DEPARTMENT OF PHYSICS

FINAL YEAR PROJECT, DISSERTATION OR PHYSICS EDUCATION REPORT

NAME:	Ben Crabbe
DEGREE COURSE:	Physics BSc
PROJECT TITLE:	NA62-Rare Kaon Decays
YEAR OF SUBMISSION:	2014
SUPERVISOR:	Dr Helen Heath
NUMBER OF WORDS:	5028

DECLARATION

This project was started by a fellow student Oliver Reardon-Smith during his internship with the Bristol Particle Physics research group during summer 2013. His report [1] was supplied to myself and my partner at the commencement of our project, he also provided us with his version of the COmPACT reader program and gave us valuable help becoming acquainted with the software. Also, he provided me with some useful python histogramming scripts which have been invaluable to both my day to day work and to the production of this report.

Since then I have built upon his attempted analysis bringing together the methods it used with others adapted from other NA62 and NA48/2 studies such as [2], [9], [27], from the ke2 analysis webpages [3] and from a couple of presentations by other scientists describing their work on this same analysis [4] [5]. These sources have been evaluated, their methods transformed into COmPACT code which has then been tested and adopted or rejected based on the results.

The COmPACT reader program used to achieve the results presented within is built upon the framework of Oliver's code but is changed and lengthened substantially. Both versions are available on `github.com/bcrabbe`.

I have drawn on the experience of my supervisor throughout the year for help on general topics and on some specific problems encountered. I have also received help via email from the NA62 2007 coordinator Dr Evgueni Goudzovski on topics specific to the dataset and to the COmPACT software.

Aside from the things mentioned above all programming, data analysis, and interpretation was my own work.

A Measurement of Charged Kaon Semileptonic Decay Braching Fractions on NA62 P5 Data

Ben Crabbe*
University of Bristol.
(Dated: May 1, 2014)

An analysis of kaon semileptonic decay ratios on NA62 (2007) data is performed. Detailed descriptions of the methods used for event selection and background reduction are presented as is the code used. The ratios of these decays has been measured as $\mathcal{R}_{K_{\mu 3}/K_{e3}} = 0.606 \pm 0.004$, $\mathcal{R}_{K_{\mu 3}/K_{2\pi}} = 0.1841 \pm 0.006$, and $\mathcal{R}_{K_{e3}/K_{2\pi}} = 0.3037 \pm 0.003$. The results are found to disagree with PDG values by around 10%. Reasons for this are discussed as are recommended directions of future work.

INTRODUCTION

In this paper we present a measurement of the ratios of the decay widths for $K^+ \rightarrow e^+ + \pi^0 + \nu_e$ ($Ke3$), $K^+ \rightarrow \mu^+ + \pi^0 + \nu_\mu$ ($K\mu3$) and $K^+ \rightarrow \pi^+ + \pi^0$ ($K2\pi$) from data taken by the CERN NA62 collaboration.

The semileptonic decays $Ke3$ and $K\mu3$ are of interest as their decay widths offer the most theoretically precise extraction of the Cabbibo-Kobayashi-Maskawa (CKM) matrix element V_{us}. For detailed theoretical background and a review of recent efforts in this field see [6] and [7].

NA62 is the latest CERN experiment studying kaon decays using a decay in flight set up. The 2007-8 period focused on the measurement of the lepton flavour violation ratio $\mathcal{R}_k$. It used predominantly the same beam and detector as the NA48/2 experiment. The period 5 data set we use here contains roughly 2 billion K^+ events 22% of which are estimated to be from one of measured channels, making this the largest data sample of its type currently in existence and therefore offering a good opportunity to improve the precision of our knowledge of these decays.

DETECTOR AND BEAMLINE

400 GeV protons from the CERN SPS are dumped into a Be target in repeated 3.8s bursts. A Beam consisting of positively charged kaons with a momentum of 74 ± 1.4 GeV is selected using dipole magnets, momentum defining slits and collimators. It enters a 114m evacuated cylindrical decay fiducial volume with an average diameter of 4mm and angular divergence of 20μrad(r.m.s). Decays in the decay volume leave the narrow beam line and enter the detector at the end, the non decaying beam particles are carried in a narrow evacuated tube through the centre of the detectors. The relevant parts of the detector are described below in the order that they appear. For a complete technical description see [8].

A magnetic spectrometer performs charged particle tracking and momentum measurements with four drift chambers (DCHs), two either side of a dipole magnet, with each DCH containing 8 planes of orthogonal sense wires.

A scintilator based Hodoscope (HOD) gives precise timing measurements on charged particles that are used primarily for triggering purposes.

A liquid Krypton electromagnetic calorimeter (LKr) supplies energy and position measurements for photons and electrons. Charged pions and muons however do not interact strongly enough to deposit their full energies. The LKr is formed of 13248 2×2 cm rectangular cells that stretch from the front, 127cm ($27X_0$) to the back. Contained inside the LKr at a depth of $9.5X_0$ is a second hodoscope (NHOD) which provides timing measurement of the electromagnetic showers and therefore of photons.

A muon detector (MUV) is located at the end of the beam line. It consists of three planes of scintilator material shielded by a 80cm thick iron wall. It provides position and timing measurements for muon identification.

TRIGGER AND RECONSTRUCTION

The triggering system used consists of 3 levels, a hardware based $L1$ trigger gives a quick indication of potentially interesting events using signals from the HOD and NHOD. This data is then reduced by the $L2$ trigger which combines more detailed signals from the HOD, DCHs and LKr. For this analysis we require the event to have passed on the $L2(ke2)$ trigger which is defined as $Q1 \times 1TRKLM \times E_{LKr}(10\ GeV)$ where: $Q1$ is a signal from the HOD indicating at least one charged particle. $1TRKLM$ is a signal from the DCHs indicating at least one track and no more than 15. $E_{LKr}(10\ GeV)$ indicates an energy deposition of at least 10 GeV in the LKr.

Following collection the raw data from each detector has been processed into a compressed format suitable for physics analysis. What follows is a simplified description of the relevant procedures.

- Consistent hits in the DCHs are grouped together to form 'tracks'. The position and slopes, i.e. $\frac{dx}{dz}$, $\frac{dy}{dz}$, before and after the magnet and the momentum, charge and DCH timings are stored.[13]
- Energy deposits in the LKr are analysed, the energies of cells within 11cm and 5 ns are summed to define a 'cluster', the position, time and energy are stored. The energy threshold for defining a cluster is $0.2\ GeV$.[15]
- The positions and directions of the tracks are extrapolated through the detector and associated to clusters and to MUV hits that are within an acceptable distance.[14]

As mentioned muons and π^+ do not deposit their full energies in the LKr, for muons this is to such an extent that there are normally no clusters associated with its track. π^+ usually deposit a more significant fraction of their energy. This has an effect on

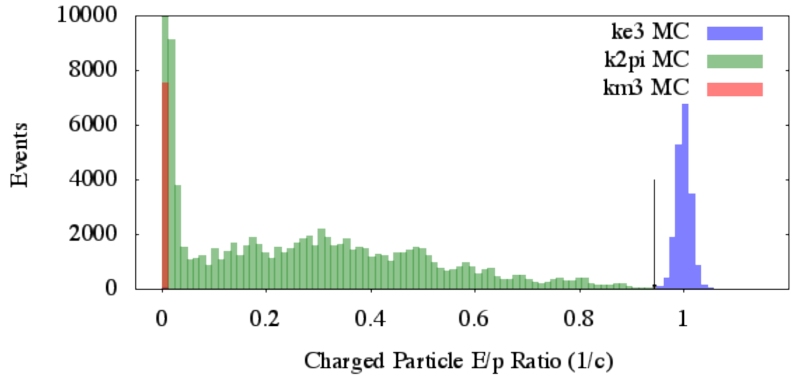


FIG. 1: Shows the energy - momentum ratio for positrons, muons and π^+ originating from simulated (see section on analysis strategy) $Ke3$, $K\mu3$, $K2\pi$ decays. The arrow, positioned at 0.943718, indicates the separation of π^+ from positrons as used for particle identification. (See section on particle ID.)

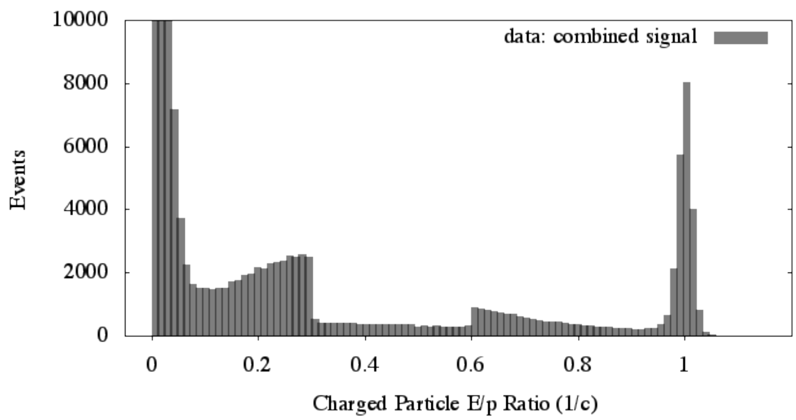


FIG. 2: Shows the energy - momentum ratio for real data after initial event selection is applied. (See common selection criteria section.)

the measured energy - momentum ratio of tracks as shown in Figure 1. This serves as a basis for particle identification in our analysis.

Following this reconstruction each event is subjected to $L3$ filters which applies loose particle identification and rejects events which do not satisfy any of the following:

- L3($Ke2$) = at least one track with $E/p \geq 0.6$ and $5 \leq p \leq 90\ GeV/c$
- $L3(Km2)$ = at least one track with $p \leq 90\ GeV/c$.
- $L3(Kmu3)$ = at least one track with $E/p \leq 0.3$ and at least two trackless clusters.
- $L3(Kpigg)$ = at least one track and at least two trackless clusters with a π^0 mass cut. (However this filter is said to have been affected by a bug which causes it to have very low efficiency and possible bias.[3])

The effect of these $L3$ triggers causes a deficit of $K2\pi$ events with E/p ratios $0.3 \leq E/p \leq 0.6$ as shown in figure 2. This may have implications for the suitability of this dataset for a $K2\pi$ measurement; it remains to be seen whether this can be corrected for with a full study of the trigger efficiencies.

Full descriptions of all triggers are found on the $Ke2$ analysis webpages.[3]

The momentum, position and slope of the beam is monitored using fully reconstructed $K \to \pi^+\pi^+\pi^-$ decays to give the average values for each run (runs are defined as a stable data taking period usually consiting of ~ 100 bursts). These values have average variations of 0.1 GeV, 1 mm and 10 μrad during each runs.[12]

ANALYSIS STRAGEGY

The goal of our analysis is to separate the raw data into sub samples that contain only $Ke3$, $K\mu3$ and $K2\pi$ decays. To do this we apply cuts/selection criteria to remove the backgrounds whilst leaving as much of the signal as possible. These cuts have been chosen and fine tuned by studying Monte Carlo simulations (MCS) of pure signal (background) events and excluding regions of low (high) rate.

These GEANT 3[10] based MCS have been produced [3] to exactly mimic each of the real data runs. Raw detector data is generated and treated using the same reconstruction procedures described above. They include appropriate fractions of radiative events (generated using the PHOTOS package [11]), those in which a photon is emmited during the decay, and events in which π^+ decay to muons. (Around 1.6% of $K2\pi$ events have such a decay inside the detector.)

However, the precise simulation of electromagnetic showers is a computationally slow process particularly for hadrons, this is avoided by using pre-generated libraries which in turn causes some disagreement between data and MCS especially in LKr measurements. Usually these effects are of the order of 10^{-6} and acceptable, but, in some cases however they become considerable. [16] One example of this is the in energy deposition of π^+ in the LKr which can be seen by comparing the shape of the distributions in figures 1 and 2.

The measurable decay products in each of our channels consists of one charged particle or 'track', and two photons (produced by the π^0 decay close enough to the primary decay vertex for them to be considered the same) which appear as 'trackless clusters' in the LKr as well as possible additional photons due to the radiative events. Since all three channels have almost the same experimental signature we first apply selection criteria common to all three channels. These serve to eliminate uninteresting decays, decays which are not fully measured, and decays which would contribute additional systematic uncertainty, and leave only a selection of combined signal events ($Ke3$, $K\mu3$ and $K2\pi$) and known sources of background. This use of selection criteria common to all three channels leads to a partial cancellation in acceptance uncertainties.[9]

We then applying particle identification on the track to distinguish between our three channels. This is based on: E/p track ratio for e^+, on parameters which distinguish 2-body from 3-body decays for π^+ and on E/p as well as MUV muon association for μ^+. We then apply further cuts on the individual signals to remove understood sources of background.

The branching ratios are then given by[9]

$$\mathcal{R}_{K_i/K_j} = \frac{Acc_{K_j} \times \epsilon_{track_{ID_j}} \times Trig_{K_j} \times N_{K_i} \times (1 + \Delta_{K_j})}{Acc_{K_i} \times \epsilon_{track_{ID_i}} \times Trig_{K_i} \times N_{K_j} \times (1 + \Delta_{K_i})} \tag{1}$$

where $i, j = e3, \mu3$ or 2π and:

- Acc_{K_j} is the acceptance of each channel. This is the fraction of events occurring in our fiducial decay volume that can actually be measured. This partly depends on the geometry of our detector and also on the cuts we have applied to data. It is measured for each signal using the MCS.

- $\epsilon_{track_{ID_j}}$ is the particle ID efficiency. This is the fraction of (for example) e^+ that are expected to be correctly identified using our particle ID criteria. Here we measure this using the MCS, however a recommended improvement to this is outlined in the discussion section.

- $Trig_{K_j}$ is the trigger efficiency. This is equal to $1-$ number of good signal events rejected by the trigger system.

- N_{K_j} is the total number of measured events.

- Δ_{K_j} is a correction for known sources of background present in the final samples. It requires knowledge of the number of non-signal events which pass all selection criteria for each channel. This is learnt from studying the MCS.

