UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSI

INSTITUTO DE FISICA

Estudio de la dispersión kaón-electrón en el experimento NA62 para la obtención del radio de carga eléctrico promedio del kaón.

TESIS PARA OBTENER EL GRADO DE:

Maestría en Física

PRESENTA:

Lic. Fis. Alan Efraín Martínez Hernández

Director de Tesis:

Dr. Jürgen Engelfried

San Luis Potosí, S. L. P. Septiembre

Contenidos

Resumen:

Se estudia la dinámica de la dispersión de un haz de kaones que se hace incidir en un blanco de electrones. Se estudian las distribuciones de ángulo y energía, así como la sección transversal para la obtención del factor de forma eléctrico del kaón. Se propone analizar los datos del experimento NA62 del CERN para la obtención del radio de carga eléctrico promedio del kaón.

Abstract:

The dynamics of the dispersion of an incident beam of kaons in a target of electrons is studied. The angle and energy distributions are studied, as well as the cross section to obtain the electrical form factor of the kaon. It is proposed to analyze the data from the NA62 experiment at CERN to obtain the average electric charge radius of the kaon.

Capítulo 1:
Motivación e introducción.

El objetivo principal de la física de partículas es el estudio de los componentes elementales de la materia y las interacciones entre ellos. A esta rama de la física se le conoce también como física de altas energías, debido a que las partículas se aceleran hasta alcanzar casi la velocidad de la luz, con el objetivo de tener un momento muy grande implicando así una longitud de onda muy pequeña, del orden del tamaño o más pequeño del objeto a estudiar. Una vez aceleradas, las partículas se hacen colisionar para así poder observar los productos de estas colisiones o bien, debido a que la mayor parte de las partículas son altamente inestables, estas decaen en otras partículas.

Para poder estudiar estos fenómenos, alrededor del mundo se han construido aceleradores de partículas cada vez más y más energéticos con el propósito de obtener mejores resultados. Entre todos los posibles experimentos que se pueden llevar a cabo en estos aceleradores, se encuentran los experimentos de dispersión, para medir el tamaño de las partículas; los experimentos de colisión, para observar los elementos fundamentales que componen a las partículas; y los experimentos de decaimientos, para medir tiempos de vida y los porcentajes de probabilidad de cada modo de decaimiento de una partícula.

En los experimentos de dispersión se usa un haz de electrones para bombardear un blanco, el cual está compuesto por la partícula a estudiar, para después obtener las distribuciones de energías y ángulos de los productos del choque, Con estas distribuciones nos es posible estimar el radio de carga promedio de la partícula. Pero en el caso en el que la partícula a estudiar es inestable no es posible usar un blanco hecho de estas partículas, por lo que se aplica la técnica de la dinámica inversa. Esta técnica consiste en que ahora se usan los electrones como blanco y se utiliza un haz con las partículas a estudiar. Esto proceso es válido ya que las partículas del haz viajan a velocidades cercanas a la velocidad de la luz, por lo que se aplican los métodos de la relatividad especial y los cuadrivectores al cuadrado de cada partícula se vuelen cantidades invariantes al sistema referencia.

Este tipo de experimentos se han llevado a cabo a través de los años, los cuales se han ido mejorando con la implementación de haces cada vez más energéticos y detectores más sensibles para una mejor toma de datos. Publicaciones anteriores muestran que es un proceso útil para la obtención del factor de forma de diversas partículas, como es el caso del $^{40}Ca,\ ^{48}Ca$ [2], ^{12}C [3], Σ^- [4], kaón [5], pion [6], el neutrón [8] y protón [9,10].

Uno de los experimentos actuales más importantes es el de la crisis del tamaño del protón [26]. Medido usando átomos de hidrogeno. La estructura electrónica del átomo de hidrógeno (cómo se distribuyen los electrones en diferentes niveles energéticos) depende de dos parámetros principales, la constante de Rydberg, que determina la escala de energía de toda la física atómica y la química, y el radio de carga del protón. Hay diversos experimentos para medir ambos parámetros, como la medida de la transición 1S–2S que permite alcanzar 15 dígitos de precisión [27],(ya que el nivel 2S tiene una larga vida, ∼1 s, y no está afectado por el principio de indeterminación de Heisenberg); también se pueden usar las transiciones entre el nivel 2S y diferentes niveles nS, nP, o nD (llamados estados de Rydberg) [28]. También se pueden realizar medidas del desplazamiento Lamb (la separación de los niveles 2S–2P). Finalmente, se puede medir el radio del protón de forma

independiente usando medidas de dispersión electrón-protón [29]. En 2010 todos estos métodos apuntaban a un valor de 0.88 ± 0.011 fm; pero la medida usando el hidrógeno muónico (un protón rodeado de un muón en lugar de un electrón), se obtuvo un valor de 0.84 ± 0.019 fm de precisión extrema (porque la masa del muón es 207 veces mayor que la del electrón). Articulo publicado en 2020 por Alexey Grinin, Arthur Matveev, …, Thomas Udem, "Two-photon frequency comb spectroscopy of atomic hydrogen. [24].

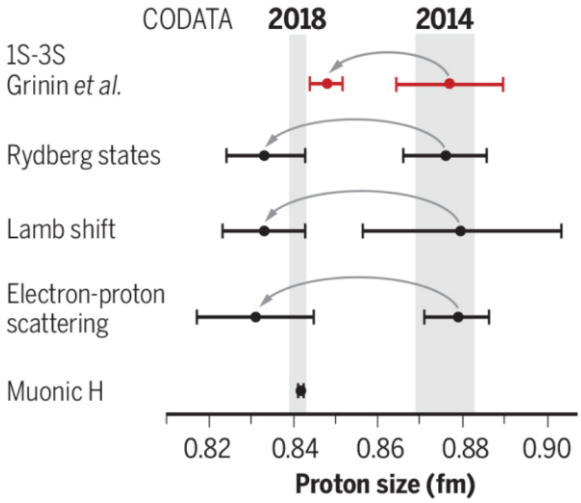


Figura 1.1: Comparación de los diferentes resultados de los experimentos para la medición del radio de carga del protón.

Otro de los experimentos actuales es el experimento MINERvA, es un experimento de dispersión de neutrinos que usa el NuMI beamline en Fermilab. Está completamente construido y toma datos continuamente. Se miden las interacciones neutrino (antineutrino)-nucleón para apoyar los experimentos de oscilación de neutrinos y para estudiar la dinámica de las interacciones fuertes de los núcleos y nucleones que afectan esas interacciones, entre otras muchas posibles mediciones. MINERvA es el primer experimento en el mundo en utilizar haces de neutrinos (antineutrinos) a alta intensidad para estudiar las interacciones con cinco diferentes núcleos simultáneamente siendo el primero en hacer comparaciones entre distintos elementos de manera autónoma.

El propósito de este trabajo es la obtención del radio de carga eléctrico promedio del Kaon positivo (K^+, la cual es una partícula inestable) mediante un experimento de dispersión usando un haz secundario de 75 GeV (constituido por 70% piones, 23% protones y 6% de kaones) proveniente del acelerador SPS del CERN. Como blanco de electrones se usó la tercera estación del espectrómetro GTK del experimento NA62. A pesar de que el experimento NA62 está diseñado para medir principalmente decaimientos de piones y kaones, tiene los detectores necesarios para también medir dispersión. Mide ángulos, energías y momento, los cuales son esenciales para obtener el radio de carga de una partícula.

El código principal del NA62 está programado para analizar datos provenientes de decaimientos, por lo que no cuenta con subrutinas especializadas en el análisis de dispersión. Debido a esto, en este trabajo se hizo un análisis meramente teórico, teniendo como resultado las distribuciones esperadas

para las energías y ángulos en función de la transferencia de cuadrimomento al cuadrado, así como una estimación del factor de forma eléctrico con valores arbitrarios del radio de carga.

Como trabajo futuro queda la implementación de nuevas subrutinas al código principal del NA62 que nos permitan obtener y analizar los datos provenientes de la dispersión, no solo de kaones, sino ahora también de protones y piones. Una vez teniendo estas subrutinas, podremos comparar los resultados de este trabajo con los arrojados del análisis.

En los capítulos siguientes se introducirá un poco a la historia de la física de altas energías, así como una breve discusión de las partículas fundamentales y de las interacciones elementales en la naturaleza, así como sus mediadores.

En el capítulo tres se hablará de los primeros experimentos de dispersión, así como de los principales aceleradores de partículas que existen en la actualidad. Se introducirá también el concepto de sección transversal, el cual es muy importante en el entendimiento del tamaño de una partícula.

En el capítulo cuatro se introduce el concepto de factor de forma y de radio de carga cuadrático medio. Se da también una justificación del porque se usa el método de cinemática inversa, así como un desarrollo matemático de las ecuaciones de movimiento presentes en el experimento.

Las distribuciones esperadas para el ángulo y las energías del electrón y del kaón se presentan en el capítulo cinco, así como una breve presentación del kaón. Se presenta también una aproximación del factor de forma eléctrico del kaón para diferentes valores arbitrarios del radio de carga cuadrático medio.

