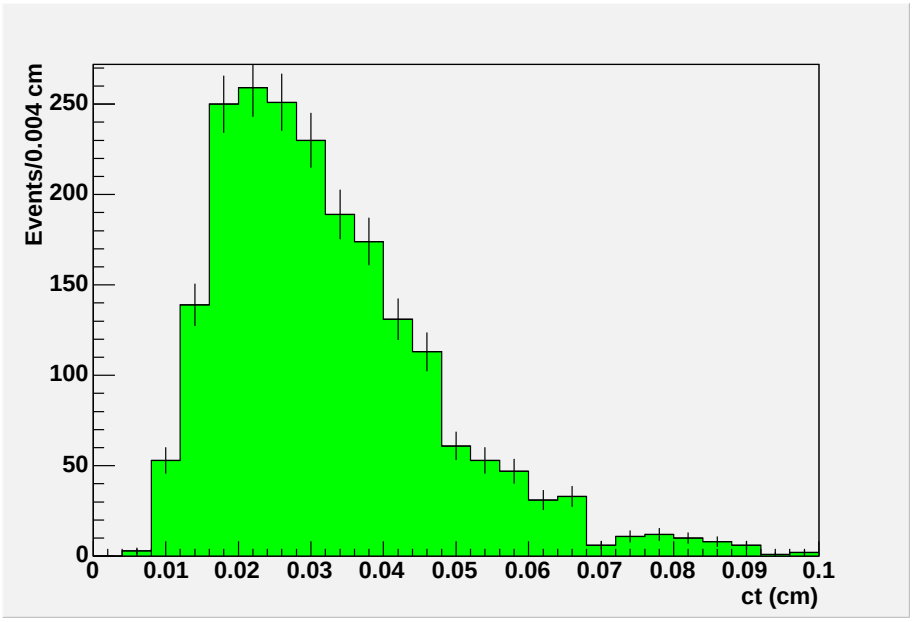

Figure 3.3. Measured $ct(D^0)$ for the mass range 1.835 GeV/c^2 to 1.895 GeV/c^2. The odd shaped non-exponential distribution is due to trigger bias.

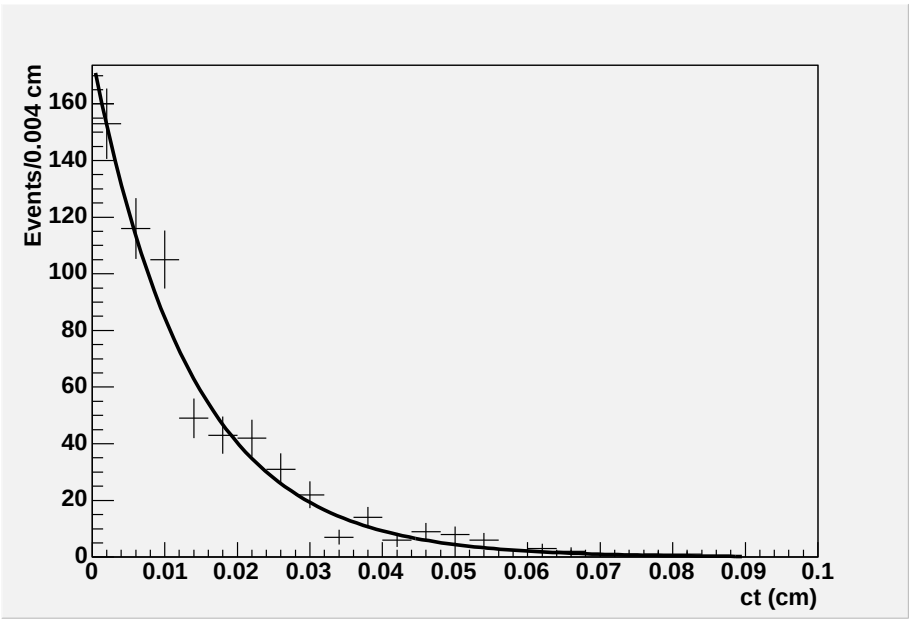

Figure 3.4. The ct distribution for the mass range 1.835 GeV/c^2 to 1.895 GeV/c^2, with trigger bias removed. The curve shows the exponential fit.

find that the background in the central mass range is approximately 180 entries.

3.5 Mean Lifetime

The background for each of the 25 bins of the *ct* distribution is equal to the average of number of entries in each bin of range 1 and range 3. The background subtracted *ct* distribution is shown in Figure 4.5. We apply the exponential fit given by the exponential decay law,

$$N(t) = N_0 e^{-\lambda t}, \tag{3.1}$$

where $N(t)$ is the number of remaining particles at time t, N_0 is the proportionality constant and λ is called the disintegration constant which is related to the mean lifetime by the relation $\tau = 1/\lambda$ and has unit of inverse time. The fit parameter λ gives the mean lifetime.