COMMON SELECTION CRITERIA

Firstly we reject events flagged as coming from a bad burst, those that were not accepted by the L2($Ke2$) trigger and any that do not contain at least one track and two trackless clusters. We then apply a small additive correction (to data only) to all clusters with an energy below 10 GeV. This is required to correct a non-linearity between the deposited and measured energy, due to the presence of $0.8X_0$ thick passive material in front of the LKr the interaction with which is poorly simulated for low energy particles in the MC, and due also to the readout threshold on LKr.[17][18]

Next we identify all the 'good' trackless clusters present in the event. Trackless clusters are defined as good if they meet all the following criteria:

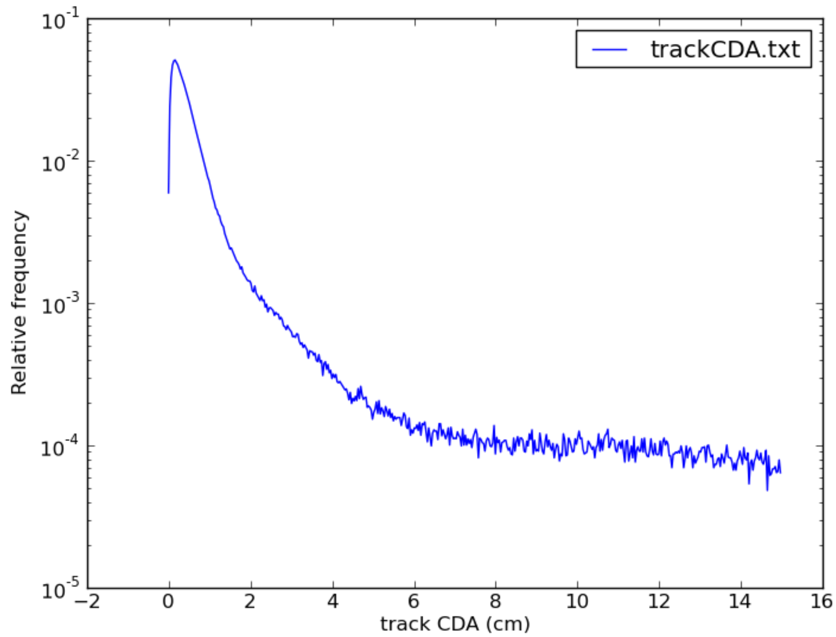


FIG. 3: The initial track CDA distribution present in data. We eliminate those $\geq$ 5cm.

- After correcting for errors in cluster position reconstruction caused by the projective geometry of the LKr (see [19]) we require the photon position at the front face have been within the acceptance of the LKr. This rejects photons hitting cells that were not online or not working correctly during the run, and those that strike too close to the edge of the detector to be fully measured.

- Have a cluster energy E $\geq$ 5 GeV. This ensures high efficiency of the $E_{LKr}(10\ GeV)$ trigger condition.

Next we aim to discard tracks not likely to have originated from a signal decay on the beam line. The tracks discarded here are mainly caused by secondary particles produced upstream of the final beam collimator and cosmic rays. Good tracks are those satisfying all of the following criteria:

- A point of closest approach algorithm is applied on the track position and slopes (measured in the DCHs prior to the spectrometer magnet) and the average beam position and slope. The closest distance of approach (CDA), whose distribution is shown in figure 3, is required to be less than 5 cm, and the z coordinate at which the CDA occurs, figure 4, must be within the decay volume i.e. between -2000 and 9000 cm as defined in the software's coordinate system.

- The position of the track at all DCHs, the LKr, and the first 2 planes of the MUV is required to be within their respective geometrical acceptances.

- The track must be positively charged.

- The track momentum, after correcting for internal misalignment in the DCHs and mis-calibration of the spectrometer magnet (see [18] and [20]) is required to be 76 GeV $\geq p_{trk} \geq$10 GeV. The upper limit excludes none kinematically plausible events, and the lower limit excludes regions of low particle ID efiicency.[21]

- The track quality, a value between 0 and 1, assigned during the reconstruction process to indicate the closeness of the track fitting, must be $\geq$ 0.7.

- The timing between the track as measured in the HOD and its associated cluster, if one exists, is required to be less the 4ns.

We then apply timing cuts, requiring every photon to be within 12 ns of at least one track. We then identify all photon pairs that are within 5ns of each other as possible $\pi^0 \to \gamma\gamma$ candidates. Every track is required to be within 5 ns of the mean time of at least

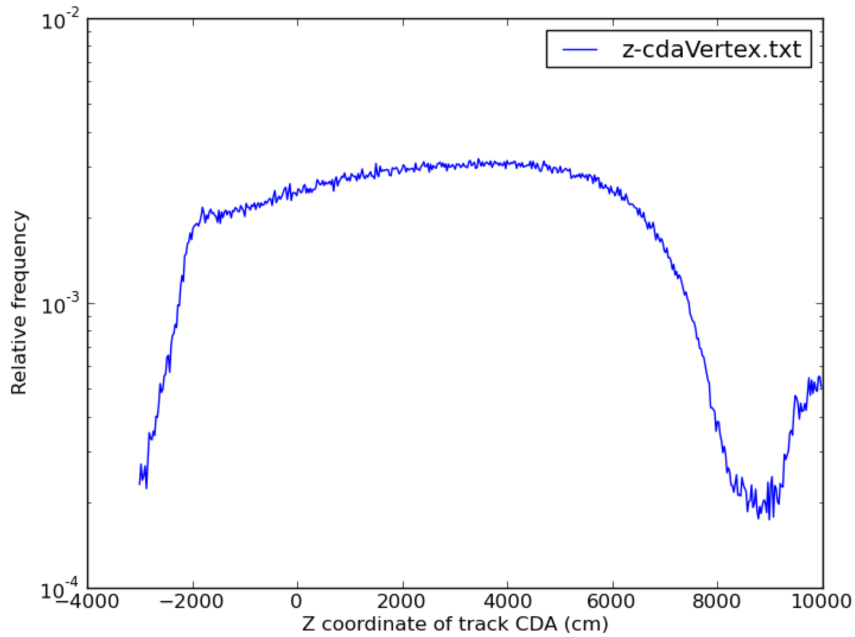


FIG. 4: The intial track z vertex of CDA distribution. We eliminate those outside the decay volume i.e. -2000 and 9000 cm.

one of these pairs. Note that timing cuts are only necessary for the data. The MCS do not contain any timing information as all particles are produced from the single decay being simulated, and everything can be assumed in time.

Next we remove any clusters or tracks whose position at the LKr front face is less than 22 cm from another in time track or cluster. This is required to exclude events whose energy measurement is affected by energy sharing effects in the cluster reconstruction procedure. Although this is supposed to be taken in to account during said procedure it was found that such events had strange behaviour when comparing charged and neutral (explained below) measures of the z coordinate of the decay vertex.

We recompute the point of closest approach between the track and beam with greater precision than before, this gives us our 'charged' decay vertex. To do this we first compute new effective slopes for the tracks[22] which take into account the effect of stray magnetic fields in the decay volume, known as the 'Blue Field' correction. These effective slopes are then used as before to compute the track-beam CDA. We then define the charged decay vertex as the point equidistant between the average beam and extrapolated track positions at this point. The improved CDA distribution is shown in figure 5, tracks with a CDA $\geq$ 3 cm are removed, as are any with a z vertex not satisfying -1600 $\leq Z_{charged} \leq$ 7000 cm. Where the acceptable decay region has been narrowed to remove events decaying close to the final collimator (as these are poorly simulated in the MC) and to remove backscattering events that can be erroneously reconstructed and bias the measurement [23]. Following this cut we require there to be no more than one acceptable track left in the event.

We then apply another correction to the energies of the track and all remaining clusters. These were developed to improve the uniformity of the LKr response across different cells and calorimeter pipeline digitisers (CPDs, responsible for the electronic readout of groups of 8x8 cells), and increase electron ID efficiency for the Ke2 analysis [24]. The implementation is described on [22].

Next we compute the 'neutral' vertex for each of the π^0 candidate pairs. This uses information from the LKr pair rather than relying on the average beam position and slope and therefore avoids the uncertainties in those values (discussed in section), By assuming π^0 mass we can compute the distance from the LKr at which the π^0 decayed to be[9]

$$d_{\pi^0} = \frac{D\sqrt{E_1 E_2}}{m_{\pi^0}} \tag{2}$$

as illustrated in figure 6. The candidate pair with the smallest $Z_{charged} - Z_{neutral}$ is chosen to be our π^0. The distribution of this measure is shown in figure 7. We do not however apply any cut as this is said to 'mix resolution with beam transversal shape and position effects' which is best avoided [4]. Events with additional remaining photons are kept so as to not reject the small fraction of radiative decays that are expected.

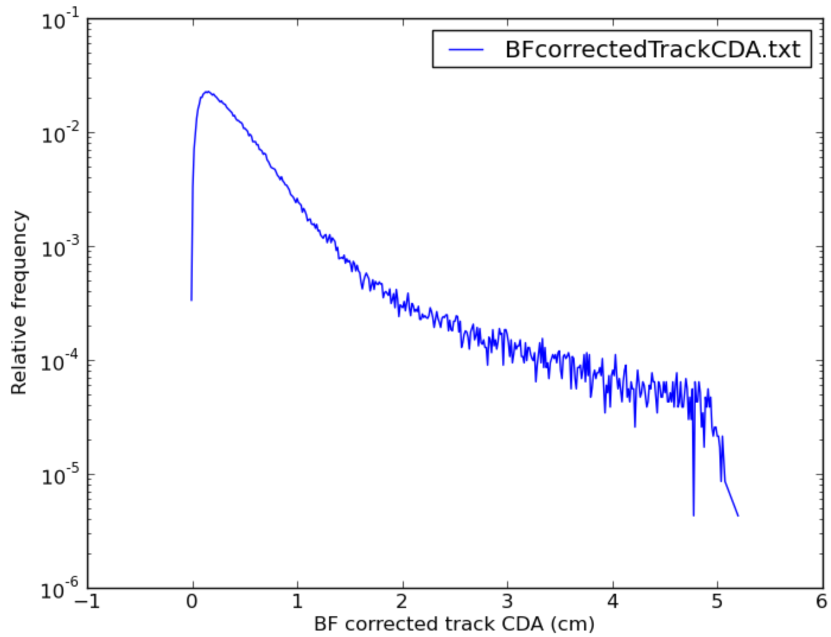


FIG. 5

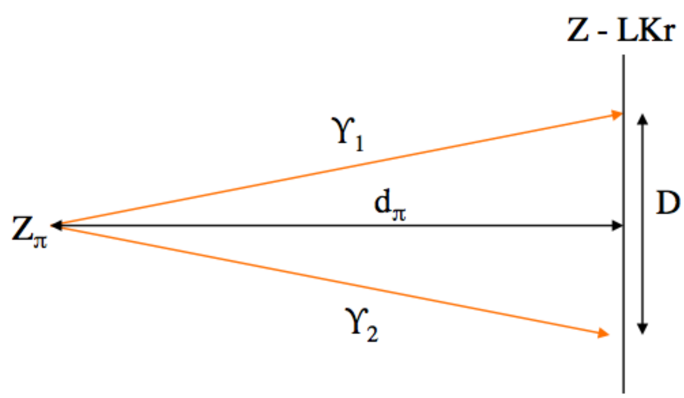


FIG. 6: Defines the parameters involved in the $Z_{neutral}$ calculation.

Finally we use the blue field corrected slopes to extrapolate the track position to the $Z_{neutral}$ plane (see figure 8) and use this as the final decay vertex for the kinematic calculations that follow. Events where the track position is at a distance greater than 3 cm from the average beam position, both extrapolated to $Z_{neutral}$, are rejected as shown in figure 9. decayVertexDiagram

The fraction of data rejected by each of these cuts is presented in table I.

PARTICLE IDENTIFICATION

A track is flagged as a muon if it passes the following criteria:

- After extrapolating its position from the (post magnet) DCHs to the MUV planes it is within a complex spacial cut of a hit in the MUV. [25].
- There is spatially consistent signal in at least the front two MUV panes. The third plane has been shown to have a reduced

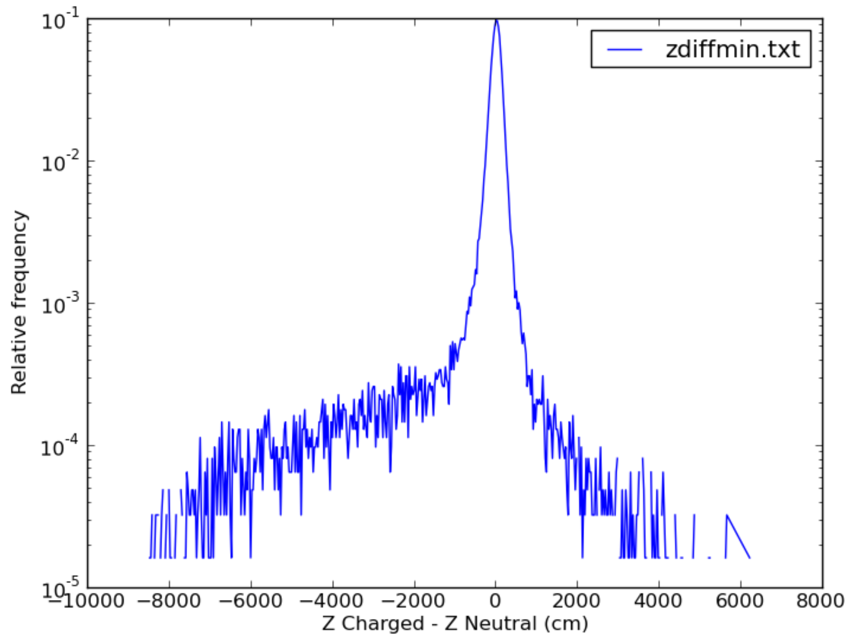


FIG. 7: Compares $Z_{charged}$ and $Z_{neutral}$ for each event. The closeness of agreement indicates that we have succeeded in correctly identifying signal events.

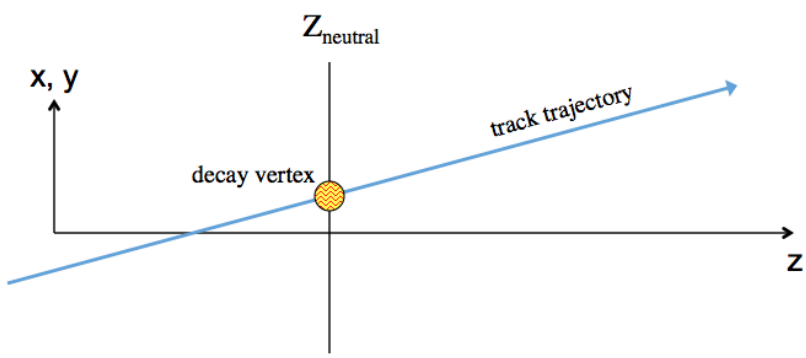


FIG. 8: Illustrates the decay vertex location preceedure.

efficiency [5] and is ignored because of this.

- The hits in the MUV are within 4.5 ns of the track time.