En el capítulo seis se habla sobre el experimento NA62. Se da una explicación detallada sobre la posición y el funcionamiento de cada detector presente en este experimento, así como el rango de ángulos que se pueden medir en la dispersión.

Una pequeña explicación sobre el proceso experimental que se plantea aplicar al código del NA62, se expone en el capítulo siete. También se presentan nuevamente las distribuciones esperadas expuestas en el capítulo cinco, pero ahora acotadas por el rango de ángulos que nos permite medir la distribución de los detectores en el experimento.

En los últimos capítulos de este trabajo se presenta la conclusión y se propone trabajo futuro para continuar con este análisis. El cual consiste en terminar la implementación de nuevas subrutinas para la obtención de datos provenientes de la dispersión tanto de kaones, como de piones y protones. Todos estos presentes en el haz del NA62.

Capítulo 2:
Componentes fundamentales de la materia.

En la búsqueda de los componentes fundamentales de la materia, los físicos han encontrado elementos cada vez más y más pequeños los cuales se ha probado que ellos mismos son sistemas compuestos. Para finales del siglo XIX se sabía que toda la materia estaba compuesta por átomos, sin embargo, la existencia de estados excitados y de experimentos que muestran propiedades periódicamente recurrentes fue una clara indicación que los átomos mismos tienen una estructura interna, por lo tanto, se rechazó la idea de que los átomos eran indivisibles.

El concepto moderno del átomo emergió a principios del siglo XX, en particular como un resultado de los experimentos de Rutherford. Un átomo está compuesto de un núcleo denso rodeado de una nube de electrones. El núcleo puede ser descompuesto en partículas más pequeñas. Después del descubrimiento del protón en 1918 y del neutrón en 1932, no había duda de que los componentes del núcleo eran los protones y los neutrones, llamados en su conjunto nucleones. En 1930 se postuló la existencia de una cuarta partícula sin carga y masa despreciable, el neutrino (descubierto en 1953). Fue propuesto para ajustar la descripción del decaimiento beta con la conservación de energía y momento.

A mediados de los años treinta estas cuatro partículas; electrón, protón, neutrón y neutrino, podían describir todos los fenómenos conocidos hasta entonces de la física atómica y nuclear. Hoy en día sabemos que estos cuatro nos son suficientes para describir otros fenómenos. Los experimentos realizados en los años cincuenta y sesenta en los aceleradores de partículas mostraron que los protones y neutrones son los principales representantes de una larga familia de partículas llamados hadrones, que pueden ser clasificados en grupos con propiedades similares. A finales de los sesenta, el modelo de los quarks estableció un orden en la familia hidrómica, los hadrones conocidos pueden ser descritos como una combinación de dos o tres quarks. Mas de 100 hadrones distintos han sido detectados.

2.1 Leptones y quarks.

Los dos tipos fundamentales de componentes de la materia son los leptones, los cuales incluyen al electrón y al neutrino, y los quarks. En los experimentos de difracción, el límite de tamaño que se ha podido medir es del orden de 10^{-18} m, mil veces más pequeños que el protón. Leptones y quarks tienen spin ½, por lo tanto, son fermiones. Hasta entonces no se han observado estados excitados de quarks o leptones, por lo aparentan ser partículas elementales.

Los nucleones formados por quarks reciben el nombre de hadrones, los cuales se clasifican según el número de quarks que contienen, bariones y mesones. Los bariones están formados por una combinación de tres quarks, mientras que los mesones están formados solo de dos quarks, un quark y un antiquark.

Ya que los quarks tienen spin 1/2, los bariones tienen spin semi entero por lo tanto son fermiones, mientras que los mesones tienen spin entero, por lo que son bosones. Se han encontrado seis quarks

diferentes, con su respectivo antiquark, los cuales pueden ser agrupados en familias según su carga y su masa.

$$\binom{u}{d} \quad \binom{c}{s} \quad \binom{t}{b}$$

Los quarks u, c y t tienen carga eléctrica de +2/3 e (sus respectivos antiquarks tienen carga eléctrica de -2/3 e), mientras que los quarks d, s y t tienen carga eléctrica de -1/3 e (sus respectivos antiquarks tienen carga eléctrica de +1/3 e). Los quarks son portadores de color.

Los leptones son fermiones fundamentales sin color (el color es la carga de la interacción fuerte). Existen seis leptones y sus correspondientes antipartículas. Electrón, muon, tau, y neutrino asociado a cada uno de los anteriores. Todos los leptones cargados tienen una sencilla unidad de carga eléctrica y todos los neutrinos y antineutrinos tienen carga cero.

2.2 Interacciones fundamentales.

Con el cambio de concepción de las partículas elementales, nuestro entendimiento de las fuerzas básicas de la naturaleza y las interacciones fundamentales entre estas partículas ha evolucionado. Alrededor del año 1800, se consideraban cuatro fuerzas básicas: gravitación, electricidad, magnetismo y las poco comprendidas fuerzas entre átomos y moléculas. A finales del siglo XIX, electricidad y magnetismo fueron entendidas como manifestaciones de una misma fuerza: electromagnetismo.

Cuando se desarrolló la física nuclear, dos nuevas fuerzas de corto alcance se unieron al juego; la fuerza nuclear fuerte y la fuerza débil. La fuerza nuclear fuerte actúa entre los nucleones, une a los quarks para formar protones y neutrones, mientras que la fuerza nuclear débil se manifiesta en decaimientos nucleares, intercambio de bosón $\pm$W, cambio de quark a otro quark y cambia de un leptón a otro. A finales de 1960 se unifico la fuerza electromagnética con la fuerza nuclear débil en la ahora llamada fuerza electrodébil.

2.3 Bosones.

Las cuatro interacciones fundamentales son entonces, gravitación, electromagnética, nuclear fuerte y nuclear débil. La gravitación es importante para la existencia de galaxias, estrellas y sistemas planetarios (y para la vida cotidiana). Sin embargo, esta no juega un rol importante en física subatómica, ya que es muy débil en comparación con las demás, no se nota su influencia en las interacciones entre partículas fundamentales. Solo la menciono por completes.

De acuerdo con las concepciones actuales, las interacciones son mediadas por el intercambio de partículas llamadas bosones, de spin entero. Estos son: el fotón para la interacción electromagnética, gluones en la interacción nuclear fuerte y los bosones $W^+, W^- y\, Z^0$ en la interacción nuclear débil, El gravitón con spin 2, se cree que es el mediador de la interacción gravitacional. Aun no se ha observado.

Cada una de estas cuatro interacciones es asociada a una carga: carga eléctrica, carga fuerte, carga débil y la masa. La carga fuerte es llamada también color. Una partícula es sujeta a una interacción si y solo si lleva la carga correspondiente:

Leptones y quarks llevan carga débil.
Los quarks están eléctricamente cargados, algunos leptones también (electrón).
El color solo es llevado por los quarks.

Existe otra partícula fundamental, la cual no pertenece a ninguno de los grupos antes mencionados, el bosón de Higgs que tiene spin 0. El bosón de Higgs fue propuesto por Peter Higgs en 1964 para explicar el origen de la masa de las partículas elementales en el marco de la unificación electrodébil. Este bosón constituye el cuanto del campo de Higgs. Fue descubierta en 2012 por los experimentos ATLAS y CMS en CERN.

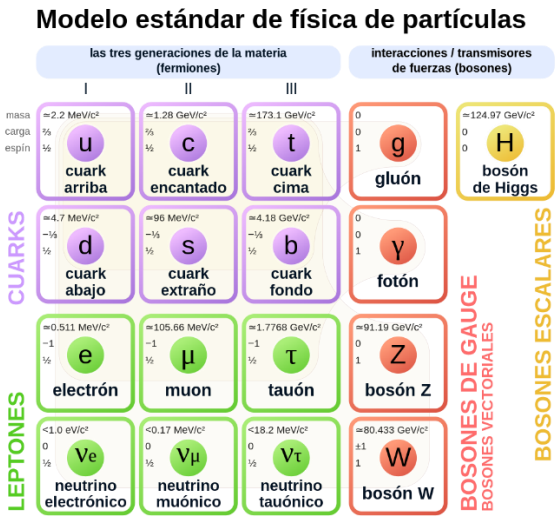


Figura 2.1. Partículas elementales del modelo estándar.

Capítulo 3:

Experimentos de dispersión.