The box in the figure shows the fit parameters. The first parameter χ^2 is the goodness-of-fit, in this case it is given by

$$\chi^2 = \sum_{i=1}^{25} [\frac{N - N_{predicted}}{\sigma}]^2$$

where N is the number of entries, $N_{predicted}$ is the number predicted from the fit function and σ is the error equal to $\sqrt{N}$. The parameters ndf, Prob, N_0 and, λ, are as defined earlier. Small probability indicates a poor fit. To improve the quality of fit, we set $\sigma = \sqrt{N_{signal} + N_{background}}$ instead of $\sigma = \sqrt{N_{signal}}$. This gives χ^2/ndf = 31.77/16 and Prob = 0.01071, which is still a poor fit. Further investigation is required, but for the current analysis, we use this fit.

We calculate the mean lifetime τ from the fit parameters as follows. From $\tau = \frac{1}{\lambda}$, if the error in λ is $\delta\lambda$ then the error in the lifetime τ is

$$\delta\tau = \frac{d\tau}{d\lambda}\delta\lambda = \frac{1}{\lambda^2}\delta\lambda,$$

and so the lifetime τ is

$$\tau = \frac{1}{\lambda} \pm \frac{1}{\lambda^2}\delta\lambda.$$

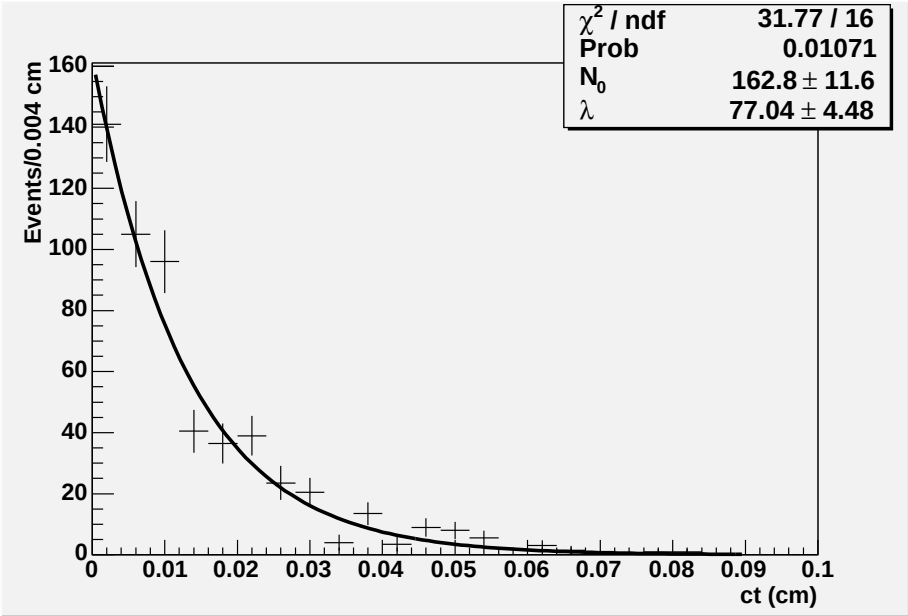

Figure 3.5. RS D^0 lifetime in $c\tau$ for the mass range 1.835 GeV/c^2 to 1.895 GeV/c^2, with background subtraction.

From the measured fit parameters (Figure 4.5), $\lambda = 77.04 \pm 4.48$, and so,

$$\tau = \frac{1}{77.04} \pm \frac{1}{77.04^2} \times 4.48 = 0.01298 \pm 0.00075 \text{ cm}.$$

3.6 Conclusion

In summary, we analyzed the decay time distribution of D^0 mesons using the CDF data. The data was collected with 0.35 fb^{-1}. After reducing the background, we applied an exponential fit and measured the decay constant λ. From λ we calculated the mean lifetime. We find that the mean lifetime of D^0 mesons $c\tau = 0.01298 \pm 0.00075$ cm which is comparable to the PDG world average 0.01230 ± 0.000045 cm. This result can be refined further by including systematic errors. The result shows that the CDF data can be used for mixing analysis.

Appendix A

DESCRIPTION OF LEAFS

The ROOT file used for the decay time distribution analysis is called "dstar_01_1.root". In this file, the folder called DFINDER under the module name "DStarD0p1AnalModule", has all the leafs required for the D^0 decay time analysis. The following is the list of leafs with description.