As displayed in figure 1 we can efficiently identify e^+ as all tracks with an $E/p \geq$0.943718 that are not flagged as muons. The ID efficiency was measured for 3 different sub samples $Ke3$ MCS data each containing 1×10^7 events. The mean value was found to be (99.620 ± 0.007) %

For π^+ ID we require that the invariant kaon mass, reconstructed under the assumption that the track is a π^+(figure 10), is within 3σ of the P.D.G. value[26] i.e. $0.4772 \leq m_{\pi^+\pi^0} < 0.5102\ GeV/c^2$. We do not reject tracks if they are flagged as muons since 1.6% of π^+ decay via $\pi^+ \rightarrow \mu^+ + \nu$ between the vertex position and the MUVs. The efficiency was measured using the same method as above to be (97.695 ± 0.016) %

For muons we require the invariant kaon mass, also reconstructed under the assumption that the track is a π^+, to be outside the range defined for the π^+. We also require a E/p ratio of less than 0.2. The efficiency was found to be $(95.89 \pm 0.02)\%$. This low

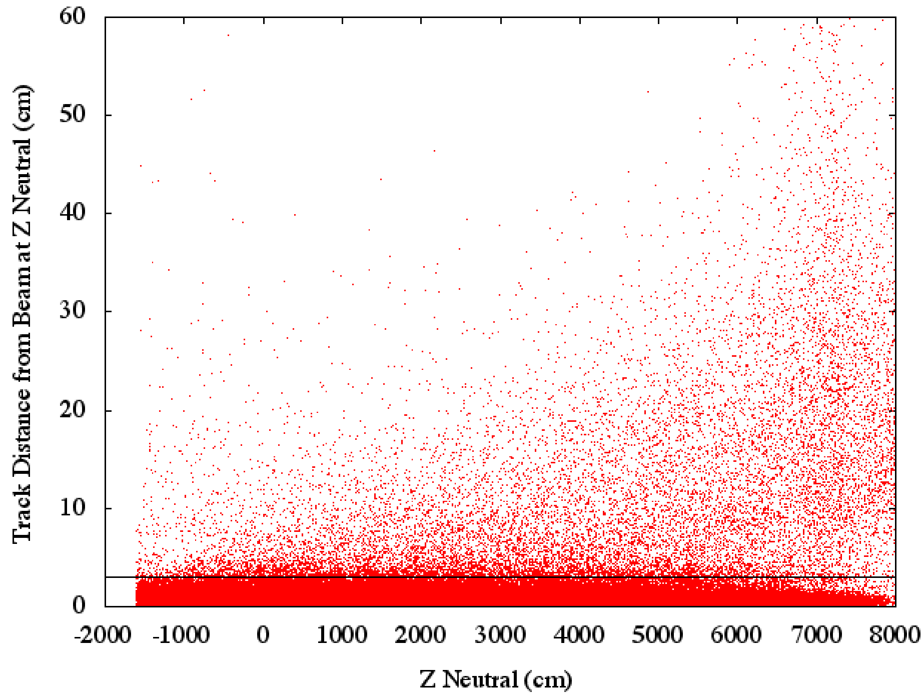


FIG. 9: Shows the distance of the final decay vertex from the average beam position. The black line is at R = 3cm, events above this are rejected.

Sample Type:	Data	
Cuts	Signal Reduction	Acceptance
Bad Burst Cut	0.01%	100.00%
L2 Trigger Cut	26.77%	73.23%
1TRK 2 Cluster Cut	23.92%	49.31%
Cluster Quality Cuts	11.59%	37.72%
Track Quality Cuts	7.48%	30.24%
Energy Sharing Cuts	4.31%	25.93%
Timing Cuts	0.15%	25.78%
No Pi0 Cut	0.02%	25.75%
Charged Z Cuts	1.57%	24.18%
Multiple Track Cut	0.13%	24.06%
Z Neutral Cut	0.59%	23.47%
Beam-Decay Vertex Cut	1.22%	22.24%
Initial Event Acceptance:		22.24%

TABLE I: Shows the fraction of real data rejected by each of the initial event selection cuts and the decrease in acceptance caused. Measured using a sample of 1×10^8 events.

efficiency is due to the fraction muons which are within the $m_{\pi^+\pi^0}$ π^+ cut, as can be seen in figure 10.

BACKGROUNDS AND ACCEPTANCE

Now we have attained separated signal samples we attempt to remove events which are incorrectly ID'd with further 'background' cuts. These cuts are applied separately to each channel. They have been optimised through study of the MCS for the signals and backgrounds.

Once the background cuts have been applied we have our final data samples, acceptances and remnant backgrounds. These

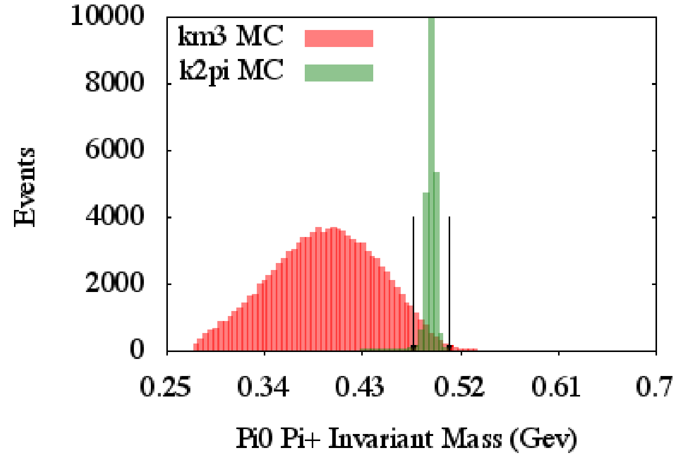
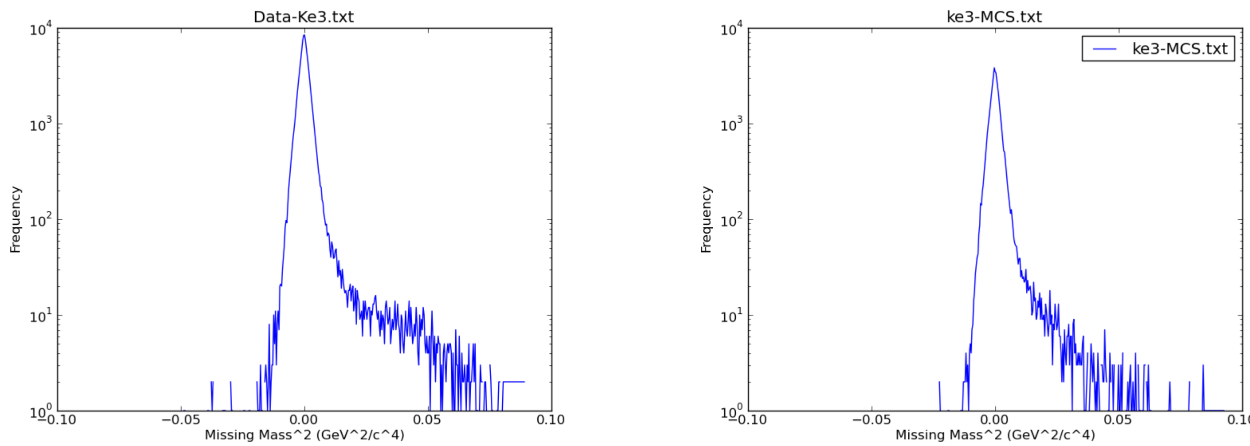


FIG. 10: Shows the distance of the final decay vertex from the average beam position. The black line is at R = 3cm, events above this are rejected.



(a) m_ν^2 for the data identified as $Ke3$.

(b) m_ν^2 for the $Ke3$ MCS.

FIG. 11: $Ke3$ m_ν^2 data - MCS comparison. The difference between the two distributions should be background. We reject events who do not satisfy $-0.0081 \geq m_\nu^2 \geq 0.0089\ GeV/c^2$ in order to retain only the regoin which is in agreement.

have been measured using the same multiple sample method as for the ID efficiency.

$Ke3$ Background

Figure 11 shows the missing mass squared, m_ν^2, distributions for $Ke3$ MCS compared with the data events identified as $Ke3$. We require $-0.0081 \geq m_\nu^2 \geq 0.0089\ GeV/c^2$ to exclude regions not in agreement. The main sources of background for this channel are $K2\pi$ and $K \rightarrow \pi^+ + \pi^0 + \pi^0$ ($K3\pi^0$) where the π^+ are misidentified as positrons. $K2\pi$ events are eliminated with a cut on events with a total measured transverse momentum $\leq 0.0294072\ GeV/c$ as shown in figure 12. $K3\pi^0$ events are reduced to a negligible level by the m_ν^2 cut as shown in figure 14.

$K\mu3$ Background

Figure 13 shows the m_ν^2 distributions for $K\mu3$. The difference between the two distributions should be background. We reject events who do not satisfy $-0.0067 \geq m_\nu^2 \geq 0.0058\ GeV/c^2$ in order to retain only the region which is in agreement. The bump

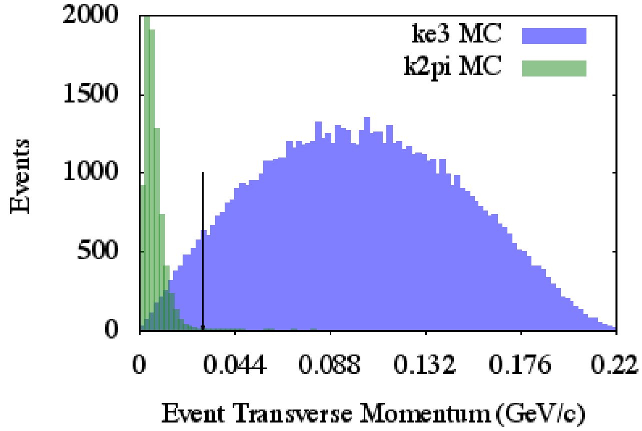
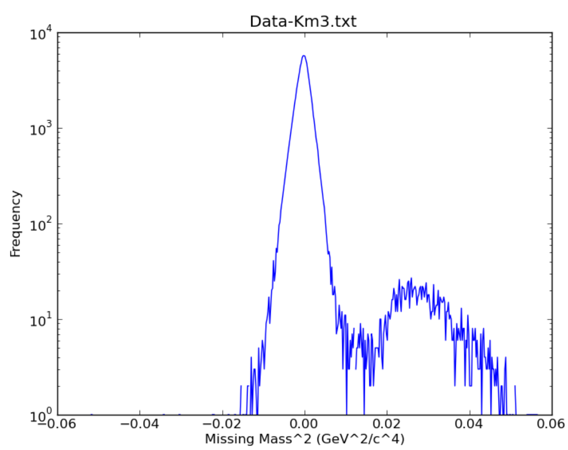
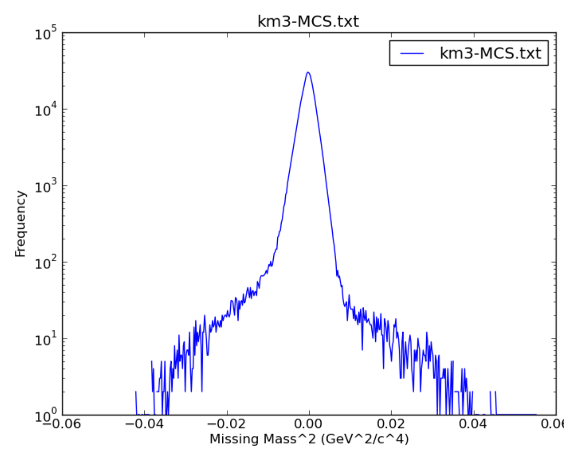


FIG. 12: Shows the transverse momentum distribution of the track and π^0 summed. $Ke3$ events smaller than $0.0294072\ GeV/c$ are rejected to reduce the $K2\pi$ background to minimum. This cut is also applied to $Km3$, being a 3-body decay also, its distribution is approximately the same.



(a) m_ν^2 for the data identified as $K\mu3$.



(b) m_ν^2 for the $K\mu3$ MCS.

FIG. 13: Shows m_ν^2 distributions for data that were identified as $K\mu3$ and the $K\mu3$ MCS. We require $-0.0067 \geq m_\nu^2 \geq 0.0058$ GeV/c^2.

in the data between 0.02 and 0.04 is due to $K3\pi^0$ background whose m_ν^2 distribution is shown in figure 14.

The main sources of background for this channel are from $K2\pi$ events in the tails of the $m_{\pi^0\pi^+}$ distribution shown in figure 10 and from $K3\pi^0$. $K2\pi$ events are eliminated by rejecting events with total measured transverse momentum $\leq 0.0294072\ GeV/c$ as shown in figure 12 and then reduced further by cutting events who, when assuming the track to be a muon, have a invariant mass $m_{\pi^0\mu^+} \geq 0.38\ GeV/c^2$ (figure 15).

$K2\pi$ Background

Figure 16 shows the m_ν^2 distribution for $K2\pi$ decays. The tails of the data distribution are rejected as coming from background non 2-body decays. The main remaining background is due to $K\mu3$ since we cannot reject tracks flagged as muons. To reduce this we apply a more restrictive $m_{\pi^0\pi^+}$ cut requiring $0.488 \geq m_{\pi^0\pi^+} \geq 0.497\ GeV/c^2$ and a cut on events with total measured transverse momenta $\geq 0.0075\ GeV/c$ (distribution shown in figure 12).

Signal losses and acceptances for these background cuts are presented for one of the three measured data samples in appendix . The mean values for the total acceptances were found to be: $Acc_{Ke3} = 0.1297 \pm 0.0007$, $Acc_{K\mu3} = 0.1327 \pm 0.0003$, and $Acc_{K2\pi} = 0.1419 \pm 0.0003$.

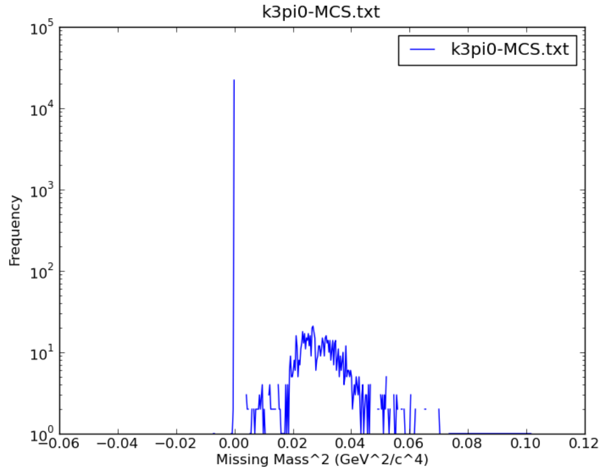
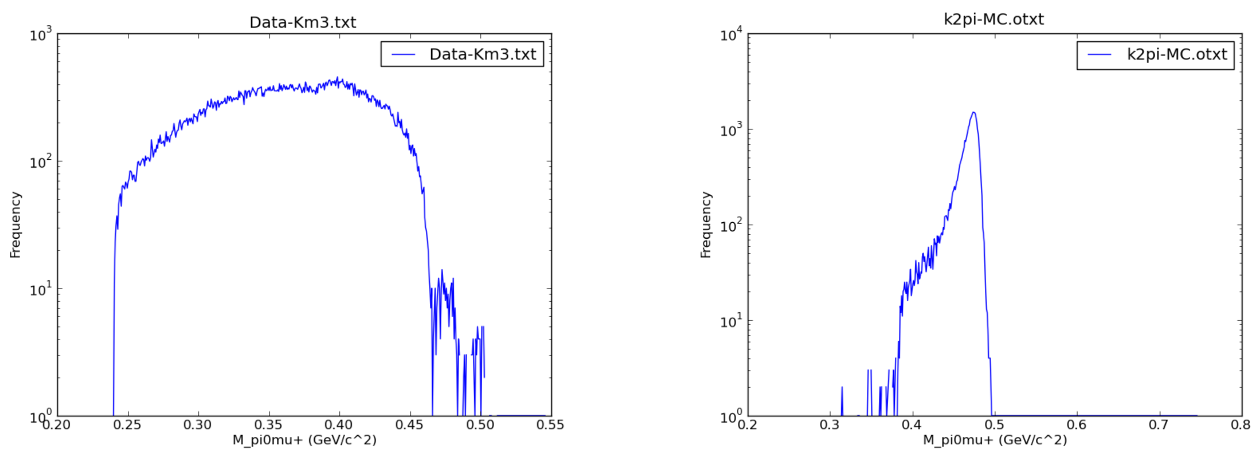


FIG. 14: Shows the m_ν^2 distribution for MCS $K3\pi^0$ events which passed initital selection criteriam, this is a resonable fraction since we allow additional photon events. m_ν^2 has been reconstructed under the assumption that the track is either a muon or positron due to false identification. These events contribute background to $Ke3$ and $K\mu3$ but are largely removed by the m_ν^2 cuts on those channels. The spike at zero is due to the large portion of unidentified events (those whose track passes non of the ID criteria) and whose m_ν^2 is therefore not calculated.