La mayoría de los experimentos en física nuclear y partículas elementales han tenido que ser llevados a cabo en aceleradores de partículas. El desarrollo y construcción de nuevos aceleradores con más energía en el haz ha hecho posible el descubrimiento de más partículas elementales, así como de su estructura interna. Uno de los principales experimentos que se pueden realizar en un acelerador de partículas son los experimentos de dispersión. Estos experimentos son usados para estudiar los detalles de la interacción entre diferentes partículas y para obtener información acerca de la estructura interna de núcleos atómicos y sus partes. En un experimento típico de dispersión, un haz de partículas con energía y momento conocido es dirigido hacia el objeto que se desea estudiar. A partir de los cambios en las cantidades cinemáticas causadas por esta interacción, podemos indagar acerca de las propiedades tanto del blanco como de la misma interacción. Se ha probado que la difracción elástica con electrones es un método confiable para medir el radio de las partículas, la estructura del blanco se vuelve más visible cuando la longitud de onda de Broglie $\lambda = h/p$ del electrón es menor que el tamaño del blanco. El patrón de difracción resultante muestra el tamaño del núcleo con bastante precisión. Hoy en día es posible producir haces de una gran variedad de partículas (electrones, protones, neutrones, iones pesados, …) con energías que van desde 10^{-3} eV para neutrones "fríos" hasta 10^{12} eV para protones. Incluso es posible producir haces de partículas secundarias las cuales a su vez han sido producidas en reacciones de altas energías. Estos haces viven por un periodo de tiempo muy corto, tal es el caso de muones, π y K, o hiperones ($\Sigma^{\pm}, \Xi^{-}, \Omega^{-}$).

Blancos sólidos, líquidos o gaseosos son usados como material de dispersión o, en experimentos de anillos, otro haz de partículas puede ser usado como el blanco. Ejemplos de este último son el gran colisionador de electrones y positrones (LEP) en CERN, Geneva, con un máximo de energía del haz de 86 GeV; el "Tevatron" en el laboratorio de aceleración nacional Fermi con una energía de 900 GeV en un haz de proton-antiproton, el anillo HERA en Hamburgo con una energía de 920 GeV para el protón y 30 GeV para el electrón y el LHC también en CERN con 7 TeV de energía del haz.

3.1 Dispersión elástica.

El desarrollo físico y matemático presentado a continuación, se basó en el libro de texto Particles and Nuclei, 6th Edition, de Bogdan Povh, 2008.

En procesos elásticos las partículas son las mismas antes y después de la interacción.

$$a + b \rightarrow a' + b'$$

El blanco b permanece en su estado base, absorbiendo parte del momento del haz cambiando también su energía cinética. El apostrofe en las letras indican que las partículas permanecen idénticas a excepción de su energía y momento. La energía y ángulo de dispersión de la partícula a´ y la energía y ángulo de producción de la partícula b´estan correlacionados. Se pueden sacar conclusiones acerca de la forma espacial del objeto de dispersión a partir de la dependencia de la tasa de dispersión y la energía y el ángulo de dispersión. Para visualizar las estructuras de objetos pequeños se requiere que

la energía del haz sea más grande, ya que la longitud de onda reducida de de Broglie viene dada por la expresión:

$$\frac{\lambda}{2\pi} = \frac{\hbar}{p} = \frac{\hbar c}{\sqrt{2mc^2 E_{kin} + E_{kin}^2}} \approx \begin{cases} \dfrac{\hbar}{\sqrt{2mE_{kin}}} & for\ E_{kin} \ll mc^2 \\ \dfrac{\hbar c}{E_{kin}} \approx \dfrac{\hbar c}{E} & for\ E_{kin} \gg mc^2 \end{cases} \tag{3.1}$$

Donde λ es la longitud de onda, m la masa de la partícula, E_{kin} es la energía cinética, p el momento, c la velocidad de la luz, y $\hbar$ la constante reducida de Planck.

Para observar una estructura de extensión lineal Δx, el tamaño de la longitud de onda deber ser del mismo orden: $\frac{\lambda}{2\pi} \lesssim \Delta x$.

Del principio de incertidumbre de Heisenberg el momento correspondiente una partícula es:

$$\Delta p \gtrsim \frac{\hbar}{\Delta x}, \qquad pc \gtrsim \frac{\hbar c}{\Delta x} \approx \frac{200\ MeV\ fm}{\Delta x}. \tag{3.2}$$

Entonces, para estudiar núcleos cuyo radio es de unos pocos fm se necesita un haz cuyo momento sea del orden de 10 – 100 MeV/c. Para estudiar las partes que constituyen al núcleo, los quarks, uno tiene que penetrar más allá del interior del núcleo. Para ello, se necesita un haz con momento de muchos GeV.

3.2 Dispersión inelástica.

En reacciones inelásticas parte de la energía cinética transferida del proyectil al blanco, se usa para excitar al blanco y llevarlo a un nivel más alto de energía. Este estado excitado regresara a su estado base emitiendo una partícula ligera o podría decaer en dos o más partículas diferentes. En este trabajo solo se abarca procesos elásticos. Solo se menciona la dispersión inelástica por completes.

3.3 Sección transversal.

La tasa de reacción medida en experimentos de difracción, el espectro de energía y las distribuciones angulares de los productos proveen información acerca de la dinámica de la interacción entre el proyectil y el objetivo, por ejemplo, la forma del potencial de la interacción o la fuerza de acoplamiento. La cantidad más importante para la descripción e interpretación de estas interacciones es la llamada sección transversal σ, la cual es una medida de la probabilidad de una reacción entre dos partículas que colisionan.

Consideremos un experimento ideal. Imagine un blanco de dispersión delgado de ancho d con N_b centros de dispersión b y con una densidad de partículas n_b. Cada partícula del blanco tiene un área de sección transversal σ_b, el cual será determinado con el experimento. Bombardeamos el blanco con un haz monoenergético de partículas a. La reacción ocurre cuando una partícula del haz golpea una partícula del blanco y asumimos que la partícula del haz es removida del haz. La tasa de reacción total $\dot{N}$, es decir, el número total de reacciones por unidad de tiempo es dada por la diferencia entre

la tasa de partículas de haz $\dot{N}_a$ antes y después del blanco. Esta es una medida directa del área de sección transversal σ_b.

Ahora asumimos que el haz tiene un área de sección transversal A y una densidad de partículas n_a. El número de proyectiles golpeando al blanco por unida de área y por unidad de tiempo es llamado flujo Φ_a. Es el producto de la velocidad y densidad de las partículas de haz:

$$\Phi_a = \frac{\dot{N}_a}{A} = n_a * v_a \tag{3.3}$$

El número total de partículas del blanco que abarca el área del haz es $N_b = n_b * A * d$. Así, la tasa de reacción $\dot{N}$ esta dada por el producto del flujo y el área total de sección transversal vista por las partículas:

$$\dot{N} = \Phi_a * N_b * \sigma_b \tag{3.4}$$

Esta ecuación es válida siempre y cuando los centros de difracción no se superponen y las partículas son solo difractadas por centros individuales. El área que presenta un centro de difracción a una partícula entrante será llamada la sección transversal geométrica de reacción:

$$\sigma_b = \frac{\dot{N}}{\Phi_a * N_b} \tag{3.5}$$

Esto definición asume un haz constante y homogéneo. La probabilidad de reacción para dos partículas es generalmente muy diferente a lo que implicarían estas consideraciones geométricas. Además, se observa una fuerte dependencia energética. La forma, fuerza y rango del potencial de interacción, determinan ante todo el área efectiva de sección transversal. La interacción puede ser determinada de la tasa de reacción si se conoce el flujo de las partículas del haz y la densidad de área de los centros de difracción.

La sección transversal es una cantidad física con dimensiones de área y es independiente del diseño específico del experimento. La unidad comúnmente usada es el barn, el cual se define como:

$$1\ barn = 1\ b = 10^{-28}\ m^2$$

En la práctica, sólo una fracción de todas las reacciones son medida. Un detector de área A_D es colocado a una distancia r y a un ángulo θ con respecto a la dirección de haz, cubriendo un ángulo sólido $\Delta\Omega = A_D/r^2$. La tasa de reacciones vista por este detector es entonces proporcional a la sección transversal diferencial $d\sigma(E,\theta)/d\Omega$. Si el detector puede determinar la energía de las partículas de difractadas entonces uno puede medir la doble diferencial de la sección transversal $d^2\sigma(E,E',\theta)/d\Omega dE'$. La sección transversal total es entonces la integral sobre todo el ángulo sólido y sobre todo las energías de difracción:

$$\sigma_{tot}(E) = \int_0^{E'_{max}} \int_{4\pi} \frac{d^2\sigma(E,E',\theta)}{d\Omega dE'} d\Omega dE' \tag{3.6}$$

3.4 Regla de oro.

Como ya vimos la sección transversal puede ser determinada experimentalmente a partir de la tasa de reacción, observemos ahora como puede ser determinada de la teoría.