Leaf name	Description
DS_Chi2xy	$\chi^2_{xy}(D^*)$
DS_Chi2	$\chi^2(D^*)$
DS_C2xy_D0	$\chi^2_{xy}(D^0)$
DS_C2_D0	$\chi^2(D^0)$
DS_Mass	right-sign m(D^*)
DS_Mass_D0	right sign m($K\pi$)
WS_Mass	wrong-sign m(D^*)
WS_Mass_D0	wrong-sign m($K\pi$)
DS_Pt	transverse momentum of D^*
DS_Pt_D0	transverse momentum of D^0
DS_Ip_D0	impact parameter
DS_Lxy	distance between z-axis and the vertex
DS_eLxy	error in L_{xy} for D^*
DS_Lxy_D0	L_{xy} for D^0
DS_eLxy_D0	error in L_{xy} for D^0
DS_ct	decay time for D^* in cm
DS_ct_D0	decay time for D^0 in cm
DS_px	x component of momentum vector for D^*
DS_py	y component of momentum vector for D^*
DS_pz	z component of momentum vector for D^*
DS_Ptot	total momentum for D^*
DS_px_D0	x component of momentum vector for D^0
DS_py_D0	y component of momentum vector for D^0
DS_pz_D0	z component of momentum vector for D^0
DS_Ptot_0	total momentum for D^0

Table A.1. Description of Leafs in DStar_01_1.root file.

Appendix B

ROOT/C++ Programs for Decay Time Analysis

B.1 ROOT Program to Generate ScatterPlot

```
/*************************************************************
* Module Name: scatterplot.c
* Author: Nagesh Kulkarni
* Date:   June 2005
* Last modified: April 3, 2006
* Copyright:
*
* Purpose:  This program generates the mass vs ct scatter plot.
*
*************************************************************/


void scatterplot()
{
   gROOT->Reset();

   TCanvas* c1 = new TCanvas("c1","c1",119,33,699,499);

   TChain chain("DStarD0piAnalModule/DFINDER");
   chain.Add("dstar_01_1.root");

   gStyle->SetOptStat(kFALSE);

   TCut cut1("DS_Chi2xy <= 12");
   TCut cut2("abs(DS_eLxy) <= 15");
   TCut cut3("abs(D0[2])<0.08");
   TCut cut4("DxPPi[2] <2.25");
   TCut cut5("DS_Lxy_D0/DS_eLxy_D0 >= 5");

   TCut cut6("DS_ct_D0 > 0.00");
   TCut cut7("DS_ct_D0 < 0.04");


   TH2F* h = new TH2F("Scatter plot of D0_ct Vs D0_Mass",""
                      ,100,1.7,2.0,100,0.0,0.25);
   h->SetFillColor(kGreen);
   h->SetLabelSize(0.03,"x");
   h->GetXaxis()->SetTitle("D^{0} Mass (GeV/c^{2})");
   h->GetYaxis()->SetTitle("ct (cm)" );
   h->GetYaxis()->SetTitleOffset(1.20);



     chain.Draw("DS_ct_D0:DS_Mass_D0 >> Scatter plot of D0_ct Vs D0_Mass",
                cut1 + cut2 + cut3 + cut4 + cut5);

}
```

B.2 Program to Generate D^0 Mass for a ct range

```
/*************************************************************
* Module Name: D0Massfor_ct_range.c
* Author: Nagesh Kulkarni
* Date:   June 2005
```

```
* Last modified: April 3, 2006
* Copyright:
*
* Purpose:  This program generates the D0 mass distribution for.
*           various ct intervals.
*
*****************************************************************/



D0Massfor_ct_range()
{
   // Clean the global objects.
   gROOT->Reset();

   TCanvas* c1 = new TCanvas("c1","c1",119,33,699,499);

   // The required leafs for data analysis of D0 mesons are in the folder DFINDER.

   TChain chain("DStarD0piAnalModule/DFINDER");
   chain.Add("dstar_01_1.root");

   gStyle->SetOptStat(kFALSE);

  // Apply standard cuts.
   TCut cut1("DS_Chi2xy <= 12");
   TCut cut2("abs(DS_eLxy) <= 15");
   TCut cut3("abs(D0[2])<0.08");
   TCut cut4("DxPPi[2] <2.25");
   TCut cut5("DS_Lxy_D0/DS_eLxy_D0 >= 5");

  // Generate all histograms.
   TString llimit = "0.04";
   TString ulimit = "0.08";
   TString s1 = "DS_ct_D0 >" + llimit;
   TString s2 = "DS_ct_D0 <" + ulimit;

   TCut cut6(s1);
   TCut cut7(s2);
   TCut cut_1("DS_ct_D0 > 0.00");
   TCut cut_2("DS_ct_D0 < 0.04");
   TCut cut_3("DS_ct_D0 > 0.04");
   TCut cut_4("DS_ct_D0 < 0.08");
   TCut cut_5("DS_ct_D0 > 0.08");
   TCut cut_6("DS_ct_D0 < 0.12");
   TCut cut_7("DS_ct_D0 > 0.12");
   TCut cut_8("DS_ct_D0 < 0.16");
   TCut cut_9("DS_ct_D0 > 0.16");
   TCut cut_10("DS_ct_D0 < 0.20");


   Int_t numberofbins = 150;
   Axis_t xmin  = 1.7;
   Axis_t xmax  = 2.0;
   Float_t tempfloat = (xmax - xmin)*1000/numberofbins;
   cout << "One bin =" << tempfloat << "MeV" << "\n";
   TString numofevents;
   numofevents += tempfloat;
   TString name = "DStar_Mass_D0";
   TString title = "RS D^{0} Mass with" + llimit + "< ct <" + ulimit;
   TString hist = "DS_Mass_D0";
   TString units = "MeV/c^{2}";

   //Fill the chain to draw
   c = new TCanvas ("c","My canvas",200,10,800,700);
   c->Divide(2,3);

   TH1F* h1 = new TH1F("h1","a",numberofbins,xmin,xmax);
```

```
   TH1F* h2 = new TH1F("h2","b",numberofbins,xmin,xmax);
   TH1F* h3 = new TH1F("h3","c",numberofbins,xmin,xmax);
   TH1F* h4 = new TH1F("h4","d", numberofbins,xmin,xmax);
   TH1F* h5 = new TH1F ("h5","e", numberofbins,xmin,xmax);

   h1->SetFillColor(kGreen);
   h1->GetXaxis()->SetTitle("D^{0} Mass (GeV/c^{2})");
   h2->SetFillColor(kGreen);
   h2->GetXaxis()->SetTitle("D^{0} Mass (GeV/c^{2})");
   h3->SetFillColor(kGreen);
   h3->GetXaxis()->SetTitle("D^{0} Mass (GeV/c^{2})");
   h4->SetFillColor(kGreen);
   h4->GetXaxis()->SetTitle("D^{0} Mass (GeV/c^{2})");
   h5->SetFillColor(kGreen);
   h5->GetXaxis()->SetTitle("D^{0} Mass (GeV/c^{2})");


   c->SetGridx();
   c->SetGridy();
   c->cd(1);
   chain.Draw(hist + ">>" + "h1", cut1 + cut2 + cut3 + cut4 + cut5 + cut_1 + cut_2);
   c->cd(2);
   chain.Draw(hist + ">>" + "h2", cut1 + cut2 + cut3 + cut4 + cut5 + cut_3 + cut_4);
   c->cd(3);

   chain.Draw(hist + ">>" + "h3", cut1 + cut2 + cut3 + cut4 + cut5 + cut_5 + cut_6);
   c->cd(4);
   chain.Draw(hist + ">>" + "h4", cut1 + cut2 + cut3 + cut4 + cut5 + cut_7 + cut_8);
   c->cd(5);
   chain.Draw(hist + ">>" + "h5", cut1 + cut2 + cut3 + cut4 + cut5 + cut_9 + cut_10);
}
```