(a) $m_{\pi^0\mu^+}$ distribution for the data identified as $K\mu3$ before cut.

(b) $m_{\pi^0\mu^+}$ distribution the $K2\pi$ MCS.

FIG. 15: Shows $m_{\pi^0\mu^+}$ distributions for $k\mu3$ background cut. Events with $m_{\pi^0\mu^+} \geq 0.38\ GeV/c^2$ are rejected. This reduces the $K2\pi$ background considerably at the cost of $\sim 30\%$ signal loss (See table III)

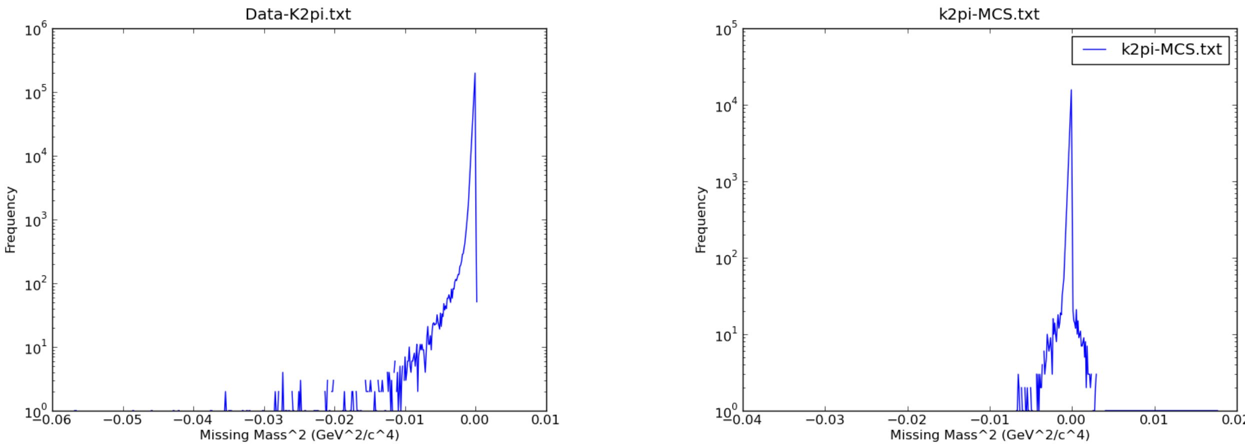
The final estimated background contributions to each channel are displayed in table. II.

RESULTS

Analysing a sample of 8×10^7 events found 2202508 $Ke3$, 1316353 $K\mu3$, and 7783260 $K2\pi$ events. A study of the $L2(Ke2)$ trigger has been performed using tightly selected $Ke3$ events in [2] it found the combined efficiency of the three to be 0.99473 ± 0.00013. This is used for all three channels. Bringing together the values quoted and computing the ratios with (1) we find: $\mathcal{R}_{K_{\mu3}/K_{e3}} = 0.606 \pm 0.004$, $\mathcal{R}_{K_{\mu3}/K_{2\pi}} = 0.1841 \pm 0.006$ and $\mathcal{R}_{K_{e3}/K_{2\pi}} = 0.3037 \pm 0.003$.

These can be compared to the current PDG values: $\mathcal{R}_{K_{\mu3}K_{e3}} = 0.668 \pm 0.008$, $\mathcal{R}_{K_{\mu3}/K_{2\pi}} = 0.159 \pm 0.003$ and $\mathcal{R}_{K_{e3}/K_{2\pi}} = 0.2470 \pm 0.238$.

Using the PDG value of $\Gamma(K2\pi) = 0.2092 \pm 0.0012$ [26], we compute $\Gamma(Ke3) = 0.0635 \pm 0.0005$ and $\Gamma(K\mu3) =$



(a) m_ν^2 for the data identified as $K2\pi$.

(b) m_ν^2 for the $K2\pi$ MCS.

FIG. 16: $K2\pi$ m_ν^2 data - MCS comparison. The difference between the two distributions should be background. We reject events who do not satisfy $-0.0014 \geq m_\nu^2 \geq 0.0002$ in order to retain only the regoin which is in agreement.

Background	
	$Ke3$
$K2\pi$	$(0.00082 \pm 0.00001)\%$
$K\mu3$	$(0.00071 \pm 0.00005)\%$
$K3\pi^0$	$(0.00196 \pm 0.00006)\%$
	$K\mu3$
$K2\pi$	$(0.00082 \pm 0.00001)\%$
$Ke3$	$(0.0 \pm 1 \times 10^{-8})\%$
$K3\pi^0$	$(0.128 \pm 0.002)\%$
	$K2\pi$
$Ke3$	$(0.0361 \pm 0.001)\%$
$K\mu3$	$(0.0 \pm 1 \times 10^{-8})\%$
$K3\pi^0$	$(0.0 \pm 1 \times 10^{-8})\%$

TABLE II: Shows the main background/signal ratios for each of the decay channels. Measured using three 1×10^7 event MCS samples for each source.

0.0308 ± 0.007. Which are compared to the PDG values of $\Gamma(Ke3) = 0.0498 \pm 0.0007$ and $\Gamma(K\mu3) = 0.0332 \pm 0.006$.

DISCUSSION

Although the these values are not in excellent agreement it is seen that using even this small fraction ($\sim 1/25$) of the data sample leads to statistical errors competitive with the current world average. As these have been the largest source of error in most other studies, including that by the NA48/2 collaboration [9], its clear this dataset will greatly reduce the uncertainties in all measured values once a full study is completed.

Before this work can be considered complete an analysis of systematic errors should be performed by looking in detail at the agreement of data with the MCS, and at the effect of varying parameters of the MCS such as the form factor models used as in [9]. We expect this may well uncover errors in the treatment presented here which will move the results closer to the PDG values. We expect that the effects of data-MCS disagreement are probably responsible for most of the differences between our measured values and the PDG averages. It is likely that these can be corrected for once fully understood. An investigation of the affects of radiative decays should also be considered. We suggest looking at MCS samples created with the KLOE generator as this has been shown to have greater agreement with data in the positive tail of the $Ke3$ m_ν^2 distribution, an area which is sensitive to radiative effects.

A dedicated study of trigger efficiencies should be also conducted for each decay channel independently and the effects of the $L3$ filters should be understood. There are some trigger control samples available [**?**] which should be compared to the rest of the data.

An improved measurement of the particle I.D. efficiency should be conducted, taking in to account the dependence on the track momentum, and using data rather than MCS as in [27].

A correction related to differences in beam kaon momentum spectrums in MCS and data should be applied to the MCS weightings. [28] [22] Also investigation into possible improvements to using the average beam momentum for the kaon momentum in kinematical calculation (see [4]).

Once this has all been completed an extraction of the form factors from the data should be performed (see [7]) before an extraction of V_{us} can be done and theoretical implications can be drawn.

CONCLUSION

The measured the ratios of the decay rates have been found to be

$$\mathcal{R}_{K_{\mu 3}/K_{e3}} = 0.606 \pm 0.004, \mathcal{R}_{K_{\mu 3}/K_{2\pi}} = 0.1841 \pm 0.006, \mathcal{R}_{K_{e3}/K_{2\pi}} = 0.3037 \pm 0.003. \tag{3}$$

Using the PDG value for the $K2\pi$ decay rate we determine $\Gamma(Ke3) = 0.0635 \pm 0.0005$, $\Gamma(K\mu 3) = 0.0308 \pm 0.007$.

These values vary by about vary by around 10% from current accepted values, we expect that with further work to understand the sources systematic error and the effect of the high level trigger this discrepancy will decrease. We also showed that, if systematic sources can be controlled, the uncertainty in this measurement will be greatly improved from the PDG value due to the vast statistics available.

* bc1508@my.bristol.ac.uk

[1] O. Reardon-Smith, *$Ke3$ branching ratio measurement and the NA48(2007) dataset*, Bristol Particle Physics Internship, (2013).

[2] S. Gallorini, R. Piandani and A. Romano, *Analysis of the $K^+ \to e^+\nu\gamma$ decay*, Note NA62-12-04, (2012).

[3] `http://goudzovs.web.cern.ch/goudzovs/ke2.html`. Accessed May 1, 2014.

[4] S. Shkarovskiy and D. Madigozhin, *Kl3 analysis reboot*, NA62 Collaboration meeting, Liverpool (2013). (Available on Indico.)

[5] M.Vormstein, *$K\mu 3$ in Period 5 (2007)*, NA62 Analysis Meeting (2013).

[6] B. Crabbe, *Measuring Kl3 Branching Ratios Using The NA62 (2007) Dataset*, Project Literature review (2013)

[7] M. Antonelli et al: FlaviaNet Working Group on Kaon Decays, *An evaluation of $\mid V_{us} \mid$ and precise tests of the Standard Model from world data on leptonic and semileptonic kaon decays)*, (2010). [arXiv:1005.2323]

[8] V. Fanti et al: NA48 Collaboration, *The beam and detector for the NA48 neutral kaon CP violation experiment at CERN*, Nuclear Instruments and Methods in Physics Research, A 574, (2007). 433-471.

[9] J. R. Batley, et al: NA48/2 Collaboration, *Measurements of Charged Kaon Semileptonic Decay Branching Fractions $K^\pm \to e^\pm + \pi^0 + \nu_e$ and $K^-\pm \to \mu^\pm + \pi^0 + \nu_\mu$ and Their Ratio*, Eur. Phys. J. C50 (2007). [arXiv:hep-ex/0702015v2].

[10] *GEANT Description and Simulation Tool*, CERN Program Library Long Writeup W5013 (1994).

[11] P. Golonka and Z. Was, *PHOTOS Monte Carlo: a precision tool for QED corrections in Z and W decays* Eur. Phys. J. C 45 (2006) 97 [arXiv:hep-ph/0506026]

[12] A. Romano, *Leptonic Decays and Kaon Identification at The NA62 Experiment at CERN*, PhD. Thesis, University of Birmingham, (2010). 3.1.4.

[13] J. B. Cheze, *NA48 Drift Chamber Reconstruction Users Guide*, NA48 Documentation (1999).

[14] T. J. Gershon, *Track based reconstruction for muons*, NA48 Note 98-25 (1998).

[15] G. Unal, *Reconstruction program for the LKr*, NA48 Note 98-1 (1998).

[16] E. M. Marinova, *Investigation of charged kaon semileptonic decays*, PhD. Thesis, University of Sofia, (2005). 4.55.

[17] E. M. Marinova, *Investigation of charged kaon semileptonic decays*, PhD. Thesis, University of Sofia, (2005).4.41.

[18] A. Romano, *Leptonic Decays and Kaon Identification at The NA62 Experiment at CERN*, PhD. Thesis, University of Birmingham, (2010).3.1.5.

[19] E. M. Marinova, *Investigation of charged kaon semileptonic decays*, PhD. Thesis, University of Sofia, (2005). 4.4.2.

[20] B. Bloch-Devaux, *Alpha and Beta corrections for NA48/2 Simulated Events*, NA48 Note 05-05, November 24, (2005).

[21] J. R. Batley, et al: NA48/2 Collaboration, *Measurements of Charged Kaon Semileptonic Decay Branching Fractions $K^\pm \to e^\pm + \pi^0 + \nu_e$ and $K^-\pm \to \mu^\pm + \pi^0 + \nu_\mu$ and Their Ratio*, Eur. Phys. J. C50 (2007). [arXiv:hep-ex/0702015v2].

[22] urlhttp://goudzovs.web.cern.ch/goudzovs/ke2/selection.html. Accesed May 1, 2014.

[23] E. M. Marinova, *Investigation of charged kaon semileptonic decays*, PhD. Thesis, University of Sofia, (2005). 5.4.1.

[24] A. Winhart, *P5 and P7 discussion; proposal for LKr cell-by-cell "recalibration"*, (2008).

[25] COmPACT muon reconstruction routine 0902: /afs/cern.ch/user/g/goudzovs/offline/compact/compact-7.3/compact/rlib/anasrc/murec0902.c

[26] J. Beringer et al: Particle Data Group, Phys. Rev. D86, 010001 (2012).

[27] E. Goudzovski et al: The NA62 collaboration, *Precision Measurement of the Ratio of the Charged Kaon Leptonic Decay Rates*, CERN-PH-EP-2012-367, (2012).

[28] A. Romano, *Leptonic Decays and Kaon Identification at The NA62 Experiment at CERN*, PhD. Thesis, University of Birmingham, (2010).3.1.5.5.