Primero, la tasa de reacción es dependiente de las propiedades del potencial de interacción descrito por el operador hamiltoniano H_{int}. En una reacción este potencial transforma la función de onda inicial ψ_i a una función de onda final ψ_f. El elemento de la matriz de transición es dado por:

$$M_{if} = \langle \psi_f | H_{int} | \psi_i \rangle = \int \psi_f^* H_{int} \psi_i dV \tag{3.7}$$

Este elemento de matriz es también llamado la amplitud de probabilidad para la transición. Además, la tasa de reacción dependerá del número de estados finales disponibles para la reacción. De acuerdo con el principio de incertidumbre cada partícula ocupa un volumen $h^3 = (2\pi\hbar)^3$ en el espacio fase, el espacio seis dimensional de momento y posición. Considere una partícula difractada en un volumen V y en un intervalo de momento entre p' y $p' + dp'$. En el espacio de momentos, el intervalo corresponde a un cascarón esférico con radio interno p' y ancho dp' con un volumen de $4\pi p'^2 dp'$. Excluyendo los procesos donde el spin cambia el número final de estados disponibles es:

$$dn(p') = \frac{V \cdot 4\pi p'^2}{(2\pi\hbar)^3} dp' \tag{3.8}$$

La energía y momento de una partícula están conectadas por:

$$dE' = v' dp' \tag{3.9}$$

Por eso la densidad de estados finales en el intervalo energía dE' es dado por:

$$\varrho(E') = \frac{dn(E')}{dE'} = \frac{V 4\pi p'^2}{v'(2\pi\hbar)^3} \tag{3.10}$$

La conexión entre la tasa de reacción, el elemento de la matriz de transición y la densidad de estados finales es expresada por la segunda regla de oro de Fermi. Esta expresa la tasa de reacción W por objetivo y por partícula del haz en la forma:

$$W = \frac{2\pi}{\hbar} |M_{fi}|^2 \cdot \varrho(E') \tag{3.11}$$

Además, sabemos de (3.3) y (3.4) que:

$$W = \frac{\dot{N}(E)}{N_b N_a} = \sigma \cdot \frac{v_a}{V} \tag{3.12}$$

Entonces la sección transversal queda:

$$\sigma = \frac{2\pi}{v_a \hbar} |M_{fi}|^2 \cdot \varrho(E') \cdot V \tag{3.13}$$

Si se conoce el potencial de interacción, la sección transversal puede ser calculada de (3.13), por otro lado, los datos de la sección transversal y la ecuación (3.13) pueden ser usados para determinar el elemento de la matriz de transición.

3.5 Sección transversal de Rutherford.

Consideremos ahora la sección transversal de un electrón con energía E difractado de un núcleo atómico con carga Ze. Para que los cálculos de la reacción sean suficientemente precisos deben ser tomados en cuenta la relatividad y la mecánica cuántica. Primero introducimos la fórmula de

difracción de Rutherford. Por definición los efectos del spin son despreciables. Para núcleos pesados y electrones de baja energía el retroceso también puede ser despreciado. En este caso la energía y el módulo del momento son los mismos antes y después de la dispersión. Siempre y cuando el radio del centro de dispersión sea más pequeño que la distancia más cercana al proyectil entonces la extensión espacial del centro de dispersión no afecta este cálculo puramente clásico. Esto lleva a la fórmula de Rutherford para la dispersión de una partícula con carga ze y energía cinética E y un núcleo como blanco con carga Ze.

$$\left(\frac{d\sigma}{d\Omega}\right)_R = \frac{(zZe^2)^2}{(4\pi\varepsilon_0)^2(4E_{kin})^2 sin^4\frac{\theta}{2}} \tag{3.14}$$

Se obtiene exactamente la misma ecuación en la mecánica cuántica no relativista usando la regla de oro de Fermi. Consideremos el caso de un blanco pesado de manera que podemos despreciar el retroceso. Usando tri-momento y si Ze es pequeño, la aproximación de Born puede ser aplicada y las funciones de onda del electrón antes y después de la dispersión pueden ser descritas como ondas planas.

$$\psi_i = \frac{1}{\sqrt{V}} e^{\boldsymbol{ipx}/\hbar} \qquad \psi_f = \frac{1}{\sqrt{V}} e^{\boldsymbol{ip'x}/\hbar} \tag{3.15}$$

Podemos dejar de lado cualquier dificultad relacionada a la normalización de las funciones de onda considerando solamente un volumen finito V. Necesitamos que este volumen sea lo suficientemente grande comparado al centro de dispersión para que los estados de energía discretos en este volumen puedan ser aproximados por un continuo.

Consideremos un haz electrones con una densidad n_a de partículas por unidad de volumen. Con el volumen de integración escogido lo suficientemente grande la condición de normalización está dada por:

$$\int_V |\psi_i|^2 dV = n_a V \qquad Donde\ V = \frac{N_a}{n_a} \tag{3.16}$$

La tasa de reacción está dada por el producto de la sección transversal y la velocidad del haz, dividido entre el volumen. Aplicando también la regla de oro tenemos:

$$W = \frac{\sigma v_a}{V} = \frac{2\pi}{\hbar} \left|\langle \psi_f | H_{int} | \psi_i \rangle\right|^2 \frac{dn}{dE_f} \tag{3.17}$$

E_f es la energía total (energía cinética y masa en reposo) del estado final. Ya que despreciamos el retroceso y la masa reposo es una constante, $dE_f = dE' = dE$.

La densidad n de los posibles estados finales en el espacio fase es, como en la ecuación (3.8):

$$dn(|\boldsymbol{p}'|) = \frac{V \cdot 4\pi |\boldsymbol{p}'|^2}{(2\pi\hbar)^3} d|\boldsymbol{p}'| \tag{3.18}$$

Por lo tanto, la sección transversal para la dispersión de un electrón en un diferencial de ángulo sólido $d\Omega$:

$$d\sigma \cdot v_a \cdot \frac{1}{V} = \frac{2\pi}{\hbar} \left|\langle \psi_f | H_{int} | \psi_i \rangle\right|^2 \frac{V \cdot |\boldsymbol{p}'|^2 d|\boldsymbol{p}'|}{(2\pi\hbar)^3 dE_f} d\Omega \tag{3.19}$$

La velocidad v_a puede ser reemplazada, como una buena aproximación, por la velocidad de la luz c. Para energías grandes del electrón, $|\boldsymbol{p}'| \approx E'/c$ aplica, obteniendo:

$$\frac{d\sigma}{d\Omega} = \frac{V^2 E'^2}{(2\pi)^2 (\hbar c)^4} \left|\langle \psi_f | H_{int} | \psi_i \rangle\right|^2 \tag{3.20}$$

El operador de interacción electromagnético para una carga e en un potencial eléctrico ϕ es $H_{int} = e\phi$. El elemento de matriz queda como:

$$\langle \psi_f | H_{int} | \psi_i \rangle = \frac{e}{V} \int e^{-i\boldsymbol{p}'\boldsymbol{x}/\hbar} \phi(\boldsymbol{x}) e^{i\boldsymbol{p}\boldsymbol{x}/\hbar} d^3x \tag{3.21}$$

Definiendo la transferencia de momento **q** como:

$$\boldsymbol{q} = \boldsymbol{p} - \boldsymbol{p}' \tag{3.22}$$

Podemos reescribir el elemento de matriz como:

$$\langle \psi_f | H_{int} | \psi_i \rangle = \frac{e}{V} \int \phi(\boldsymbol{x}) e^{i\boldsymbol{q}\boldsymbol{x}/\hbar} \, d^3x \tag{3.23}$$

Aplicando el teorema de Green de la siguiente manera:

$$e^{i\boldsymbol{p}\boldsymbol{x}/\hbar} = \frac{-\hbar^2}{|\boldsymbol{q}|^2} \cdot \nabla^2 e^{\frac{i\boldsymbol{q}\boldsymbol{x}}{\hbar}} = \frac{-\hbar^2}{|\boldsymbol{q}|^2} \cdot \Delta e^{\frac{i\boldsymbol{q}\boldsymbol{x}}{\hbar}} \qquad con\ \nabla^2 = \Delta \tag{3.24}$$

Podemos reescribir el elemento de matriz como:

$$\langle \psi_f | H_{int} | \psi_i \rangle = \frac{-\hbar^2 e}{|\boldsymbol{q}|^2 V} \int \Delta\phi(\boldsymbol{x}) e^{i\boldsymbol{q}\boldsymbol{x}/\hbar} \, d^3x \tag{3.25}$$

El potencial y la densidad de carga están relacionados por la ecuación de Poisson:

$$\Delta\phi(\boldsymbol{x}) = -\frac{\varrho(\boldsymbol{x})}{\varepsilon_0} \tag{3.26}$$

A partir de ahora se asume que la densidad de carga es estática, independiente del tiempo. Se define la función de distribución de carga f como $\varrho(\boldsymbol{x}) = Zef(\boldsymbol{x})$ la cual satisface la condición de normalización $\int f(\boldsymbol{x}) d^3x = 1$ y reescribimos el elemento de la matriz como:

$$\begin{aligned} \langle \psi_f | H_{int} | \psi_i \rangle &= \frac{\hbar^2 e}{\varepsilon_0 |\boldsymbol{q}|^2 V} \int \varrho(\boldsymbol{x}) e^{i\boldsymbol{q}\boldsymbol{x}/\hbar} \, d^3x \\ &= \frac{Z 4\pi\alpha\hbar^3 c}{|\boldsymbol{q}|^2 V} \int f(\boldsymbol{x}) e^{i\boldsymbol{q}\boldsymbol{x}/\hbar} \, d^3x \end{aligned} \tag{3.27}$$

La integral que aparece en la ecuación anterior es la transformada de Fourier de la función de carga $f(\boldsymbol{x})$, normalizada a la carga total. Esta es llamada el factor de forma de la distribución de carga. El factor de forma contiene toda la información acerca de la distribución espacial de la carga del objeto

qué está siendo estudiado. Para calcular la sección transversal de Rutherford se despreció la extensión espacial, es decir reemplazamos la distribución de carga por una función δ. Por eso, el factor de forma se fija como la unidad. Insertando el elemento de matriz en la ecuación (3.20) se obtiene:

$$\left(\frac{d\sigma}{d\Omega}\right)_R = \frac{4Z^2\alpha^2(\hbar c)^2E'^2}{|\boldsymbol{q}c|^4} \tag{3.28}$$

La dependencia sobre q de la sección transversal implica una tasa de eventos muy baja para la dispersión de electrones con mucha transferencia de momento. La tasa de eventos cae tan rápidamente que pequeños errores en la medición de q puede falsificar los resultados.