B.3 Program for Decay Time Distribution

```
/*****************************************************************
* Module Name: DecayTime.c
* Author: Nagesh Kulkarni
* Date:   June 2005
* Last modified: April 3, 2006
* Copyright:
*
* Purpose:  This program generates Decay time distribution.
*
*****************************************************************/


void DecayTime()
{
  /*
   This program is to generate histogram for ct distribution.
   We create three 'ct histograms' for the following ranges of masses.

   ct1 for m(D0) = 1.775 to 1.835 GeV
   ct2 for m(D0) = 1.835 to 1.895 GeV
   ct3 for m(D0) = 1.895 to 1.955 GeV

   Each of the ct histogram has 25 bins in it.

   To subtract background we subtract average of bincontents of ct1
   and ct3 from the bincontent of ct2
   Then generate ct4 as a ct distribution without background

   The same program can be used to generate with bkg and bkg
   subtracted histograms, by just toggling the integer bkgsubtract .
```

```
  */

   gROOT->Reset();

   TCanvas* c1 = new TCanvas("c1","c1",119,33,699,499);

   TChain chain("DStarD0piAnalModule/DFINDER");
   chain.Add("dstar_01_1.root");

   gStyle->SetOptStat("");
   gStyle->SetOptFit(1111);

   TCut cut1("DS_Chi2xy <= 10");
   TCut cut2("abs(DS_eLxy) <= 15");
   TCut cut3("abs(D0[2])<0.056");
   TCut cut4("DxPPi[2] <2.25");
   TCut cut5("DS_Lxy_D0/DS_eLxy_D0 >= 4");
   TCut cut6("abs(DS_Ip_D0)< 0.01");
   TCut cut7("DxPK[0] < 1.2 ");


   TCut cut_1("DS_Mass_D0 > 1.775");
   TCut cut_2("DS_Mass_D0 < 1.835");
   TCut cut_3("DS_Mass_D0 > 1.835");
   TCut cut_4("DS_Mass_D0 < 1.895");
   TCut cut_5("DS_Mass_D0 > 1.895");
   TCut cut_6("DS_Mass_D0 < 1.955");


   TH1F* ct2 = new TH1F("ct2","",25,0.000,0.1);//central mass range
   ct2->GetXaxis()->SetTitle("ct (cm)");
   ct2->GetYaxis()->SetTitle("Events/0.004 cm");
   ct2->GetXaxis()->SetLabelSize(0.04);

   TH1F* ct1 = new TH1F("ct1","",25,0.00,0.1);//Left mass range
   TH1F* ct3 = new TH1F("ct3","",25,0.00,0.1);//Right mass range

   //Fill these histograms with real data.

chain.Draw("DS_ct_D0 - 0.038 >> ct1",cut1 + cut2 + cut3 + cut4
                          + cut5 + cut6 + cut7 + cut_1 + cut_2);
chain.Draw("DS_ct_D0 - 0.038 >> ct2",cut1 + cut2 + cut3 + cut4
                           + cut5 + cut6 + cut7 + cut_3 + cut_4);
chain.Draw("DS_ct_D0 - 0.038 >> ct3",cut1 + cut2 + cut3 + cut4
                           + cut5 + cut6 + cut7 + cut_5 + cut_6);


TH1F* ct4 = new TH1F("ct4","",25,0.00,0.1);
//Central mass range-this will be filled with bkg subtracted data .
    ct4->SetLabelSize(0.02,"x");
    ct4->GetXaxis()->SetTitle("ct (cm)");
    ct4->GetYaxis()->SetTitle("Events/0.004 cm");
    ct4->GetXaxis()->SetLabelSize(0.04);


    //ct4->SetFillColor(kBlue);

      Float_t  N1,Nav;

       for(Int_t i =1; i<26; i++)
       {

  N1 = ct2->GetBinContent(i);
  Nav = ( ct1->GetBinContent(i) + ct3->GetBinContent(i))/2;

         ct4->SetBinContent(i,N1-Nav);

       }
```

```
 Int_t bkgsubtract = 0;//0 means no subtraction
if(bkgsubtract ==0)
 {
  gStyle->SetOptStat(kFALSE);
  gStyle->SetOptFit(kFALSE);
  f1 = new TF1("f1","[0]*TMath::Exp(-[1]*x)",0.00,0.09);
  f1->SetParameter(0,170);
  f1->SetParameter(1,80);
  ct2->Fit("f1","R + I");
  ct2->Draw("e");

 }
 else
 {
  f1 = new TF1("f1","[0]*TMath::Exp(-[1]*x)",0.00,0.09);
  f1->SetParameter(0,170);
  f1->SetParameter(1,80);
  f1->SetParName(0,"N_{0}");
  f1->SetParName(1,"#lambda");
  ct4->Fit("f1","R + I");
  ct4->Draw("e");
 }
}
```