Appendix 1: Acceptance and Signal Loss for Cuts

Sample Type:	$Ke3$		$K\mu3$		$K2\pi$		$K3\pi^0$	
	Signal Loss	Acceptance	Signal Loss	Acceptance	Signal Loss	Acceptance	Signal Loss	Acceptance
Initial Selection Cuts								
Bad Burst Cut	N/A	100.00%	N/A	100.00%	N/A	100.00%	N/A	100.00%
L2 Trigger Cut	N/A		N/A		N/A		N/A	
1TRK 2 Cluster Cut	67.89%	32.11%	62.52%	37.48%	55.51%	44.49%	41.77%	58.23%
Cluster Quality Cuts	29.77%	22.55%	28.78%	26.69%	27.46%	32.27%	7.60%	53.80%
Track Quality Cuts	22.13%	17.56%	13.99%	22.96%	13.47%	27.93%	28.06%	38.70%
Energy Sharing Cuts	7.68%	16.21%	6.15%	21.55%	14.14%	23.98%	21.32%	30.45%
Timing Cuts	N/A		N/A		N/A		N/A	
No Pi0 Cut	0.00%	16.21%	0.00%	21.55%	0.00%	23.98%	0.00%	30.45%
Charged Z Cuts	8.25%	14.87%	6.05%	20.24%	6.67%	22.38%	7.40%	28.20%
Multiple Track Cut	0.28%	14.83%	0.17%	20.21%	0.24%	22.32%	1.29%	27.84%
Z Neutral Cut	3.07%	14.38%	2.48%	19.71%	2.28%	21.82%	13.64%	24.04%
Beam-Decay Vertex Cut	4.29%	13.76%	4.29%	19.71%	4.14%	20.91%	22.37%	18.66%
Background Cuts								
Missing Mass^2 Cut	2.58%	13.40%	0.30%	19.65%	0.15%	20.88%		
Event Trans Momentum Cut	3.28%	12.97%	4.54%	18.76%	22.17%	16.25%		
m_Pi0Muon Cut	N/A		43.06%	13.25%	N/A			
m_pi0pi+ Tight Cut	N/A		N/A		12.58%	14.21%		
Final Acceptance:		12.97%		13.25%		14.21%		

TABLE III: Show the ammount of signal lost on each cut for each of our measured channel and the resulting acceptances. Measured with a sample of 1E7 events. This is one of three such samples used in the background and acceptance calculations.

Appendix 3: MCS Samples

$Ke3$: /afs/cern.ch/user/g/goudzovs/www/ke2/lists/mc.p5.ke3.radcor.list (PHOTOS)
$K\mu3$: /afs/cern.ch/user/g/goudzovs/www/ke2/lists/mc.p5.km3.radcor.list
$K2\pi$: /afs/cern.ch/user/g/goudzovs/www/ke2/lists/mc.p5.k2pig.list

Appendix 2: Code

This is the routine which is called for every real data event. The rest of the program is available at:`www.github.com/bcrabbe`.

```
#include "user.h"
/*****************************************************************/
/* COmPAC user routine: user_superCmpEvent(superCmpEvent *sevt)  */
/*                                                               */
/*  User routine called everytime an event '*sevt' is            */
/*  loaded. A return value of greater than zero denotes          */
/*  an error condition has occured.                              */
/*                                     BH 13/2/98    RWM 20/6/97 */
/*****************************************************************/


int user_superCmpEvent(superBurst *sbur,superCmpEvent *sevt)
{
    /* WARNING: do not alter things before this line */
    /*---------- Add user C code here ----------*/


    ++numberOfEventsRead;
    int cutWhichKilledEvent = SURVIVED;
#if ENABLE_EXCLUDE_BAD_BURSTS
    if( sbur->BadB.Dch!=0 || sbur->BadB.Phys!=0  )
    {
       // printf("47\n");
        //cutWhichKilledEvent = 47;
        //fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++badBurstCut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif
    /**Trigger Cut ***********************/
    //this makes sure the data has the "minimum bias" flag set
    //it only rejects only the small fraction of the data which got through on the auto pass trigger
#if ENABLE_MIN_BIAS_CUT
    if((sevt->trigWord & 0x0400)!= 0x0400)// !((sevt->trigWord >> 11) & 1))//
    {
        cutWhichKilledEvent = 93;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++minBiasCut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif

    int numUntrackedClusters = 0;

    //we use this to store the indeces of the clusters that don't have a track
    int * tracklessClusters = NULL;

    user_lkrcalcor_SC (sbur,sevt,1); //lkr nonlinearity correction to all cluster energies

    /*** extract untracked clusters **************************/
    for(int i = 0; (i < sevt->Ncluster);++i)
    {
        if(  sevt->cluster[i].iTrack == -1 && sevt->cluster[i].energy>3.0 )
            // iTrack = -1 if there is no associated track... greater than 3 GeV en to be a photon
        {
            ++numUntrackedClusters;
            tracklessClusters = (int *)realloc(tracklessClusters,
                                        numUntrackedClusters * sizeof(int));
            tracklessClusters[numUntrackedClusters - 1] = i;
        }
```

```
    }
#if ENABLE_DCH_LKR_VETO
    //DCH & Lkr veto:
    if( (numUntrackedClusters < 2) || ( sevt->Ntrack == 0) )
     {
      //   printf("72\n");
        cutWhichKilledEvent = 73;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++VetoCut;

#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif




    /** LKr Quality cuts on trackless clusters
    ************************************************************/
    //clusters must be >2cm from dead cell, must have acceptable cluster status, and be in lkr acceptance

    int numGoodTracklessClusters=numUntrackedClusters;
    for(int i = 0;i<numUntrackedClusters;++i)
    {
        //the clusters positions at lkr face, corrected for projectivity:
        float clusterIPenetrationDepth = 20.8 + 4.3*logf(sevt->cluster[tracklessClusters[i]].energy);
        float lkrPlaneX = (sevt->cluster[tracklessClusters[i]].x + 0.136 + 0.00087*sevt->cluster[tracklessClusters[i]].y) *
                        (1+clusterIPenetrationDepth/10998);
        float lkrPlaneY = (sevt->cluster[tracklessClusters[i]].y + 0.300 - 0.00087*sevt->cluster[tracklessClusters[i]].x) *
                        (1+clusterIPenetrationDepth/10998);
        //printf("non corrected: \%f, \%f  corrected \%f, \%f \n",sevt->cluster[tracklessClusters[i]].x, sevt->cluster[tracklessClusters[i]].y,
          //          lkrPlaneX,lkrPlaneY);
        if( (sevt->cluster[tracklessClusters[i]].status > 4) || (sevt->cluster[tracklessClusters[i]].dDeadCell < 2) ||
                LKr_acc(sbur->nrun, lkrPlaneX, lkrPlaneY, 8)!=0  || sevt->cluster[tracklessClusters[i]].energy<5 )
        {
            tracklessClusters[i]=-1;//if a cluster is bad- remove it by replacing its index with -1
            --numGoodTracklessClusters;
        }
    }
#if ENABLE_BASIC_QUALITY_CUTS
    if(numGoodTracklessClusters<2 && (cutWhichKilledEvent == SURVIVED))
    {
        cutWhichKilledEvent = 130;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++clusterQualCut;

#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif


    /** Track Quality Cut **************************************************/
    // track must be in acceptance of DCHs
    //track must have less than 6\% error on p measurement
    //track must be +ve - this only for p5 data which is +Kaons only.
    //must be within expected momentum range
    //must be within MUV acceptance
    //have a closest distance  of approach to run average beam position of less than 3.5 cm
    int numGoodTracks=sevt->Ntrack;//the number of potentially good tracks left
    int tracks[numGoodTracks], trackedClusters[numGoodTracks];//to store the index of good tracks, and the
```

```
     index of the associated clusters
    float tracksCDA[numGoodTracks],tracksTime[numGoodTracks], tracksChargedVertex[numGoodTracks][3];//to
    store the closest dist approach/vertex of each track
    float dzLkrDCH = Geom->Lkr.z - Geom->DCH.z;
    float dzMuv1DCH = Geom->Muv1.z - Geom->DCH.z;
    float dzMuv2DCH = Geom->Muv2.z - Geom->DCH.z;
    float dzMuv3DCH = Geom->Muv3.z - Geom->DCH.z;

    float beamPoint[3],beamVel[3];
    beamPoint[0]= abcog_params.pkxoffp;
    beamPoint[1] = abcog_params.pkyoffp;
    beamPoint[2] = 0.0;
    beamVel[0] = abcog_params.pkdxdzp;
    beamVel[1] = abcog_params.pkdydzp;
    beamVel[2] = 1.0;
    for(int i=0; i<sevt->Ntrack; i++)//check all tracks
    {
        float trackRadiusDCHb = sqrt(pow(sevt->track[i].bx,2)+pow(sevt->track[i].by,2));
        float trackRadiusDCH = sqrt(pow(sevt->track[i].x,2)+pow(sevt->track[i].y,2));
        float lkrPlaneX = sevt->track[i].x + dzLkrDCH*sevt->track[i].dxdz;
        float lkrPlaneY = sevt->track[i].y + dzLkrDCH*sevt->track[i].dydz;
        int trackCharge = sevt->track[i].q;
        float trackMomentum = sevt->track[i].p;
        float abCorrectedTrackMom = p_corr_ab(trackMomentum,trackCharge);//see http://goudzovs.web.cern.ch
    /goudzovs/ke2/selection.html

        float muv1x = sevt->track[i].x + dzMuv1DCH*sevt->track[i].dxdz;
        float muv1y = sevt->track[i].y + dzMuv1DCH*sevt->track[i].dydz;
        float muv2x = sevt->track[i].x + dzMuv2DCH*sevt->track[i].dxdz;
        float muv2y = sevt->track[i].y + dzMuv2DCH*sevt->track[i].dydz;
        float muv3x = sevt->track[i].x + dzMuv3DCH*sevt->track[i].dxdz;
        float muv3y = sevt->track[i].y + dzMuv3DCH*sevt->track[i].dydz;

        float chargedPartPoint[3],chargedPartVel[3];
        //for the charged particle here we work with the before magnetic field data so we can get
        //the vertex location
        chargedPartPoint[0] = sevt->track[i].bx;//x location of track in pre magnet DCH (DCHb)
        chargedPartPoint[1] = sevt->track[i].by;//y "
        chargedPartPoint[2] = Geom->DCH.bz;//z location of DCHb

        chargedPartVel[0] = sevt->track[i].bdxdz;//track moves in the dirc. (bdxdz,bdydz,1)
        chargedPartVel[1] = sevt->track[i].bdydz;
        chargedPartVel[2] = 1.0;

        float cda;//closest distance approach for each track
        float cdaVertex[3];
        //function from src/user.c:

        closap_(chargedPartPoint,beamPoint,chargedPartVel,beamVel,&cda,cdaVertex);

      //  fprintf(FP2,"\%f\n",cda);
       // fprintf(FP1,"\%f\n",cdaVertex[2]);

        tracksCDA[i] = cda;
        tracksChargedVertex[i][0]=cdaVertex[0];
        tracksChargedVertex[i][1]=cdaVertex[1];
        tracksChargedVertex[i][2]=cdaVertex[2];
        //fprintf(FP2,"\%f\n",cdaVertex[2]);
        //get track time:
        if (sevt->track[i].hodstatus==2)
        {
                tracksTime[i] = sevt->track[i].hodTime;
        }
        else if (sevt->track[i].quality>0.9)
        {
                tracksTime[i]=sevt->track[i].time;
        }
        else if(sevt->track[i].hodstatus==1)
        {
                tracksTime[i] = sevt->track[i].hodTime;
```

```
        }
        else
        {
                tracksTime[i]=sevt->track[i].time;
        }
        //Quality cuts on the track:
        if(  (sevt->track[i].quality < 0.7) || LKr_acc(sbur->nrun, lkrPlaneX, lkrPlaneY, 8)!=0 || (trackRadiusDCH<14) || trackRadiusDCH>115 ||
            (trackRadiusDCHb<12) || trackRadiusDCHb>115 || ((sevt->track[i].perr/abCorrectedTrackMom)>0.06) || trackCharge!=1 || abCorrectedTrackMom<10 ||
                abCorrectedTrackMom>75 || muvAccept(muv1x,muv1y)!=1 || muvAccept(muv2x,muv2y)!=1 /*||muvAccept(muv3x,muv3y)!=1*/ || cda>5 ||
                (cdaVertex[2] < -2000) || (cdaVertex[2] > 9000) )
        {//we get rid of tracks that do not pass all of these
            --numGoodTracks;
            tracks[i]=-1;
            //printf("126\n");
        }
        else
        {//Cluster associated with track:
            int cluster = sevt->track[i].iClus;//this gives the index of the cluster or -1 if there is none

            //Cluster quality cuts:
            if( cluster>-1 && ( (sevt->cluster[cluster].status > 4) ||
                    (sevt->cluster[cluster].dDeadCell < 2) )  )
            {
                  --numGoodTracks;
                    tracks[i]=-1;
            }
            else
            {
                tracks[i]=i;//stores index of track
                trackedClusters[i]=sevt->track[i].iClus;//stores index of associated cluster
            }
        }
    }
#if ENABLE_BASIC_QUALITY_CUTS
    if( numGoodTracks<1 && (cutWhichKilledEvent == SURVIVED) )
    {
        cutWhichKilledEvent = 147;
        fprintf(cutWhichKilledEventFP,"%d\n",cutWhichKilledEvent);
        ++trackQualCut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif
 //   printf("tracks: %d tracks left: %d\n",sevt->Ntrack,numGoodTracks);


/*** Track and its Cluster Time Cut ******************************************/
//the timing of the track must be within 4ns of the associated lkr hit
#if ENABLE_TIMING_CUTS
    for(int i=0;i<sevt->Ntrack;++i)//for all tracks
    {
        if(tracks[i]>-1 && trackedClusters[i]>-1)//that haven't been discarded and have an associated cluster
        {//this wont work in MC (theres no timings)
             float dtTrackCluster =  tracksTime[i] - sevt->cluster[ trackedClusters[i] ].time;
             if (dtTrackCluster>4)
             {
                 tracks[i]=-1;
                 trackedClusters[i]=-1;
                 --numGoodTracks;
                 //printf("167\n");
             }
        }
    }