Ya que el retroceso es despreciable en la dispersión de Rutherford, la energía y la magnitud del momento del electrón no cambian. La magnitud de la transferencia de momento q es entonces:

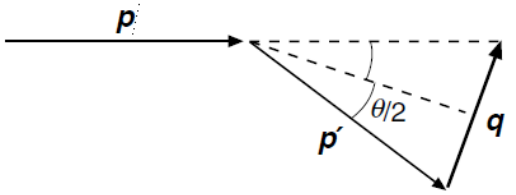


Figura 3.1. Esquema de la transferencia de momento ***p-p'.***

$$|\boldsymbol{q}| = 2 \cdot |\boldsymbol{p}|\sin\frac{\theta}{2} \tag{3.29}$$

Recordando que $E = |\boldsymbol{p}|c$ es una buena aproximación, obtenemos la ecuación relativista para la dispersión de Rutherford:

$$\left(\frac{d\sigma}{d\Omega}\right)_R = \frac{Z^2\alpha^2(\hbar c)^2}{4E^2\sin^4\frac{\theta}{2}} \tag{3.30}$$

3.6 Sección transversal de Mott.

Hasta ahora se ha despreciado el spin del electrón y del blanco. Sin embargo, en energías relativistas la sección transversal de Rutherford es modificada por los efectos del spin. La sección transversal de Mott, la cual describe la dispersión de electrones incluyendo los efectos debido al spin, puede ser descrita como:

$$\left(\frac{d\sigma}{d\Omega}\right)_M^* = \left(\frac{d\sigma}{d\Omega}\right)_R \cdot \left(1 - \beta^2\sin^2\frac{\theta}{2}\right) \quad , \quad con\ \beta = \frac{v}{c} \tag{3.31}$$

El asterisco indica que el retroceso del núcleo ha sido despreciado en la derivación de esta ecuación. La expresión muestra que, en energías relativistas, la sección transversal de Mott cae más rápidamente en ángulos de dispersión grandes a comparación de la sección transversal de Rutherford. En el caso limite cuando $\beta \to 1$, y usando la identidad $\sin^2\theta + \cos^2\theta = 1$, la sección transversal de Mott puede ser escrita de una manera más simple:

$$\left(\frac{d\sigma}{d\Omega}\right)_M^* = \left(\frac{d\sigma}{d\Omega}\right)_R \cdot \cos^2\frac{\theta}{2} = \frac{4Z^2\alpha^2(\hbar c)^2E'^2}{|\boldsymbol{q}c|^4}\cos^2\frac{\theta}{2} \tag{3.32}$$

Capítulo 4:

Dispersión con nucleones.

Actualmente, en los experimentos de dispersión, con núcleos o nucleones, se observa que la sección transversal de Mott concuerda con la sección transversal experimental solo en el límite cuando $|\boldsymbol{q}| \to 0$. Para valores grandes de $|\boldsymbol{q}|$, la sección transversal experimental es sistemáticamente pequeña. Si se consideran sistemas esféricamente simétricos, el factor de forma depende únicamente de la transferencia de momento $\boldsymbol{q}$. Este hecho si simboliza escribiendo el factor de forma como $F(\boldsymbol{q}^2)$.

Experimentalmente, la magnitud del factor de forma es determinada por la razón de la sección transversal experimental medida y la sección transversal de Mott:

$$\left(\frac{d\sigma}{d\Omega}\right)_{exp} = \left(\frac{d\sigma}{d\Omega}\right)^{*}_{Mott} \cdot |F(\boldsymbol{q}^2)|^2 \tag{4.1}$$

Se mide la sección transversal para una energía fija del haz variando el ángulo y se divide por la sección transversal de Mott antes calculada.

4.1 Factor de forma de los nucleones.

El factor de forma $F(\boldsymbol{q}^2)$, bajo ciertas condiciones, es la transformada de Fourier de la distribución de carga $f(\mathbf{x})$:

$$F(\boldsymbol{q}^2) = \iiint_{-\infty}^{\infty} e^{\frac{i\boldsymbol{q}\boldsymbol{x}}{\hbar}} f(\boldsymbol{x}) d^3x \tag{4.2}$$

Para casos simétricamente esféricos f depende solamente del radio $r = |\boldsymbol{x}|$. Integrando sobre todo el ángulo solido la integral queda:

$$F(\boldsymbol{q}^2) = 4\pi \int_0^{\infty} f(\boldsymbol{r}) \frac{sin|\boldsymbol{q}|r/\hbar}{|\boldsymbol{q}|r/\hbar} r^2 dr \tag{4.3}$$

Con la condición de normalización:

$$1 = \int f(\boldsymbol{x})\, d^3x = \int_0^{\infty} \int_{-1}^{+1} \int_0^{2\pi} f(r) r^2 d\phi dcos\vartheta dr = 4\pi \int_0^{\infty} f(r) r^2 dr \tag{4.4}$$

En principio, la distribución radial de carga podría ser determinada a través de la transformada inversa de Fourier:

$$f(r) = \frac{1}{(2\pi)^3} \iiint_{-\infty}^{\infty} F(\boldsymbol{q}^2) e^{-i\boldsymbol{q}\boldsymbol{r}/\hbar} d^3q \tag{4.5}$$

Sin embargo, experimentalmente, no se puede medir debido a que los limites van a infinito y no es posible calcular de manera exacta la distribución de carga. Para resolver esto, se proponen distribuciones arbitrarias y se comparan con los datos experimentales para ver cual se ajusta mejor.

La dispersión con un objeto con una superficie afilada resulta en un bien definido máximo y mínimo de difracción. Para una esfera homogénea de radio R, por ejemplo, un mínimo se encuentra en:

$$\frac{|\boldsymbol{q}| \cdot R}{\hbar} \approx 4.5 \tag{4.6}$$

La posición del mínimo muestra el tamaño del núcleo dispersado.

4.2 Radio de carga cuadrático medio

La información acerca del radio nuclear no solo puede ser obtenida de la posición del mínimo del factor de forma, sino que también de su comportamiento cuando $\boldsymbol{q}^2 \to 0$. Si la longitud de onda es considerablemente más grande que el radio nuclear R, entonces:

$$\frac{|\boldsymbol{q}| \cdot R}{\hbar} \ll 1 \tag{4.7}$$

Por lo tanto $F(\boldsymbol{q}^2)$ se puede expandir en potencias de $|\boldsymbol{q}|$:

$$\begin{aligned} F(\boldsymbol{q}^2) &= \int f(\boldsymbol{x}) \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{i|\boldsymbol{q}||\boldsymbol{x}| cos\vartheta}{\hbar} \right)^n d^3x \qquad\qquad con\ \vartheta = \measuredangle(\boldsymbol{x}, \boldsymbol{q}) \\ &= \int_0^{\infty} \int_{-1}^{+1} \int_0^{2\pi} f(r)[1 - \frac{1}{2} \left(\frac{|\boldsymbol{q}| r}{\hbar} \right)^2 \cos^2\vartheta + \cdots] d\phi dcos\vartheta r^2 dr \\ &= 4\pi \int_0^{\infty} f(r) r^2 dr - \frac{1}{6} \frac{\boldsymbol{q}^2}{\hbar^2} 4\pi \int_0^{\infty} f(r) r^4 dr + \cdots \end{aligned} \tag{4.8}$$

Se define al radio de carga cuadrado promedio como el primer momento de la distribución $f(r)$:

$$\langle r^2 \rangle = 4\pi \int_0^{\infty} r^2 \cdot f(r) r^2 dr \tag{4.9}$$

Usando la condición de normalización y la definición del radio de carga, el factor de forma se puede escribir como:

$$F(\boldsymbol{q}^2) = 1 - \frac{1}{6} \frac{\boldsymbol{q}^2 \langle r^2 \rangle}{\hbar^2} + \cdots \tag{4.10}$$

Entonces es necesario medir el factor de forma para valores pequeños de $\boldsymbol{q}^2$ para determinar $\langle r^2 \rangle$. Derivando la ecuación anterior queda:

$$\langle r^2 \rangle = -6\hbar \left. \frac{dF(\boldsymbol{q}^2)}{d\boldsymbol{q}^2} \right|_{\boldsymbol{q}^2 = 0} \tag{4.11}$$

4.3 Cinemática inversa.

El radio de carga de partículas estables puede ser medido a través del método tradicional de dispersión con un haz de electrones bombardeando el material a estudiar. Sin embargo, para partículas inestables, como es el caso de los hadrones inestables, este método no es lo suficientemente preciso debido al corto tiempo de vida de estas partículas. Para el estudio de partículas con tiempo de vida cortos se usa el procedimiento de cinemática inversa. Este consiste en que, en lugar de tener un blanco hecho del material a estudiar y bombardear con un haz de electrones, se usa un haz de estas partículas con momento muy grande y un blanco de electrones. El hecho de que el haz tenga momento muy grande implica utilizar cálculos relativistas, por lo que el factor de Lorentz en la dilatación temporal "aumenta" el tiempo de vida de las partículas vistas desde el marco de referencia del laboratorio.