Appendix C

ROOT/C++ programs for Simulation of $D^0 - \bar{D}^0$ Mixing

C.1 Program to Generate Wrong-sign Distribution

```
/*****************************************************************
* Module Name: NWS.c
* Author: Nagesh Kulkarni
* Date:   June 2005
* Last modified: April 3, 2006
* Copyright:
*
* Purpose:  This program generates wrong-sign distribution.
*
*****************************************************************/

void NWS()
{
   gROOT->Reset();

   TCanvas* c1 = new TCanvas("c1","c1",119,33,699,499);
   c1->SetFillColor(18);

   //define these variables for readability.
   Float_t xmin = 0.0;
   Float_t xmax = 10.0;
   Int_t bins = 50;
   Float_t binwidth = (xmax-xmin)/bins;
   Float_t xprime_sq = 0.5/1000;
   Float_t yprime = 0.00;
   Float_t otherterms = sqrt(0.0036)*yprime + (xprime_sq + yprime*yprime)/2;

   //define formula for N_WS(t).
   //we use FixParameter() function because TF1(--)
   //does not allow variables.
   //It takes only constant and x as only one variable.

   f1 = new TF1("f1","(650/(0.0036 + [2] ))*exp(-x)*(0.0036 +
      sqrt(0.0036)*[1]*x + (([0] + [1]*[1])/4)*x*x)", xmin,xmax);
   f1->FixParameter(0,xprime_sq);
   f1->FixParameter(1,yprime);
   f1->FixParameter(2,otherterms);

   //Generate random distribution for N_WS(t) using f1.
   TH1D* h1 = new TH1D("h1","N_{WS}(t)",bins,xmin,xmax);

   cout << "bandwidth =" << binwidth << endl;
   cout << "Integral =" << f1->Integral(xmin,xmax) << endl;
   Float_t randomh1 =  f1->Integral(xmin,xmax);
   randomh1 = randomh1/binwidth;
   cout << "randomh1 =" << randomh1 << endl;
   h1->FillRandom("f1",randomh1);

  //Fit the distribution using the same fit function
  f2 = new TF1("f2","(650/(0.0036 + [2] ))*exp(-x)
   *(0.0036 + sqrt(0.0036)*[1]*x + (([0] + [1]*[1])/4)*x*x)", xmin,xmax);
  f2->SetParameter(0,xprime_sq);
  f2->SetParameter(1,yprime);
  f2->FixParameter(2,otherterms);
```

```
  f2->SetParName(0,"x'");
  f2->SetParName(1,"y'");

  gStyle->SetOptStat(kFALSE);
  gStyle->SetOptFit(kFALSE);
  h1->GetXaxis()->SetTitle("t");
  h1->GetYaxis()->SetTitle("N_{WS}(t)");

  h1->Fit("f2","B + R + I");
  h1->Draw("E");

  Int_t n = h1->Integral()*binwidth;
  cout << "Number of events=" << n  << endl;

}
```