```

```
    if( numGoodTracks<1 && (cutWhichKilledEvent == SURVIVED) )
    {
        cutWhichKilledEvent = 278;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++trackQualCut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif




#if ENABLE_ENERGY_SHARING_CUTS
/***** Distance between in time gammas cut *************************/
//gammas within 5ns must be more than 22cm apart to avoid energy sharing
    for(int i=0; i<numUntrackedClusters; i++)//for each gamma...
    {
        if(tracklessClusters[i] > -1)//that is good...
        {//using cluster projectivity corrections to get both positions at front face
            float clusterIPenetrationDepth = 20.8 + 4.3*logf(sevt->cluster[tracklessClusters[i]].energy);
            float clusterIx = (sevt->cluster[tracklessClusters[i]].x + 0.136 + 0.00087*sevt->cluster[
    tracklessClusters[i]].y) *
                    (1+clusterIPenetrationDepth/10998);
            float clusterIy = (sevt->cluster[tracklessClusters[i]].y + 0.300 - 0.00087*sevt->cluster[
    tracklessClusters[i]].x) *
                    (1+clusterIPenetrationDepth/10998);

            for(int s=0; s<numUntrackedClusters; s++)//comparing it and all the other gammas..
            {//that are within 5ns:
                float dtgigs = fabs(sevt->cluster[ tracklessClusters[i] ].time - sevt->cluster[
    tracklessClusters[s] ].time);
                if((s!=i) && (dtgigs<5))//..and are also good... and not itself..
                {
                    float clusterSPenetrationDepth = 20.8 + 4.3*logf(sevt->cluster[tracklessClusters[s]].
    energy);
                    float clusterSx = (sevt->cluster[tracklessClusters[s]].x + 0.136 +
                                    0.00087*sevt->cluster[tracklessClusters[s]].y) * (1+
    clusterSPenetrationDepth/10998);
                    float clusterSy = (sevt->cluster[tracklessClusters[s]].y + 0.300 -
                                    0.00087*sevt->cluster[tracklessClusters[s]].x) * (1+
    clusterSPenetrationDepth/10998);

                    float distanceBetweenGammas_X = fabs(clusterSx - clusterIx);
                    float distanceBetweenGammas_Y = fabs(clusterSy - clusterIy);
                    float clusterClusterDist = sqrt(pow(distanceBetweenGammas_X,2) + pow(
    distanceBetweenGammas_Y,2));
                    if(clusterClusterDist<22)//if cluster i is within 22cm of another in time cluster...
                    {//then its a bad cluster
                        tracklessClusters[i]=-1;//if a cluster is bad- remove it by replacing its index
    with -1
                        --numGoodTracklessClusters;
                    }
                }
            }
        }
    }

/*** Cluster distance from all in time tracks cut ******************************/
//clusters must be > 22 cm away from any tracks that are within 10ns
    float minClusterTrackDist[numUntrackedClusters];
    for(int i=0; i < numUntrackedClusters; i++)
    {
        minClusterTrackDist[i]=1e308;
    }
    for(int n=0; n < sevt->Ntrack; ++n)//for all tracks...
    {
        for(int i=0; i < numUntrackedClusters; i++)//for all gammas
        {
```

```
            if((fabs( sevt->cluster[ tracklessClusters[i] ].time - tracksTime[n] ) < 10) )//track and cluster in time
            {
                float clusterIPenetrationDepth = 20.8 + 4.3*logf(sevt->cluster[tracklessClusters[i]].energy);
                float clusterIx = (sevt->cluster[tracklessClusters[i]].x + 0.136 + 0.00087*sevt->cluster[tracklessClusters[i]].y) *
                    (1+clusterIPenetrationDepth/10998);
                float clusterIy = (sevt->cluster[tracklessClusters[i]].y + 0.300 - 0.00087*sevt->cluster[tracklessClusters[i]].x) *
                    (1+clusterIPenetrationDepth/10998);

                float dzDchClusterZ = Geom->Lkr.z + clusterIPenetrationDepth - Geom->DCH.z;
                float TrackLkrPlaneX = sevt->track[n].x + dzDchClusterZ*sevt->track[n].dxdz;
                float TrackLkrPlaneY = sevt->track[n].y + dzDchClusterZ*sevt->track[n].dydz;

                float distanceFromTrack_X = fabs(clusterIx - TrackLkrPlaneX);
                float distanceFromTrack_Y = fabs(clusterIy - TrackLkrPlaneY);
                float clusterTrackDist = sqrt(pow(distanceFromTrack_X,2) + pow(distanceFromTrack_Y,2));
                if(clusterTrackDist<minClusterTrackDist[i])
                {
                    minClusterTrackDist[i]=clusterTrackDist;
                }
                if( (tracklessClusters[i] > -1) && clusterTrackDist<22)
                {
                    tracklessClusters[i]=-1;//if a cluster is bad- remove it by replacing its index with -1
                    --numGoodTracklessClusters;
                }
                if( tracks[n]>-1  && clusterTrackDist<22 )
                {
                    tracks[n]=-1;
                    --numGoodTracks;
                   // printf("266\n");
                }
            }
        }
    }

    if( ( (numGoodTracklessClusters<2) || (numGoodTracks<1) ) && (cutWhichKilledEvent == SURVIVED) )
    {
        cutWhichKilledEvent = 367;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++enSharingCut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif

/**Time difference between the tracks and untracked clusters ******************************************
    */
#if ENABLE_TIMING_CUTS
// the tracks be within 12 ns of atleast 2 photons
    for(int n=0; n<sevt->Ntrack; ++n)
    {
        if(tracks[n]>-1)
        {
            int numInTimePhotons=0;
            for(int i=0; i < numUntrackedClusters; i++)
            {
                if(tracklessClusters[i]>-1)
                {
                    float iClusterTime = sevt->cluster[ tracklessClusters[i] ].time;
                    if(  fabs(iClusterTime - tracksTime[n]) < 12 )//time cut 12ns
                    {
                        ++numInTimePhotons;
                    }
                }
            }
```

```
            if(numInTimePhotons<2)
            {
                tracks[n]=-1;
                --numGoodTracks;
               // printf("312\n");
            }
        }

    }
    //we require that the untracked cluster be within 12ns of at least one good track.
    for(int i=0; i < numUntrackedClusters; i++)
    {
        if( tracklessClusters[i] > -1)
        {
            int numInTimeTracks=0;
            float iClusterTime = sevt->cluster[ tracklessClusters[i] ].time;
            for(int n=0; n<sevt->Ntrack; ++n)
            {
                if(tracks[n] > -1)
                {
                    if(  (fabs(iClusterTime - tracksTime[n]) < 12) )//time cut 12ns
                    {
                        ++numInTimeTracks;
                    }
                }


            }
            if(numInTimeTracks<1)
            {
                tracklessClusters[i]=-1;
                --numGoodTracklessClusters;
            }
        }

    }

    if( ( (numGoodTracklessClusters<2) || (numGoodTracks<1) ) && (cutWhichKilledEvent == SURVIVED) )
    {
        cutWhichKilledEvent = 432;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++timingCut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif

 /****** time difference between the untrackedClusters *******************************/
 /*** selects pi0 gamma candidates ***/

//finds pairs of untracked clusters that arrive at lkr within 5ns of eachother, these are then candidate pi0 photons.
    float dtgigs;//time difference between gamma i and gamma s.
    int pi0GammaCandidatePairs[numGoodTracklessClusters*numGoodTracklessClusters][2];
    int numPi0GammaCandidatePairs=0;
    //this will probably be the best pi0 combination, however we will not assume so....
     //that is decided by z vertex location... if non are less than 2ns then we get rid of event
    for(int i=0; i<numUntrackedClusters; i++)//for each gamma...
    {
        if(tracklessClusters[i] > -1)//that is good...
        {
            for(int s=0; s<numUntrackedClusters; s++)
            //calcuate the time difference between it and all the other gammas..
            {
                if((s!=i) && (tracklessClusters[s] > -1) )//..that are also good... and not itself..
                {
                    dtgigs = fabs(sevt->cluster[ tracklessClusters[i] ].time -
                                                 sevt->cluster[ tracklessClusters[s] ].time);
```

```
                    if(dtgigs<5)//if within 2ns of each other then we store them as pi0 gamma pair candidate
                    {

                        if(i<s)//makes sure we dont store same combination twice.
                        {

                            ++numPi0GammaCandidatePairs;
                            pi0GammaCandidatePairs[ numPi0GammaCandidatePairs-1 ][0] = tracklessClusters[i];
                            pi0GammaCandidatePairs[ numPi0GammaCandidatePairs-1 ][1] = tracklessClusters[s];

                            //now pi0Cands contains the cluster indices of the possible pi0 gamma pairs in each row
                        }
                    }

                }
            }
        }
    }
#if ENABLE_TIMING_CUTS

        if( (numPi0GammaCandidatePairs < 1) && (cutWhichKilledEvent == SURVIVED) )
            //if less than two - we dont have a pi0
        {
            cutWhichKilledEvent = 484;
            fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
            ++noPi0Cut;
#if BREAK_ON_FAILED_CUT
            return -1;
#endif
        }
#endif

/**** Track in time with Pi0 cut*********************************************/
// get rid of tracks that are not in time with one of the pi0 candidates.
  for(int n=0; n<sevt->Ntrack; ++n)
    {
        if(tracks[n]>-1)
        {
            int inTimeWithPi0=0;
            for(int i=0; i<numPi0GammaCandidatePairs; ++i)
            {
                float pi0time = (sevt->cluster[ pi0GammaCandidatePairs[i][0] ].time +
                                 sevt->cluster[ pi0GammaCandidatePairs[i][1] ].time )/2;
                if(   (fabs(pi0time - tracksTime[n] ) < 10) )
                {
                    ++inTimeWithPi0;
                }
            }
            if(inTimeWithPi0==0)
            {
                tracks[n]=-1;
                --numGoodTracks;
            }
        }

    }
    #if ENABLE_TIMING_CUTS
    if(numGoodTracks<1 && cutWhichKilledEvent == SURVIVED)
    {
        cutWhichKilledEvent = 522;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++timingCut;

#if BREAK_ON_FAILED_CUT
        return -1;
#endif
```

```
    }
#endif


/*** Find Blue Field Corrected charged Decay Vertex
    ********************************************************/
//uses the algorith explained on http://goudzovs.web.cern.ch/goudzovs/ke2/selection.html
//cut on bf corrected zCharged and CDA
   // float tracksBfCorrChargedVertex[sevt->Ntrack][3];
    float zCharged;
    for(int i=0;i<sevt->Ntrack;++i)
    {
        if(tracks[i]>-1)
        {

            float chargedPartVel[3],chargedPartPoint[3],bfCorrChargedPartVel[3],bfCorrChargedPartPoint
    [3];;
            chargedPartVel[0] = sevt->track[tracks[i]].bdxdz; //track moves in the dirc. (bdxdz,bdydz,1)
            chargedPartVel[1] = sevt->track[tracks[i]].bdydz;
            chargedPartVel[2] = 1.0;
            chargedPartPoint[0] = sevt->track[tracks[i]].bx; //x location of track in pre magnet DCH (DCHb)
            chargedPartPoint[1] = sevt->track[tracks[i]].by; //y "
            chargedPartPoint[2] = Geom->DCH.bz; //z location of DCHb
            bfCorrChargedPartVel[0]=chargedPartVel[0];
            bfCorrChargedPartVel[1]=chargedPartVel[1];
            bfCorrChargedPartVel[2]=chargedPartVel[2];

            bfCorrChargedPartPoint[0]=chargedPartPoint[0];
            bfCorrChargedPartPoint[1]=chargedPartPoint[1];
            bfCorrChargedPartPoint[2]=chargedPartPoint[2];

            float decayVertex[3];
            decayVertex[0] = tracksChargedVertex[i][0];
            decayVertex[1] = tracksChargedVertex[i][1];
            decayVertex[2] = tracksChargedVertex[i][2];
            int trackCharge = 1;
            float trackMomentum = sevt->track[tracks[i]].p;
            float abCorrectedTrackMom = p_corr_ab(trackMomentum,trackCharge);
            blue_tack_(&trackCharge,&abCorrectedTrackMom,decayVertex,bfCorrChargedPartPoint,
    bfCorrChargedPartVel);
         //  printf("old slopes: \%f ,\%f new slopes: \%f, \%f \n",chargedPartVel[0],chargedPartVel[1],
    bfCorrChargedPartVel[0],bfCorrChargedPartVel[1]);
            //printf("old points: \%f ,\%f new points: \%f, \%f \n",chargedPartPoint[0],chargedPartPoint
    [1],bfCorrChargedPartPoint[0],bfCorrChargedPartPoint[1]);

            //float trackMidPointPosition[3];
            //trackMidPointPosition[2] = (Geom->DCH.bz - tracksChargedVertex[i][2])/2;
            //trackMidPointPosition[0] = chargedPartPoint[0] - trackMidPointPosition[2]*chargedPartVel[0];
            //trackMidPointPosition[1] = chargedPartPoint[1] - trackMidPointPosition[2]*chargedPartVel[1];
            //to get the coordinates of the decay vertex we use the charged tracks coordinates and
    velocity at DCHb (before magnet)
            //then extrapolate it back to Zneutral
            float bfCorrVertex[3], bfCorrCda;
            closap_(bfCorrChargedPartPoint,beamPoint,bfCorrChargedPartVel,beamVel,&bfCorrCda,bfCorrVertex)
    ;
            //printf("cda: \%f ---> \%f\n",tracksCDA[i],bfCorrCda);
           // printf("zChar: \%f ---> \%f\n",tracksChargedVertex[i][2],bfCorrVertex[2]);

           // fprintf(FP1,"\%f\n",bfCorrCda);
          // fprintf(FP2,"\%f\n",bfCorrVertex[2]);
            if( bfCorrCda>3 || bfCorrVertex[2]<-1600 || bfCorrVertex[2]>7000 )
            {
                tracks[i]=-1;
                --numGoodTracks;
                //tracksCDA[i]=bfCorrCda;
               // tracksBfCorrChargedVertex[i][0] = bfCorrVertex[0];