El uso de energías relativistas implica la introducción de cuadri vectores en los cálculos y de cantidades invariantes bajo transformaciones de Lorentz.

4.4 Cinemática de la dispersión elástica hadrón – electrón.

En altas energías del haz del orden de GeV, el retroceso ya no puede ser despreciado, por lo que la sección transversal de Mott debe ser modificada tomando en cuenta la energía final e inicial del haz:

$$\left(\frac{d\sigma}{d\Omega}\right)_{Mott} = \left(\frac{d\sigma}{d\Omega}\right)^{*}_{Mott} \cdot \frac{E'}{E} \tag{4.12}$$

El cuadrado de la transferencia de cuadri momento es una cantidad invariante la cual se calcula como:

$$q^2 = (p - p')^2 \tag{4.13}$$

Si $p = \left(\frac{E}{c}, \boldsymbol{p}\right)$ y $p' = \left(\frac{E'}{c}, \boldsymbol{p}'\right)$ entones q^2 queda como:

$$q^2 = p^2 - 2pp' + p'^2 \tag{4.14}$$

De la relatividad especial se sabe que el cuadrado del cuadri momento es una cantidad invariante igual a la masa al cuadrado de la partícula, por lo que para el electrón se tiene que:

$$q^2 = 2m_e^2c^2 - 2\left(\frac{EE'}{c^2} - \boldsymbol{p}\cdot\boldsymbol{p}'\right)$$

$$q^2 = 2m_e^2c^2 - 2\left(\frac{EE'}{c^2} - |\boldsymbol{p}||\boldsymbol{p}'|cos\theta\right) \tag{4.15}$$

Para energías muy grandes se cumple que:

$$E \gg m_e^2c^2 \qquad y \qquad |\boldsymbol{p}| = \frac{E}{c} \tag{4.16}$$

Por lo tanto:

$$q^2 \approx -2\left(\frac{EE'}{c^2} - \frac{EE'}{c^2}cos\theta\right)$$

$$\approx -2\frac{EE'}{c^2}(1 - cos\theta)$$

$$q^2 \approx -4\frac{EE'}{c^2}\sin^2\frac{\theta}{2} \tag{4.17}$$

Con el objetivo de trabajar solo con cantidades positivas, se define:

$$Q^2 = -q^2 \tag{4.18}$$

El punto de inicio para los cálculos cinemáticos de la dispersión hadrón – electrón es la introducción de los cuadri vectores de ambas partículas. El cuadri momento del hadrón incidente de masa M es $p_M = \left(\frac{E_M}{c}, \boldsymbol{p_M}\right)$ y el cuadri momento del electrón de masa m inicialmente en reposo es $p_e = (mc, 0)$. De la conservación de momento se cumple que:

$$p_M + p_e = p'_M + p'_e \tag{4.19}$$

Elevando ambos lados al cuadrado;

$$p_M^2 + 2p_M p_e + p_e^2 = p_M'^2 + 2p'_M p'_e + p_e'^2 \tag{4.20}$$

Como el cuadri momento cuadrado es una cantidad invariante ($p_i^2 = p_i'^2$), queda:

$$p_M p_e = p'_M p'_e \tag{4.21}$$

Usando la ecuación (4.19), tenemos:

$$p_M p_e = p'_M(p_M + p_e - p'_M)$$

$$= p'_M p_M + p'_M p_e - p_M'^2$$

$$= \frac{E_M E'_M}{c^2} - \boldsymbol{p}_M \cdot \boldsymbol{p}'_M + E'_M m - M^2c^2$$

$$= \frac{E_M E'_M}{c^2} - |\boldsymbol{p}_M||\boldsymbol{p}'_M|cos\theta_M + E'_M m - M^2c^2 \tag{4.22}$$

El coseno del ángulo de dispersión del haz entonces es:

$$cos\theta_M = \frac{\frac{E_M E'_M}{c^2} + E'_M m - p_M p_e - M^2c^2}{|\boldsymbol{p}_M||\boldsymbol{p}'_M|}$$

$$= \frac{\frac{E'_M}{c^2}(E_M + mc^2) - E_M m - M^2c^2}{|\boldsymbol{p}_M||\boldsymbol{p}'_M|} \tag{4.23}$$

Para el cálculo del ángulo de dispersión del electrón el procedimiento es similar, de la ecuación (4.21) y (4.19)

$$\begin{aligned} p_M p_e &= (p_M + p_e - p'_e)p'_e \\ &= p_M p'_e + p_e p'_e - p'^2_e \\ &= \frac{E_M E'_e}{c^2} - \boldsymbol{p}_M \cdot \boldsymbol{p}'_e + mE'_e - m^2c^2 \\ &= \frac{E_M E'_e}{c^2} - |\boldsymbol{p}_M||\boldsymbol{p}'_e| cos\theta_e + mE'_e - m^2c^2 \end{aligned} \tag{4.24}$$

El ángulo de dispersión del electrón queda:

$$\begin{aligned} cos\theta_e &= \frac{\frac{E_M E'_e}{c^2} - p_M p_e + mE'_e - m^2c^2}{|\boldsymbol{p}_M||\boldsymbol{p}'_e|} \\ &= \frac{\frac{E'_e}{c^2}(E_M + mc^2) - E_M m - m^2c^2}{|\boldsymbol{p}_M||\boldsymbol{p}'_e|} \end{aligned} \tag{4.25}$$

La energía del centro de masa (la cual es una cantidad invariante) en el marco de referencia del laboratorio se calcula con el cuadrado de la suma de los cuadri momentos del haz y del electrón inicialmente en reposo.

$$\begin{aligned} s = (p_M + p_e)^2 = p_M^2 + 2p_M p_e + p_e^2 \\ = M^2c^2 + 2E_M m + m^2c^2 \approx M^2c^2 + 2E_M m \end{aligned} \tag{4.26}$$

Para determinar la máxima transferencia de momento posible, se calcula la energía del centro de masa en el marco de referencia del centro de masa. Para ello se usan los cuadri momentos del haz y del electrón, vistos desde este marco de referencia: $p_M = \left(\frac{E_M}{c}, \boldsymbol{p_M}\right)$ y $p_e = \left(\frac{E_e}{c}, \boldsymbol{p_e}\right)$.

$$\begin{aligned} s = (p_M + p_e)^2 = p_M^2 + 2p_M p_e + p_e^2 \\ = M^2c^2 + \frac{2E_M E_e}{c^2} - 2\boldsymbol{p_M} \cdot \boldsymbol{p_e} + m^2c^2 \end{aligned} \tag{4.27}$$

Usando $p^2 = \frac{E^2}{c^2} - |\boldsymbol{p}|^2 = M^2c^2$ y suponiendo que $\boldsymbol{p_M} = -\boldsymbol{p_e}$, la ecuación (4.27) queda:

$$\begin{aligned} s = M^2c^2 + 2\sqrt{M^2c^2 + |\boldsymbol{p_M}|^2}\sqrt{m^2c^2 + |\boldsymbol{p_e}|^2} + 2|\boldsymbol{p_M}|^2 + m^2c^2 \\ \approx M^2c^2 + 2\sqrt{M^2c^2 + |\boldsymbol{p_M}|^2}|\boldsymbol{p_e}| + 2|\boldsymbol{p_M}|^2 \end{aligned} \tag{4.28}$$

Para $|\boldsymbol{p_e}| = |\boldsymbol{p_M}|$:

$$s \approx M^2c^2 + 2|\boldsymbol{p_M}|\left(\sqrt{M^2c^2 + |\boldsymbol{p_M}|^2} + |\boldsymbol{p_M}|\right) \tag{4.29}$$