C.2 Program to Generate Right-sign Distribution

```
/*************************************************************
* Module Name: NRS.c
* Author: Nagesh Kulkarni
* Date:   June 2005
* Last modified: April 3, 2006
* Copyright:
*
* purpose:  This program generates right-sign distribution.
*
*************************************************************/


void NRS()
{
   gROOT->Reset();

   TCanvas* c1 = new TCanvas("c1","c1",119,33,699,499);
   c1->SetFillColor(18);

   //define these variables for readability.
   Float_t xmin = 0.0;
   Float_t xmax = 10.0;
   Int_t bins = 50;
   Float_t binwidth = (xmax-xmin)/bins;
   Float_t xprime = 0.5/1000;
   Float_t yprime = 0.00;
   Float_t otherterms = sqrt(0.0036)*yprime + (xprime*xprime + yprime*yprime)/2;

   //formula for N_RS(t)
   f1 = new TF1("f1","(650/(0.0036 + [0]))*exp(-x)", xmin,xmax);
   f1->FixParameter(0,otherterms);

   //Generate random distribution for N_RS(t) using f1
   TH1D* h2 = new TH1D("h2","N_{RS}(t)",bins,xmin,xmax);

   Float_t randomh1 =  f1->Integral(xmin,xmax);
   randomh1 = randomh1/binwidth;
   h2->FillRandom("f1",randomh1);

  f2 = new TF1("f2","[0]*TMath::Exp(-[1]*x)",xmin,xmax);
  f2->SetParameter(0,(650/(0.0036 + otherterms)));
  f2->SetParameter(1,1.0);
  f2->SetParName(0,"N_{0}");
  f2->SetParName(1,"#lambda");

  gStyle->SetOptStat("");
  gStyle->SetOptFit(1111);
```

```
  h2->GetXaxis()->SetTitle("t");
  h2->GetYaxis()->SetTitle("N_{RS}(t)");
  h2->Fit("f2","B + R + I + Q");
  h2->Draw("E");

  Int_t n = h2->Integral()*binwidth;
  cout <<"Constant=" << (650/(0.0036 + otherterms)) << endl;
  cout << "Number of events=" << n  << endl;

}
```

C.3 Program to Generate WS/RS Ratio

```
/*************************************************************
* Module Name: Rt.c
* Author: Nagesh Kulkarni
* Date:   June 2005
* Last modified: April 3, 2006
* Copyright:
*
* Purpose:  This program generates r(t), the WS/RS ratio.
*
*************************************************************/


void Rt()
{
  // Reset global variables.
   gROOT->Reset();

   Float_t xmin = 0.0;
   Float_t xmax = 10.0;
   Float_t xprime_sq =0.5/1000;
   Float_t yprime = 0.00;
   Int_t bins = 50;
   Float_t binwidth = (xmax-xmin)/bins;
   Float_t otherterms = sqrt(0.0036)*yprime + (xprime_sq + yprime*yprime)/2;

   //Formula for wrong-sign distribution
   f1  = new TF1 ("f1", "(650/(0.0036 + [2]))
                         *exp(-x)*(0.0036 +
                          sqrt(0.0036)*[1]*x +
                         (([0] + [1]^2)/4)*x*x) ",xmin, xmax);
   f1->FixParameter(0,xprime_sq);
   f1->FixParameter(1,yprime);
   f1->FixParameter(2,otherterms);

   //Formula for right-sign distribution
   f2  = new TF1 ("f2", "(650/(0.0036 + [0]))*exp(-x)",xmin, xmax);
   f2->FixParameter(0,otherterms);

   TH1D* h1 = new TH1D("h1","T(t)(ws)",bins,xmin,xmax);
   TH1D* h2 = new TH1D("h2","T(t)(rs)",bins,xmin,xmax);
   TH1D* h3 = new TH1D("h3","R(t)",bins,xmin,xmax);

   // Generate WS and RS distributions
   Float_t randomh1,randomh2,h1Integral,h2Integral;
   randomh1 = f1->Integral(xmin,xmax);
   randomh2 = f2->Integral(xmin,xmax);

   randomh1 = randomh1/binwidth;
   randomh2 = randomh2/binwidth;

   h1->FillRandom("f1",randomh1);
   h2->FillRandom("f2",randomh2);
```

```
   h3->Sumw2();

   // Take a ratio
   h3->Divide(h1,h2);

   h3->GetXaxis()->SetTitle("t");
   h3->GetYaxis()->SetTitle("R(t)");

  // fit using polynomial of second order
   f3 = new TF1("f3","pol2",xmin,xmax);
   f3->FixParameter(0,0.0036);
   f3->SetParameter(1,sqrt(0.0036)*yprime);
   f3->SetParameter(2,(xprime_sq + yprime*yprime)/4);
   h3->SetMaximum(0.05);
   h3->SetMinimum(0.00);

   //Draw the histogram.
   TCanvas* c1 = new TCanvas("c1","c1",119,33,699,499);
   c1->SetFillColor(18);
   gStyle->SetOptFit(1100);
   gStyle->SetOptStat("");

   h3->Fit("f3","B + R + I");


}
```