                //tracksBfCorrChargedVertex[i][1] = bfCorrVertex[1];
                //tracksBfCorrChargedVertex[i][2] = bfCorrVertex[2];
            }
            else
```

```
                zCharged=bfCorrVertex[2];
        }
    }

#if ENABLE_Z_COORD_CUT
    if(numGoodTracks<1 && cutWhichKilledEvent == SURVIVED)
    {
        cutWhichKilledEvent = 597;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++chargedZcut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif


/*** Track Veto *********************************************************************/
//at this point the chance of having more than one good track is  small
//if it does happen we can just get rid of the event with little affect on the efficiency
    int iTrack;//=0;
    int iTrackedCluster;//=sevt->track[0].iClus;
   // float decayVertexClosestApproach[3];    //coordinates of decay
    //float iTrackCda;

   if(numGoodTracks!=1 && cutWhichKilledEvent == SURVIVED)
    {
        //printf("too many tracks\n");
        cutWhichKilledEvent = 534;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++multipleTracksCut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
    else
    {
        for(int i=0;i<sevt->Ntrack;++i)
        {
            if(tracks[i]>-1)
            {
                //fprintf(FP2,"\%f\n",tracksChargedVertex[i][2]);
                //fprintf(FP1,"\%f\n",tracksCDA[i]);
                iTrack = tracks[i];
                iTrackedCluster = sevt->track[i].iClus;

            }
        }

    }
  //  float zCharged = decayVertexClosestApproach[2];//Z coord of the vertex according to track
   //  printf("\%f \%f\n",zCharged,iTrackCda);


/*** Here we calculate untracked cluster energies with CPD corrections ***************************/
//might need to make this do all good untracked clusters, depends if we decide to use that info or not.
    float tracklessClustersCorrectedEnergies[sevt->Ncluster];
    for(int i=0; i<numUntrackedClusters; ++i)
    {
        if(tracklessClusters[i]>-1)
        {
            // First find out to which cell is pointing the cluster hit (define CPDindex and CELLindex)
            // take lkr cluster position at front face
            double clusterPenetrationDepth = 20.8 + 4.3*logf(sevt->cluster[ tracklessClusters[i] ].energy)
    ;
            double clusterx = (sevt->cluster[ tracklessClusters[i] ].x + 0.136 + 0.00087*sevt->cluster[
    tracklessClusters[i] ].y) *
                        (1+clusterPenetrationDepth/10998);
            double clustery = (sevt->cluster[ tracklessClusters[i] ].y + 0.300 - 0.00087*sevt->cluster[
    tracklessClusters[i] ].x) *
```

```
                              (1+clusterPenetrationDepth/10998);
            int CELLindex;
            int CPDindex;
            //int * cpd_index=&CPDindex, *cell_index=&CELLindex;
            GetCpdCellIndex(clusterx, clustery, &CPDindex, &CELLindex);
            if( CELLindex==-1 || CPDindex==-1 )
            {
                tracklessClustersCorrectedEnergies[ tracklessClusters[i] ] = sevt->cluster[
    tracklessClusters[i] ].energy;

            }
            // Ke3 E/p correction for each cell
            tracklessClustersCorrectedEnergies[ tracklessClusters[i] ] = sevt->cluster[ tracklessClusters
    [i] ].energy / EopCorr[CPDindex][CELLindex];
           // printf("old: \%f    new: \%f \n", sevt->cluster[ tracklessClusters[i] ].energy,
    tracklessClustersCorrectedEnergies[i]);
        }
    }

    /** Calculates the decay vertex from cluster data **********************************/
    /** selects pi0 pair ***/

    float zDiffMin=1e308;
    int pi0pair=-1;
    float zPi,zNeutral;
    float pi0Photon1Energy,pi0Photon2Energy,pi0LkrEnergy;
    for(int i=0; i<numPi0GammaCandidatePairs; i++)
    {
        float gamma1Energy = tracklessClustersCorrectedEnergies[ pi0GammaCandidatePairs[i][0] ];
        float gamma2Energy = tracklessClustersCorrectedEnergies[ pi0GammaCandidatePairs[i][1] ];

        float gamma1PenetrationDepth = 20.8 + 4.3*logf(gamma1Energy);
        float gamma2PenetrationDepth = 20.8 + 4.3*logf(gamma2Energy);


        float gamma1LkrVertex[3],gamma2LkrVertex[3];

        gamma1LkrVertex[0] = (sevt->cluster[pi0GammaCandidatePairs[i][0]].x + 0.136 +
                              0.00087*sevt->cluster[pi0GammaCandidatePairs[i][0]].y) * (1+
    gamma1PenetrationDepth/10998);

        gamma1LkrVertex[1] = (sevt->cluster[pi0GammaCandidatePairs[i][0]].y + 0.300 -
                              0.00087*sevt->cluster[pi0GammaCandidatePairs[i][0]].x) * (1+
    gamma1PenetrationDepth/10998);

        gamma1LkrVertex[2] = Geom->Lkr.z; // + gamma1PenetrationDepth;

        gamma2LkrVertex[0] = (sevt->cluster[pi0GammaCandidatePairs[i][1]].x + 0.136 +
                              0.00087*sevt->cluster[pi0GammaCandidatePairs[i][1]].y) * (1+
    gamma2PenetrationDepth/10998);

        gamma2LkrVertex[1] = (sevt->cluster[pi0GammaCandidatePairs[i][1]].y + 0.300 -
                              0.00087*sevt->cluster[pi0GammaCandidatePairs[i][1]].x) * (1+
    gamma2PenetrationDepth/10998);

        gamma2LkrVertex[2] = Geom->Lkr.z; // + gamma2PenetrationDepth;

        float dispGamma1Gamma2[3];
        for(int j = 0; j < 3; ++j)
        {
            dispGamma1Gamma2[j] = gamma1LkrVertex[j] - gamma2LkrVertex[j];
        }

        float gamma1Gamma2Distance = f3vmag(dispGamma1Gamma2);
        //fprintf(FP1,"g1g2dist: \%f gEn: \%f \%f  \n",gamma1Gamma2Distance,gamma1Energy,gamma2Energy);
        float dispPiLkr = gamma1Gamma2Distance*sqrt(gamma1Energy * gamma2Energy)/PI0_MASS;
        //disp of pi0 decay to lkr
        zPi = (Geom->Lkr.z) - dispPiLkr; //Z coord of vertex according to pi0
        //fprintf(FP2,"\%f\n",zPi);
        //if( ((zPi > -1600) && (zPi < 9000)))
```

```
        {
            float zDiff = fabs(zCharged - zPi);

            if (zDiff<zDiffMin)
            {
                zDiffMin =zDiff;
                pi0pair = i;
                zNeutral = zPi;
                pi0Photon1Energy = gamma1Energy;
                pi0Photon2Energy = gamma2Energy;
                pi0LkrEnergy = pi0Photon1Energy +pi0Photon2Energy;
            }
        }
    }
   // fprintf(FP2,"\%f \%f \n",minClusterTrackDist[pi0GammaCandidatePairs[pi0pair][0]],zDiffMin);
    //fprintf(FP2,"\%f \%f \n",minClusterTrackDist[pi0GammaCandidatePairs[pi0pair][1]],zDiffMin);

#if ENABLE_Z_COORD_CUT
    if( ((zNeutral< -1600) || (zNeutral > 7000) || (pi0pair==-1) || pi0LkrEnergy<15 )&& (
    cutWhichKilledEvent == SURVIVED) )
    {
        cutWhichKilledEvent = 727;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++zNeutralCut;

#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif
      // fprintf(FP2,"\%f\n",zDiffMin);
       //fprintf(ke3FP,"\%f\n",zCharged);
        //fprintf(FP1,"\%f\n",zNeutral);
/*
#if ENABLE_Z_COORD_CUT
    if( ( (zDiffMin< -8000) || (zDiffMin > 8000) ) && (cutWhichKilledEvent == SURVIVED) )
    {
        cutWhichKilledEvent = 737;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++zDiffCut;

#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif
*/


/*** Decay Vertex calculation using z neutral and bf corrected slopes***/

    float chargedPartVel[3],chargedPartPoint[3],bfCorrChargedPartVel[3],bfCorrChargedPartPoint[3];;
    chargedPartVel[0] = sevt->track[iTrack].bdxdz;//track moves in the dirc. (bdxdz,bdydz,1)
    chargedPartVel[1] = sevt->track[iTrack].bdydz;
    chargedPartVel[2] = 1.0;
    chargedPartPoint[0] = sevt->track[iTrack].bx;//x location of track in pre magnet DCH (DCHb)
    chargedPartPoint[1] = sevt->track[iTrack].by;//y "
    chargedPartPoint[2] = Geom->DCH.bz;//z location of DCHb
    bfCorrChargedPartVel[0]=chargedPartVel[0];
    bfCorrChargedPartVel[1]=chargedPartVel[1];
    bfCorrChargedPartVel[2]=chargedPartVel[2];

    bfCorrChargedPartPoint[0]=chargedPartPoint[0];
    bfCorrChargedPartPoint[1]=chargedPartPoint[1];
    bfCorrChargedPartPoint[2]=chargedPartPoint[2];
    //using zn, first iteration vertex is
    float decayVertex[3];
    decayVertex[0] = chargedPartPoint[0]-chargedPartVel[0]*(chargedPartPoint[2]-zNeutral);
    decayVertex[1] = chargedPartPoint[1]-chargedPartVel[1]*(chargedPartPoint[2]-zNeutral);
    decayVertex[2] = zNeutral;
```

```
    int trackCharge = 1;
    float trackMomentum = sevt->track[iTrack].p;
    float abCorrectedTrackMom = p_corr_ab(trackMomentum, trackCharge);
    //get bf corrections;
    blue_tack_(&trackCharge,&abCorrectedTrackMom, decayVertex, bfCorrChargedPartPoint, bfCorrChargedPartVel);
    //second iteration vertex using bf corrections:
    decayVertex[0] = bfCorrChargedPartPoint[0]-bfCorrChargedPartVel[0]*(bfCorrChargedPartPoint[2]-zNeutral);
    decayVertex[1] = bfCorrChargedPartPoint[1]-bfCorrChargedPartVel[1]*(bfCorrChargedPartPoint[2]-zNeutral);
    decayVertex[2] = zNeutral;

    //now work out the displacement of decay vertex from beam:
    //first the beam position at Zn:
    float beamPositionZn[3];
    beamPositionZn[0] = beamPoint[0] + (zNeutral*beamVel[0]);//beamPoint/vel defined line 156
    beamPositionZn[1] = beamPoint[1] + (zNeutral*beamVel[1]);
    beamPositionZn[2] = zNeutral;//the z coord found from pi0 data

    float dispBeamDecayVertexX = beamPositionZn[0] - decayVertex[0];
    float dispBeamDecayVertexY = beamPositionZn[1] - decayVertex[1];

    float radialDistBeamDecayVertex= sqrt( pow(dispBeamDecayVertexY, 2) + pow(dispBeamDecayVertexX, 2) );

   // fprintf(FP1,"\%f\n",dispBeamDecayVertexX);
    // fprintf(FP2,"\%f \%f\n",dispBeamDecayVertex[0],dispBeamDecayVertex[1]);
    //fprintf(FP2,"\%f \n",radialDistBeamDecayVertex);

    //  fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        //return 0;}

    //fprintf(FP2,"\%f\n",zPi);
    //fprintf(FP1,"\%f \%f\n",zNeutral,radialDistBeamDecayVertex);
    // printf("\%f\n",zDiffMin);
#if ENABLE_Z_COORD_CUT
    //cut on radial decay distance
    if(radialDistBeamDecayVertex > 3 )
    {
        cutWhichKilledEvent = 795;
        fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);
        ++radialDecayVertexCut;
#if BREAK_ON_FAILED_CUT
        return -1;
#endif
    }
#endif
//fprintf(FP1,"\%f \n",radialDistBeamDecayVertex);




/*

    free(tracklessClusters);

 ++nEvents;



        return 0;}


*/
```

```
/*** Kinematics Calulations *************************************************************************************/
/***------------------------------------------***/

    /*** Here we find the 2 pi0 gammas three momentum *********************************************/
    //to work this out we need two points on the photons path
    //we use the points where each photon hits the Lkr:

    float gamma1Energy = pi0Photon1Energy; //to avoid rewriting code  below.
    float gamma2Energy = pi0Photon2Energy;

    float gamma1PenetrationDepth = 20.8 + 4.3*logf(gamma1Energy);
    float gamma2PenetrationDepth = 20.8 + 4.3*logf(gamma2Energy);
    float gamma1LkrVertex[3],gamma2LkrVertex[3];

    gamma1LkrVertex[0] = (sevt->cluster[pi0GammaCandidatePairs[pi0pair][0]].x + 0.136 +
            0.00087*sevt->cluster[pi0GammaCandidatePairs[pi0pair][0]].y) * (1+gamma1PenetrationDepth/10998);

    gamma1LkrVertex[1] = (sevt->cluster[pi0GammaCandidatePairs[pi0pair][0]].y + 0.300 -
            0.00087*sevt->cluster[pi0GammaCandidatePairs[pi0pair][0]].x) * (1+gamma1PenetrationDepth/10998);

    gamma1LkrVertex[2] = Geom->Lkr.z; // + gamma1PenetrationDepth;

    gamma2LkrVertex[0] = (sevt->cluster[pi0GammaCandidatePairs[pi0pair][1]].x + 0.136 +
            0.00087*sevt->cluster[pi0GammaCandidatePairs[pi0pair][1]].y) * (1+gamma2PenetrationDepth/10998);

    gamma2LkrVertex[1] = (sevt->cluster[pi0GammaCandidatePairs[pi0pair][1]].y + 0.300 -
            0.00087*sevt->cluster[pi0GammaCandidatePairs[pi0pair][1]].x) * (1+gamma2PenetrationDepth/10998);
    gamma2LkrVertex[2] = Geom->Lkr.z; // + gamma1PenetrationDepth;



    //and the second is the decay vertex...

    //we then have a vector of the photons paths after the decay
    float dispDecayVertexGamma1Vertex[3],dispDecayVertexGamma2Vertex[3];
    for(int i = 0; i < 3;++i)
    {
        dispDecayVertexGamma1Vertex[i] = gamma1LkrVertex[i] - decayVertex[i];
        dispDecayVertexGamma2Vertex[i] = gamma2LkrVertex[i] - decayVertex[i];
    }

    float distDecayVertexGamma1Vertex = f3vmag(dispDecayVertexGamma1Vertex);
    float distDecayVertexGamma2Vertex = f3vmag(dispDecayVertexGamma2Vertex);

    float gamma1VectMom[3],gamma2VectMom[3];
    for(int i = 0; i < 3; ++i)
    {
        //normalise then multiply by momentum magnitude (same as energy for a photon)
        gamma1VectMom[i] = gamma1Energy*(dispDecayVertexGamma1Vertex[i]/distDecayVertexGamma1Vertex);
        gamma2VectMom[i] = gamma2Energy*(dispDecayVertexGamma2Vertex[i]/distDecayVertexGamma2Vertex);
    }

    /****HERE WE RECONSTRUCT THE PI0 INVARIANT MASS**********************/

    float pi0VectMom[3];
    for(int i = 0; i < 3; ++i)
    {
        pi0VectMom[i] = gamma1VectMom[i] + gamma2VectMom[i];
    }
```

```
    float pi0Energy = gamma1Energy + gamma2Energy;
    float pi0MomMag = f3vmag(pi0VectMom);

    float pi0FourMom[4]={ pi0Energy, pi0VectMom[0], pi0VectMom[1], pi0VectMom[2] };

    float pi0ReconstructedMass2 = f4vdot(pi0FourMom,pi0FourMom);
    float pi0ReconstructedMass = sqrt(pi0ReconstructedMass2);
    //sometimes reconstruction makes the mass imaginary
    //(this is bad so we ignore it and pretend it doesnt happen)
    //In c an imaginary number becomes "-nan" which has the property than -nan != -nan

    if( pi0ReconstructedMass != pi0ReconstructedMass)
    {
        printf("Error: pi0 mass reconstruction, line 593\n");
       // return -1;
    }

    //fprintf(FP1,"\%f\n",pi0ReconstructedMass);



    /*** Here we calculate four momentum of additional (radiative) gammas ******************/
    //array to store the 4 mom of each additional photon

    int numberOfAdditionalPhotons=0;