La transferencia de momento en este sistema de referencia es entonces:

$$\begin{aligned} -Q^2 = (p_e - p'_e)^2 = p_e^2 - 2p_e p'_e + p'^2_e \\ = -2p_e p'_e + 2m^2c^2 \end{aligned}$$

$$= -2\left(\frac{E_e E'_e}{c^2} - \boldsymbol{p}_e \cdot \boldsymbol{p}'_e\right) + 2m^2c^2$$

$$\approx -2(|\boldsymbol{p}_e||\boldsymbol{p}'_e| - |\boldsymbol{p}_e||\boldsymbol{p}'_e|cos\theta_e)$$

$$\approx -2|\boldsymbol{p}_e|^2(1 - cos\theta_e) = \approx -2|\boldsymbol{p}_M|^2(1 - cos\theta_e) \quad (4.30)$$

La máxima transferencia de energía se da cuando $\theta_e = \pi$

$$Q^2_{max} = 4|\boldsymbol{p}_M|^2 \quad (4.31)$$

Haciendo uso de la energía del centro de masa, ecuación (4.29):

$$s - M^2c^2 = 2|\boldsymbol{p_M}|\left(\sqrt{M^2c^2 + |\boldsymbol{p_M}|^2} + |\boldsymbol{p_M}|\right)$$

$$(s - M^2c^2)^2 = 4|\boldsymbol{p_M}|^2\left(\sqrt{M^2c^2 + |\boldsymbol{p_M}|^2} + |\boldsymbol{p_M}|\right)^2$$

$$(s - M^2c^2)^2 = 4|\boldsymbol{p_M}|^2\left(M^2c^2 + |\boldsymbol{p_M}|^2 + 2|\boldsymbol{p_M}|\sqrt{M^2c^2 + |\boldsymbol{p_M}|^2} + |\boldsymbol{p_M}|^2\right)$$

$$(s - M^2c^2)^2 = 4|\boldsymbol{p_M}|^2\left(M^2c^2 + 2|\boldsymbol{p_M}|\left(\sqrt{M^2c^2 + |\boldsymbol{p_M}|^2} + |\boldsymbol{p_M}|\right)\right)$$

$$(s - M^2c^2)^2 = 4|\boldsymbol{p_M}|^2 \cdot s$$

De (4.31) $$Q^2_{max} = \frac{(s - M^2c^2)^2}{s} \quad (4.32)$$

En el sistema de referencia del laboratorio $s = M^2c^2 + 2mE_M$, por lo que la ecuación (4.32) queda como:

$$Q^2_{max} = \frac{(2mE_M)^2}{M^2c^2 + 2mE_M} \quad (4.33)$$

La variable Q^2 esencial para los cálculos del radio de carga. Q^2 se puede determinar experimentalmente a través de cuatro métodos:

1. De la energía del electrón dispersado:

$$Q^2 = -(p_e - p'_e)^2$$

$$= -(-2p_e p'_e + 2m^2c^2)$$

$$= 2mE'_e - 2m^2c^2 \quad (4.34)$$

2. De la energía del haz antes y después de la dispersión:

$$Q^2 = -(p_e - p'_e)^2$$

$$= 2p_e p'_e - 2m^2c^2$$

$$= 2p_e(p_M + p_e - p'_M) - 2m^2c^2$$

$$= 2p_e p_M + 2p_e p_e - 2p_e p'_M - 2m^2c^2$$
$$= 2mE_M + 2m^2c^2 - 2mE'_M - 2m^2c^2$$
$$= 2m(E_M - E'_M) \tag{4.35}$$

3. Del ángulo de dispersión del electrón, partiendo de la ecuación (4.25):

$$cos\theta_e = \frac{\frac{E'_e}{c^2}(E_M + mc^2) - E_M m - m^2c^2}{|\boldsymbol{p}_M||\boldsymbol{p}'_e|}$$
$$2mc^2 cos\theta_e = \frac{2mE'_e(E_M + mc^2) - 2E_M m^2c^2 - 2m^3c^4}{|\boldsymbol{p}_M||\boldsymbol{p}'_e|}$$
$$2mc^2|\boldsymbol{p}_M||\boldsymbol{p}'_e|cos\theta_e = 2mE'_e(E_M + mc^2) - 2m^2c^2(E_M + mc^2)$$
$$\frac{2mc^2|\boldsymbol{p}_M||\boldsymbol{p}'_e|cos\theta_e}{E_M + mc^2} = 2mE'_e - 2m^2c^2$$

De (4.35)

$$Q^2 = \frac{2mc^2|\boldsymbol{p}_M||\boldsymbol{p}'_e|cos\theta_e}{E_M + mc^2} \tag{4.36}$$

4. Del ángulo de dispersión del haz, usando (4.23):

$$cos\theta_M = \frac{\frac{E'_M}{c^2}(E_M + mc^2) - E_M m - M^2c^2}{|\boldsymbol{p}_M||\boldsymbol{p}'_M|}$$
$$2mc^2|\boldsymbol{p}_M||\boldsymbol{p}'_M|cos\theta_M = 2mE'_M(E_M + mc^2) - 2E_M m^2c^2 - 2mM^2c^4$$
$$2mc^2|\boldsymbol{p}_M||\boldsymbol{p}'_M|cos\theta_M = 2mE'_M(E_M + mc^2) - 2E_M m^2c^2 - 2mc^2\left(\frac{E_M^2}{c^2} - |\boldsymbol{p_M}|^2\right)$$
$$2mc^2|\boldsymbol{p}_M||\boldsymbol{p}'_M|cos\theta_M = 2mE'_M(E_M + mc^2) - 2mE_M(mc^2 + E_M) + 2mc^2|\boldsymbol{p_M}|^2$$
$$\frac{2mc^2|\boldsymbol{p}_M||\boldsymbol{p}'_M|cos\theta_M - 2mc^2|\boldsymbol{p_M}|^2}{E_M + mc^2} = 2mE'_M - 2mE_M$$
$$2m(E_M - E'_M) = \frac{2mc^2|\boldsymbol{p_M}|^2 - 2mc^2|\boldsymbol{p}_M||\boldsymbol{p}'_M|cos\theta_M}{E_M + mc^2}$$

De (4.35):

$$Q^2 = \frac{2mc^2|\boldsymbol{p_M}| \cdot (|\boldsymbol{p}_M| - |\boldsymbol{p}'_M|cos\theta_M)}{E_M + mc^2} \tag{4.37}$$

Una vez redefinida la transferencia de momento al cuadrado $-q^2$ como Q^2, se puede describir ahora el radio de carga cuadrático promedio y los factores de forma en términos de esta nueva variable. La ecuación 4.11 queda como:

$$\langle r^2 \rangle = -6\hbar \frac{dF(Q^2)}{dQ^2}\bigg|_{Q^2=0} \tag{4.38}$$

La interacción de la carga de prueba con el momento magnético y el momento magnético anómalo del blanco conduce a una separación del factor de forma en una parte magnética $G_M(Q^2)$ y otra eléctrica $G_E(Q^2)$:

$$F^2(Q^2) = G^2(Q^2) + \frac{Q^4 G_M^2(Q^2)}{8m^2E_M^2 - 2Q^2(M^2c^2 + 2mE_m)} \tag{4.39}$$

Donde:

$$G^2(Q^2) = \frac{G_E^2(Q^2) + \frac{Q^2 G_M^2(Q^2)}{4M^2c^2}}{1 + \frac{Q^2}{4M^2c^2}} \tag{4.40}$$

La contribución de los momentos magnético y eléctrico, son tomadas en cuenta por la ecuación de Rosenbluth en su forma de invariante de Lorentz: (4.41)

$$\frac{d\sigma}{dQ^2} = \frac{2\pi\alpha^2\hbar^2}{Q^4} \frac{Q^2(Q^2 - 2m^2c^2)G_M^2(Q^2) + 2((s - m^2c^2 - M^2c^2)^2 - Q^2(s - m^2c^2))G^2(Q^2)}{(s - m^2c^2 - M^2c^2)^2 - 4m^2M^2c^4}$$

Usando la definición de la energía del centro de masa s. (ec. (4.26)), la ecuación de Rosenbluth toma la forma:

$$\frac{d\sigma}{dQ^2} = \frac{2\pi\alpha^2\hbar^2}{Q^4} \frac{Q^2(Q^2 - 2m^2c^2)G_M^2(Q^2) + 2((2mE_M)^2 - Q^2(M^2c^2 + 2mE_M))G^2(Q^2)}{(2mE_M)^2 - 4m^2M^2c^4}$$

Como $E_M \gg mc^2$:

$$\frac{d\sigma}{dQ^2} = \frac{2\pi\alpha^2\hbar^2}{Q^4} \frac{Q^4 G_M^2(Q^2) + 2((2mE)^2 - Q^2(M^2c^2 + 2mE_M))G^2(Q^2)}{(2mE_M)^2}$$

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4} \left\{ \frac{Q^4 G_M^2(Q^2)}{2(2mE_M)^2} + \frac{((2mE)^2 - Q^2(M^2c^2 + 2mE_M))G^2(Q^2)}{(2mE_M)^2} \right\}$$

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4} \left\{ \frac{Q^4 G_M^2(Q^2)}{2(2mE_M)^2} + \left(1 - Q^2 \frac{(M^2c^2 + 2mE_M)}{(2mE_M)^2}\right) G^2(Q^2) \right\}$$

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4} \left\{ \frac{Q^4 G_M^2(Q^2)}{2(2mE_M)^2} + \left(1 - \frac{Q^2}{Q_{max}^2}\right) G^2(Q^2) \right\}$$

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4} \left(1 - \frac{Q^2}{Q_{max}^2}\right) \left\{ \frac{Q^4 G_M^2(Q^2)}{2(2mE_M)^2 \left(1 - \frac{Q^2}{Q_{max}^2}\right)} + G^2(Q^2) \right\}$$

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4}\left(1 - \frac{Q^2}{Q_{max}^2}\right)\left\{\frac{Q^4 G_M^2(Q^2)}{2(2mE_M)^2\left(1 - \frac{Q^2(M^2c^2 + 2mE_M)}{(2mE_M)^2}\right)} + G^2(Q^2)\right\}$$

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4}\left(1 - \frac{Q^2}{Q_{max}^2}\right)\left\{\frac{Q^4 G_M^2(Q^2)}{8m^2E_M^2 - 2Q^2(M^2c^2 + 2mE_M)} + G^2(Q^2)\right\}$$

De (4.39):

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4}\left(1 - \frac{Q^2}{Q_{max}^2}\right)F^2(Q^2) \tag{4.42}$$

Capítulo 5:

Dispersión Kaón – electrón.