C.4 Program to Generate 95% C.L. Ellipse

```
/*************************************************************
* Module Name: CL.c
* Author: Nagesh Kulkarni
* Date:   June 2005
* Last modified: April 3, 2006
* Copyright:
*
* Purpose:  This program generates 95% CL ellipse in x'^2- y' plane.
*
*************************************************************/

void CL()
{
   // Clean all objects.
   gROOT->Reset();

   // The following variables are required in the equations.
   Float_t xmin = 0.0;
   Float_t xmax = 10;
   Float_t xprime_sq = 0.5/1000;
   Float_t yprime = 0.00;
   Int_t bins = 10;
   Float_t binwidth = (xmax-xmin)/bins;
   Float_t otherterms = sqrt(0.0036)*yprime + (xprime_sq + yprime*yprime)/2;

   //Equation (2.3) used to generate the wrong-sign distribution.
   f1  = new TF1 ("f1", "(650/(0.0036 + [2]))
                 *exp(-x)*(0.0036 + sqrt(0.0036)*[1]*x +
                 (([0] + [1]^2)/4)*x*x) ",xmin, xmax);
   f1->FixParameter(0,xprime_sq);
   f1->FixParameter(1,yprime);
   f1->FixParameter(2,otherterms);

   // Equation (2.4) used to generate right-sign distribution
```

```
f2  = new TF1 ("f2", "(650/(0.0036 + [0]))*exp(-x)",xmin, xmax);
f2->FixParameter(0,otherterms);

// Define the histograms to be generated
TH1D* h1 = new TH1D("h1","T(t)(ws)",bins,xmin,xmax);
TH1D* h2 = new TH1D("h2","T(t)(rs)",bins,xmin,xmax);
TH1D* h3 = new TH1D("h3","R(t) vs t",bins,xmin,xmax);
TGraph* dt = new TGraph(); // This is used to draw the ellipse.

//To generate random numbers....
Float_t randomh1,randomh2,h1Integral,h2Integral;
randomh1 = f1->Integral(xmin,xmax);
randomh2 = f2->Integral(xmin,xmax);

randomh1 = randomh1/binwidth;
randomh2 = randomh2/binwidth;

// Now generate the WS and RS distributions.
h1->FillRandom("f1",randomh1); // generate the distribution for formula f1
h2->FillRandom("f2",randomh2); // generate the distribution for formula f2

//Here we set the flag for the third histogram to specify how to calculate errors.
h3->Sumw2();

// Take the ratio
h3->Divide(h1,h2);

// Fit the ratio with polynomial of second order.
// Fit it for p0, p1 and p2 as adjustable parameters.
f3 = new TF1("f3","pol2",xmin,xmax);
f3->SetParameter(0,0.0036);
f3->SetParameter(1,sqrt(0.0036)*yprime);
f3->SetParameter(2,(xprime_sq + yprime*yprime)/4);

h3->Fit("f3","B + R + I + Q");

//Define a canvas to draw the ellipse.
TCanvas* c1 = new TCanvas("c1","c1",119,33,699,499);
c1->SetFillColor(18);

// Get the error-matrix for polynomial fit function.
fitter = TVirtualFitter::GetFitter();
TMatrixD matrix(3,3,fitter->GetCovarianceMatrix());
cout << "matrix:" << endl;
matrix.Print();

Float_t  spy = sqrt(matrix[1][1]);
Float_t spx = sqrt(matrix[2][2]);
Float_t sxsyp = (spx*spy);
Float_t  rho = matrix[1][2]/sxsyp;

/* Calculate the errors in x'^2 and y' from the error matrix.
   This gives sigma_x and sigma_y. */

Float_t p1 = f3->GetParameter(1);
Float_t p2= f3->GetParameter(2);
Float_t y_prime = p1/sqrt(0.0036);
Float_t x_prime_sq = 4*p2 -y_prime*y_prime;
Float_t error_in_p1 =spy;
Float_t error_in_p2 = spx;
Float_t s2 = error_in_p1/sqrt(0.0036);
Float_t s1 = sqrt(16*error_in_p2*error_in_p2 + 4*yprime*yprime*s2*s2);

TGraph*dt = new TGraph();

/*
  The equation of ellipse is
```

```
      1        1       2*rho*x*y       (1-rho^2)(1.64)^2
     ----  + ----- - ------------  =  --------------------
      sx^2    sy^2     sx * sy                2

      using polar coordinates, calculate the points p(x,y) and plot by setting each point.
      Divide 2Pi in 5000 divisions. For fraction of angle, calculate r and using
      trigonometric relations, find x and y coordinates.
   */

    Float_t xc,yc,u1,u2,r,phi,theta; //center
    Int_t points = 5000;
    Double_t x1,y1,dphi(2*TMath::Pi()/points);
    xc = 0.0;
    yc = 0.0;
    Double_t LHS = (1-rho*rho)*1.64*1.64/2;
    for(Int_t i =0; i< points;i++)
    {
       theta= i*dphi;
       phi= atan2(s2*sin(theta),s1*cos(theta));
       u1 = cos(phi)/s1;
       u2 = sin(phi)/s2;

       r = sqrt(LHS/(u1*u1  + u2*u2 - 2*rho*u1*u2));
       x1= xc + r*cos(phi);
       y1= yc + r*sin(phi);
       dt->SetPoint(i,x1,y1);
       if(i == 0)
       {
        dt->GetYaxis()->SetLimits(y1,y1);
        dt->SetPoint(points, x1, y1);
       }

    }

    // Set axis title and limits.
    dt->GetXaxis()->SetLimits(0.0,0.0009);
    dt->SetTitle("95% C.L.");
    dt->GetXaxis()->SetTitle("x'^{2}");

    dt->GetYaxis()->SetTitle("y'");
    dt->GetYaxis()->SetNoExponent();
    TGaxis::SetMaxDigits(3);

    // Draw the ellipse now.
    dt->Draw("AC");

    //for debugging purpose, print these variables

    cout <<"rho=" << rho << endl;

    cout << "Error in p1="<< f3->GetParError(0) << endl;
    cout << "Error in p2=" << f3->GetParError(1) << endl;
    cout << "sqrt(Matrix[0][0])=" << sqrt(matrix[0][0]) << endl;
    cout << "sqrt(Matrix[1][1])=" << sqrt(matrix[1][1]) << endl;

}
```