    float additionalGamma4Mom[numGoodTracklessClusters][4];
    float additionalGammaEnSum = 0;
    if(numGoodTracklessClusters > 2)
    {

        for(int s=0; s<numGoodTracklessClusters; ++s)
        {//raditive gammas are those which did not get eliminated (set to -1) and are not the pi0
            if(tracklessClusters[s]>-1 && ( tracklessClusters[s]!=pi0GammaCandidatePairs[pi0pair][0] ||
                    tracklessClusters[s]!=pi0GammaCandidatePairs[pi0pair][1])    )
            {
                float gammaEnergy = tracklessClustersCorrectedEnergies[tracklessClusters[s]];
                additionalGammaEnSum += gammaEnergy;
                float gammaPenetrationDepth = 20.8 + 4.3*logf(gammaEnergy);

                //now do the same as for the pi0 gammas
                float gammaLkrVertex[3];
                gammaLkrVertex[0] = (sevt->cluster[tracklessClusters[s]].x + 0.136 +
                        0.00087*sevt->cluster[tracklessClusters[s]].y) * (1+gammaPenetrationDepth
    /10998);
                gammaLkrVertex[1] = (sevt->cluster[tracklessClusters[s]].y + 0.300 -
                        0.00087*sevt->cluster[tracklessClusters[s]].x) * (1+gammaPenetrationDepth
    /10998);
                gammaLkrVertex[2] = Geom->Lkr.z;// + gamma1PenetrationDepth;

                float dispDecayVertexGammaVertex[3];
                for(int i = 0; i < 3;++i)
                {
                    dispDecayVertexGammaVertex[i] = gammaLkrVertex[i] - decayVertex[i];
                }
                float distDecayVertexGammaVertex = f3vmag(dispDecayVertexGammaVertex);

                additionalGamma4Mom[numberOfAdditionalPhotons][0]=gammaEnergy;
                for(int i = 1; i < 4; ++i)
                {
                    additionalGamma4Mom[numberOfAdditionalPhotons][i] = gammaEnergy*(
    dispDecayVertexGammaVertex[i-1]/distDecayVertexGammaVertex);
                }
                ++numberOfAdditionalPhotons;
               // printf("radiative\n");
            }
        }
    }

/*    if(numGoodTracklessClusters > 2 && numberOfAdditionalPhotons==0 )
    {
```

```
        printf("————————————————————\n pi0 indices=\%d, \%d\n",pi0GammaCandidatePairs[pi0pair][0], pi0GammaCandidatePairs[pi0pair][1]);
        for(int i=0; i<numUntrackedClusters; ++i)
        {
            printf(" \%d\t",tracklessClusters[i]);
        }
        printf("\n");
    }*/


/***Here we find 4 momentum of kaon(assumed to be beam average) and charged track *************************/
    //first we find momentum of kaon, assumed to be beam average.
    //this is really dodgy because we've already shown that abcog_params lies to us!
    //need to find out how they do it in "Reboot" slides.
    float beamVelocity[3];
    beamVelocity[0] = abcog_params.pkdxdzp;
    beamVelocity[1] = abcog_params.pkdydzp;
    beamVelocity[2] = 1.0;
    float beamVelMag = f3vmag(beamVelocity);

    float beamMomMag =  p_corr_ab(abcog_params.pkp,1);//alpha-Beta correction
    beamMomMag = beamMomMag*(1+abcog_params.beta);//additiional beta correction
    //E = (p^2 + m^2)^1/2
    float beamKaonEnergy = sqrt( pow(beamMomMag,2) + pow(abcog_params.mkp,2) );
    float kaonEnergy = beamKaonEnergy;//sqrt(f3vmag2(beamVectMom) + _POWER2(abcog_params.mkp) );

    //track:
    float chargedPartMomMag = p_corr_ab(sevt->track[iTrack].p,1);
    //using chargedPartVel declared line 592
    float chargedPartVelMag = f3vmag(chargedPartVel);



/*** He we find track energy with CPD Corrections ***************************************************/
    float chargedPartEnergy;
    if ( iTrackedCluster > -1)
    {
        // First find out to which cell is pointing the deflected track (define CPDindex and CELLindex)
        //find track coords at lkr face:
        float dzDchClusterZ = Geom->Lkr.z - Geom->DCH.z;
        double trkatlkr[2];
        trkatlkr[0] = sevt->track[iTrack].x + dzDchClusterZ*sevt->track[iTrack].dxdz;
        trkatlkr[1] = sevt->track[iTrack].y + dzDchClusterZ*sevt->track[iTrack].dydz;

        int CELLindex;
        int CPDindex;

        GetCpdCellIndex(trkatlkr[0] , trkatlkr[1], &CPDindex, &CELLindex);
        if( CELLindex==-1 || CPDindex==-1 )
        {
            printf("GetCpdCellIndex error\n");
            chargedPartEnergy = sevt->cluster[iTrackedCluster].energy;
        }
        // Now that you know the cell hit by the track, correct the energy for the track cluster (is there is one associated to the track)
        else
            chargedPartEnergy = sevt->cluster[iTrackedCluster].energy / EopCorr[CPDindex][CELLindex];   // Ke3 E/p correction for each cell
    }
    else
    {
        chargedPartEnergy = 0;
    }

    float chargedPartEPRatio = chargedPartEnergy / chargedPartMomMag;

    fprintf(FP1,"\%f\n",chargedPartEPRatio);
```

```
    float chargedPartVectMom[3], beamVectMom[3];

    for(int i = 0; i < 3; ++i)
    {
        chargedPartVectMom[i] = chargedPartMomMag*chargedPartVel[i]/chargedPartVelMag;
        beamVectMom[i] = beamMomMag * beamVelocity[i] / beamVelMag;
    }
    float chargedPart4Mom[4] = {chargedPartEnergy, chargedPartVectMom[0], chargedPartVectMom[1],
    chargedPartVectMom[2] };
    float beam4Mom[4] = { beamKaonEnergy, beamVectMom[0], beamVectMom[1], beamVectMom[2]  };


    /****Missing (three) Momentum Calculation *****************************************/

    float detectedMom[3];
    for(int i = 0; i < 3; ++i)
    {
        detectedMom[i] = pi0VectMom[i] + chargedPartVectMom[i];
    }

    float detectedMomIncRadiation[3]={detectedMom[0],detectedMom[1],detectedMom[2]};
    if(numberOfAdditionalPhotons > 0)
    {
        for(int s=0; s<numberOfAdditionalPhotons;++s)
        {
             for(int i = 0; i < 3; ++i)
            {
                detectedMomIncRadiation[i] = detectedMomIncRadiation[i] + additionalGamma4Mom[s][i+1];

            }
        }
    }

    float missingMom[3];
    for( int i = 0; i < 3; ++i)
    {
        missingMom[i] = beamVectMom[i] - detectedMom[i];
    }

    float missingMomIncRad[3];
    for( int i = 0; i < 3; ++i)
    {
        missingMomIncRad[i] = beamVectMom[i] - detectedMomIncRadiation[i];
    }

    float missingMomMag = f3vmag(missingMom);

    /***HERE WE ASSUME THE CHARGED PARTICLE IS A PI+ AND CALCULATE THE KAON MASS*************/
    // see arXiv:hep-ex/0702015v2  4.2
    //see the note in user.h about naming the pi+ Pi
    float pi0EnergyNonLkrMeasure = sqrt( _POWER2(PI0_MASS) + _POWER2(pi0MomMag));
    float pi1Energy = sqrt(_POWER2(PI1_MASS) + _POWER2(chargedPartMomMag));
    float pi0Pi1Mass = sqrt(_POWER2(pi0EnergyNonLkrMeasure + pi1Energy) - f3vmag2(detectedMom));
    float pi0Pi1MassIncRad = sqrt(_POWER2(pi0EnergyNonLkrMeasure + pi1Energy + additionalGammaEnSum) -
    f3vmag2(detectedMomIncRadiation));
    /*** same, assuming electron, and muon*************************************************/
    float eEnergy = sqrt(_POWER2(ELECTRON_MASS) + _POWER2(chargedPartMomMag));
    float pi0ElectronMass =  sqrt(_POWER2(pi0EnergyNonLkrMeasure + eEnergy) - f3vmag2(detectedMom));
    float muEnergy = sqrt(_POWER2(MUON_MASS) + _POWER2(chargedPartMomMag));
    float pi0MuonMass =  sqrt(_POWER2(pi0EnergyNonLkrMeasure + muEnergy) - f3vmag2(detectedMom));

   // fprintf(FP2,"\%f\n",pi0Pi1Mass);


    /***find assoiciated muon if present************************************/

    //This is -1 if there is no muon, apparently with high reliability
    //the muon structure is filled by the murec0902 routine
    // source code: /afs/cern.ch/user/g/goudzovs/offline/compact/compact-7.3/compact/rlib/anasrc/murec0902
    .c
```

```
    //it leaves you to check which planes the was a hit in, encoded in the status variable, we require hits in all 3 or just 1 and 2.
    //it leaves you to test the time resultion, we require within 3.5 ns
    int iMuon = sevt->track[iTrack].iMuon; //index of associated muon
    int muon; //if we
    if(iMuon == -1)
    {
        muon = 0;
    }
    else
    {
        float timeBetweenTrackAndMuon = fabs( tracksTime[iTrack] - sevt->muon[iMuon].time);
        if( (sevt->muon[iMuon].status==1 || sevt->muon[iMuon].status==2) && timeBetweenTrackAndMuon<=4.5 )
        {
            //fprintf(FP2,"\%f\n", chargedPartEPRatio);
            muon = 1; //then we have a muon
        }
    }



    /***Here we calculate the transverse momenta ******************************/
    float beamDirection[3];
    for(int i = 0;i<3;++i)
    {
        beamDirection[i] = beamVelocity[i]/beamVelMag;
    }
    //for the pi0:
    float pi0BeamDirMomMag = f3vdot(pi0VectMom,beamDirection);
    float pi0TransMom[3];
    for(int i = 0;i<3;++i)
    {
        pi0TransMom[i] = pi0VectMom[i] - pi0BeamDirMomMag*beamDirection[i];
    }
    float pi0TransMomMag = f3vmag(pi0TransMom);
   // fprintf(FP1,"\%f\n",pi0TransMomMag);
    //the track:
    float trackBeamDirMomMag = f3vdot(chargedPartVectMom,beamDirection);
    float trackTransMom[3];
    for(int i = 0;i<3;++i)
    {
        trackTransMom[i] = chargedPartVectMom[i] - trackBeamDirMomMag*beamDirection[i];
    }
    float trackTransMomMag = f3vmag(trackTransMom);

    float addPhotonsTransMomMag=0;
    float addPhotonsTransMom[3]={0,0,0};
    if(numberOfAdditionalPhotons>0)
    {
        for(int s=0;s<numberOfAdditionalPhotons;++s)
        {
            float photonBeamDirMomMag = f3vdot(&additionalGamma4Mom[s][1] ,beamDirection);

            for(int i = 0;i<3;++i)
            {
                addPhotonsTransMom[i] += additionalGamma4Mom[s][i+1] - photonBeamDirMomMag*beamDirection[i];
            }
        }
        addPhotonsTransMomMag=f3vmag(addPhotonsTransMom);
    }

    //event:
    float eventTransMom[3];
    for(int i = 0;i<3;++i)
    {
        eventTransMom[i] = trackTransMom[i] + pi0TransMom[i]+addPhotonsTransMom[i];
    }
    float eventTransMomMag = f3vmag(eventTransMom);
```

```
   // fprintf(FP2,"\%f\n",eventTransMomMag);

    ++nEvents;

    /**** Here we I.D. The Track ******************************************************/
   int trackID;
    float missingMassSquared;
    //positron selection criteria
    if(chargedPartEPRatio >0.943718 && muon==0 )
    {
        ++ke3Signal;
        trackID = ELECTRON;
        missingMassSquared = pow(kaonEnergy,2) + pow(pi0Energy,2) + pow(ELECTRON_MASS, 2) + pow(chargedPartMomMag,2) +
            2*pi0Energy*sqrt( pow(ELECTRON_MASS, 2) + pow(chargedPartMomMag,2) ) - 2*kaonEnergy*pi0Energy -
            2*kaonEnergy*sqrt( pow(ELECTRON_MASS, 2) + pow(chargedPartMomMag,2) ) - pow(missingMomMag,2);
       if(missingMassSquared <-0.0081 || missingMassSquared >0.0089)
        {
            ++ke3MissingMassSquaredCut;
            //return -1;
        }
        else if(eventTransMomMag <0.0294072)
        {
            ++ke3EventTransMomCut;
            //return -1;
        }
        else
            ++ke3Count;

    }
    //muon selection criteria
    else if( (pi0Pi1Mass <0.4772 || pi0Pi1Mass >0.5102) && muon==1 && chargedPartEPRatio <0.2)
    {
        ++km3Signal;

        trackID = MUON;
        missingMassSquared = pow(kaonEnergy,2) + pow(pi0Energy,2) + pow(MUON_MASS, 2) + pow(chargedPartMomMag,2) +
            2*pi0Energy*sqrt( pow(MUON_MASS, 2) + pow(chargedPartMomMag,2) ) - 2*kaonEnergy*pi0Energy -
            2*kaonEnergy*sqrt( pow(MUON_MASS, 2) + pow(chargedPartMomMag,2) ) - pow(missingMomMag,2);
        if(missingMassSquared <-0.0067 || missingMassSquared >0.0058)
        {
            ++km3MissingMassSquaredCut;
            //return -1;
        }
        else if(eventTransMomMag <0.0294072)
        {
            ++km3EventTransMomCut;
            //return -1;
        }
        else if(pi0MuonMass >0.38)
        {
            ++km3Pi0MuonMassCut;
        }
        else
            ++km3Count;
    }
    //pi+ selection criteria
    else if( pi0Pi1Mass >0.4772 && pi0Pi1Mass <0.5102 && chargedPartEPRatio <0.943718 )
    {
        ++k2piSignal;

        trackID = PIPLUS;
        missingMassSquared = pow(kaonEnergy,2) + pow(pi0Energy,2) + pow(PI1_MASS, 2) + pow(chargedPartMomMag,2) +
            2*pi0Energy*sqrt( pow(PI1_MASS, 2) + pow(chargedPartMomMag,2) ) - 2*kaonEnergy*pi0Energy -
            2*kaonEnergy*sqrt( pow(PI1_MASS, 2) + pow(chargedPartMomMag,2) ) - pow(missingMomMag,2);
        //fprintf(Datak2piFP,"\%f\n",pi0Pi1Mass);
        if( missingMassSquared <-0.0014 || missingMassSquared >0.0002)
```

```
        {
            ++k2piMissingMassSquaredCut;
            //return -1;
        }
        else if(pi0Pi1Mass <0.488||pi0Pi1Mass >0.497 )
        {
            ++k2pipi0Pi1MassCut;
        }
        else if(eventTransMomMag >0.0075)
        {
            ++k2piEventTransMomCut;
            //return -1;
        }
        else
            ++k2piCount;
    }
    else
        ++numUnidentified;










    free(tracklessClusters);

    //fprintf(cutWhichKilledEventFP,"\%d\n",cutWhichKilledEvent);


   //  nuserevt++;

    /*------------- End of user C code -------------*/
    return 0;
}
```

Certification of ownership

Project Report/Dissertation presented as part of, and in accordance with, the requirements for the Final Degree of BSc at the University of Bristol, Faculty of Science.

I hereby assert that I own exclusive copyright in the item named below. I give permission to the University of Bristol Library to add this item to its stock and to make it available for consultation in the library, and for inter-library lending for use in another library. It may be copied in full or in part for any bona fide library or research worked, on the understanding that users are made aware of their obligations under copyright legislation, i.e. that no quotation and no information derived from it may be published without the authors prior consent.

Author: Ben Crabbe
Title: A Measurement of Charged Kaon Semileptonic Decay Braching Fractions on NA62 P5
Date submitted: May 1, 2014

signed: Ben Crabbe

full name: Benjamin Crabbe

Date: May 1, 2014