BIBLIOGRAPHY

[1] S. Eidelman *et al.* (Particle Data Group), Phys. Lett. B **592** 1(2004).

[2] S. Bianco and F. L. Fabbri, Riv. Nuovo Cim. **26** N 7 (2003) 1; hep-ex/0309021 (2003).

[3] A. Falk, Y. Nir and A. Petrov, JHEP **9912** (1999) 019.

[4] B. Aubert, *et al.* (BABAR Collaboration), Phys. Rev. Lett. **91**, 171801 (2003).

[5] J. Li *et al.*, (BELLE Collaboration) Phys. Rev. Lett. **94** (2005) 071801.

[6] D. Acosta *et al.* (CDF Collaboration), Phys. Rev. D **71**, 032001 (2005).

[7] A. Sill *et al.*, Nucl. Instrum. Meth. A **447**, 1 (2000).

[8] E. Blucher *et al.*, hep-ph/0512039 (2005).

[9] L. Lyons, *Statistics for Nuclear and Particle Physicists* (Cambridge university press, 1986).

[10] W. Ashmanksas *et al.*, Nucl. Instrum. Meth. A **518**, 532 (2004).

[11] Root home page, http://root.cern.ch/.

[12] R. Blair *et al.*, FERMILAB-PUB-96/390-E (1996).

ABSTRACT

SIMULATION OF D^0-$\bar{D}^0$ MIXING AND PRELIMINARY ANALYSIS OF CDF DATA

by

NAGESH P. KULKARNI

May 2006

Advisor: Dr. Paul Karchin

Major: Physics

Degree: Master of Science

We estimate the accuracy of measuring $D^0 - \bar{D}^0$ mixing parameters in the CDF II detector by simulating the ratio of wrong sign decays to Cabibbo-favored decays. We fit the ratio distribution for x' and y' to obtain a 95% C.L. region in the $x'^2 - y'$ plane, assuming no CP violation. With the total number of doubly Cabibbo suppressed decays equal to 2100, as observed at $\sqrt{s} = 1.96$ TeV and 0.35 fb^{-1}, we find $x'^2 \leq 0.6 \times 10^{-3}$ and $-0.0035 \leq y' \leq 0.0035$. These expected limits are more restrictive than the results from other experiments. We also analyzed the decay time distribution of D^0 mesons using the CDF II data. Our value of mean lifetime is comparable to the accepted mean lifetime of D^0 mesons. This analysis shows that it is possible to establish the world's best limit on the mixing parameters using the CDF II detector.

AUTOBIOGRAPHICAL STATEMENT

NAGESH P. KULKARNI

Email:au2442@wayne.edu

Education: Master of Science (Physics)

University: Wayne State University, Detroit, MI, USA.

Research Interests:

Elementary Particle Physics.

Professional experience:

2005-2006	Research assistant, Wayne State University.
2004-2005	CMMI software quality consultant for IS&S of General Motors.
2003-2004	Senior Systems engineer on a project with State Street Inc.
1999-2003	Senior Systems engineer for Nortel networks.
1998-1999	Software engineer, Tangent software Pvt. Ltd.