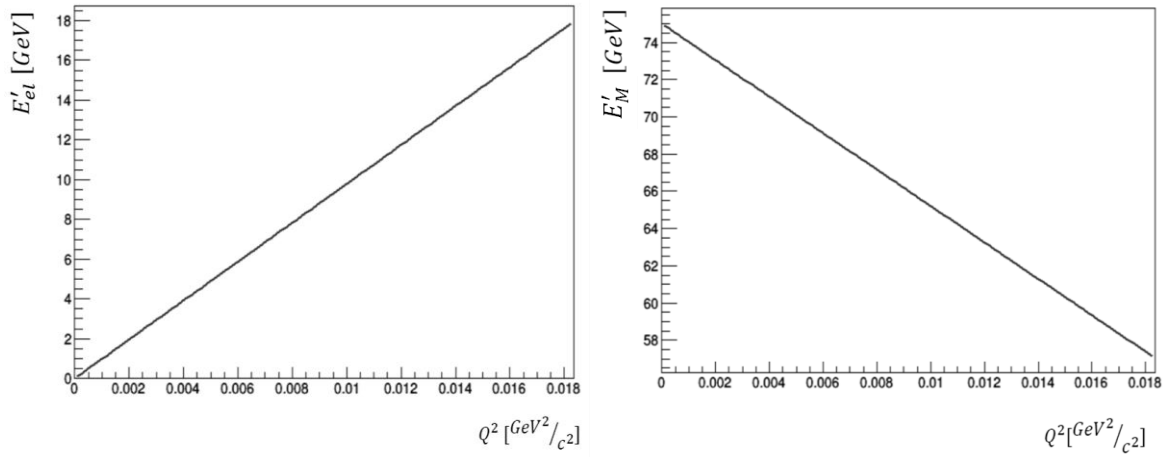
El kaón es un hadrón perteneciente al grupo de los mesones, ya que está compuesto de una combinación de dos quarks. El kaón cargado positivamente K^+ está formado por un quark up y un anti strange ($u\bar{s}$), su masa es de 493.66 MeV. El kaón cargado negativamente K^- está formado por un quark anti-up y un strange ($\bar{u}s$), su masa es idéntica a la del K^+. El kaón neutro K^0 está formado por un quark down y un anti strange ($d\bar{s}$), con una masa de 497.64 MeV. Y el $\overline{K^0}$ está formado por un quark anti down y un strange ($\bar{d}s$) con una masa idéntica al K^0.

Este trabajo está enfocado en el estudio del radio de carga del K^+ en el experimento NA62 en CERN, para el cual se hizo incidir un haz de esta partícula (masa M) con energía de $E_M = 75$ GeV en un blanco de electrones, m = 0.511 MeV. Los detalles sobre el haz, el blanco y la distribución de los detectores que se usaron, son expuestos en el siguiente capitulo. El estudio cinemático del capítulo anterior será ahora aplicado a estas dos partículas. La ecuación (4.33) describe la máxima transferencia de momento, la cual para estas partículas es:

$$Q^2_{max} = \frac{(2mE_M)^2}{M^2c^2 + 2mE_M}$$

$$= 0.01833\ GeV^2/c^2 \tag{5.1}$$

Las distribuciones esperadas para las variables experimentales se muestran en la figura 5.1 y 5.2:



.

Figura 5.1.: Energías esperadas del kaón y el electrón después de la dispersión en función de Q^2 (Ecuaciones 4.34 y 4.35). Energía del electrón (izquierda) comienza en cero (ignorando m_e) ya que estaba inicialmente en reposo. Energía del kaón (derecha) comienza en 75 GeV que es la energía del haz.

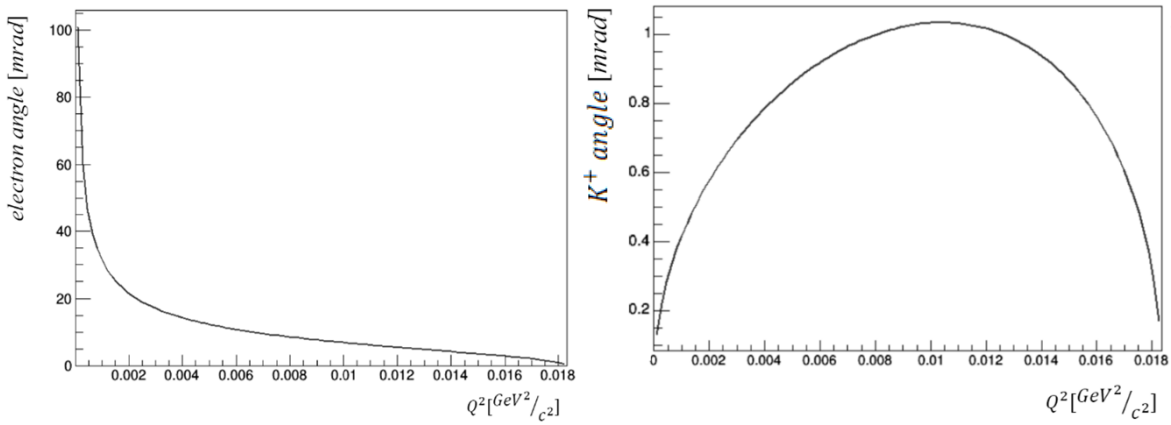


Figura 5.2. Distribuciones esperadas de los ángulos de dispersión del electrón (izquierda) y del kaón (derecha) en función de Q^2 (Ecuaciones 4.36 y 4.37). Es observable el límite de transferencia de momento $Q^2_{max} = 0.01833\ GeV^2/c^2$

Para sistemas compuestos de dos quarks, como el caso del K^+, se asume un ajuste monopolar para el factor de forma eléctrico mientras que el factor de forma magnético desaparece, ya que es un mesón.

$$G_E(Q^2) = G_E(0) \cdot \left(1 + \frac{Q^2}{\Lambda^2}\right)^{-1} \approx G_E(0) \cdot \left(1 - \frac{Q^2}{\Lambda^2}\right), \qquad \text{Con } 1 \gg \frac{Q^2}{\Lambda^2} \tag{5.2}$$

Donde el parámetro Λ^2 será definido a continuación.

Por lo que el factor de forma para el kaón, ecuación 4.39, queda:

$$F^2(Q^2) = \frac{G_E^2(Q^2)}{1 + \frac{Q^2}{4M^2c^2}} = \frac{\left[G_E(0) \cdot \left(1 + \frac{Q^2}{\Lambda^2}\right)^{-1}\right]^2}{1 + \frac{Q^2}{4M^2c^2}} \tag{5.3}$$

Redefiniendo el radio de carga cuadrado promedio como:

$$\langle r^2 \rangle = -6\hbar \frac{dG_E(Q^2)}{dQ^2}\bigg|_{Q^2=0} \tag{5.4}$$

Derivando la ecuación (5.2) y comparando con la ecuación (5.4), llagamos a la relación para el parámetro Λ^2:

$$\Lambda^2 = \frac{6\hbar^2}{\langle r^2 \rangle} \tag{5.5}$$

Reuniendo todo esto, la ecuación (4.42) se puede escribir como:

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4}\left(1 - \frac{Q^2}{Q^2_{max}}\right)\left(1 + \frac{\langle r^2 \rangle Q^2}{6\hbar^2}\right)^{-2}\left(1 + \frac{Q^2}{4M^2c^2}\right)^{-1} \tag{5.6}$$

La figura 5.3 muestra la sección trasversal diferencial del kaón para diferentes valores arbitrarios del radio de carga, ($\langle r^2 \rangle = 0.25\ fm^2, 0.34\ fm^2\ y\ 0.45\ fm^2$) además del caso de una partícula puntual.

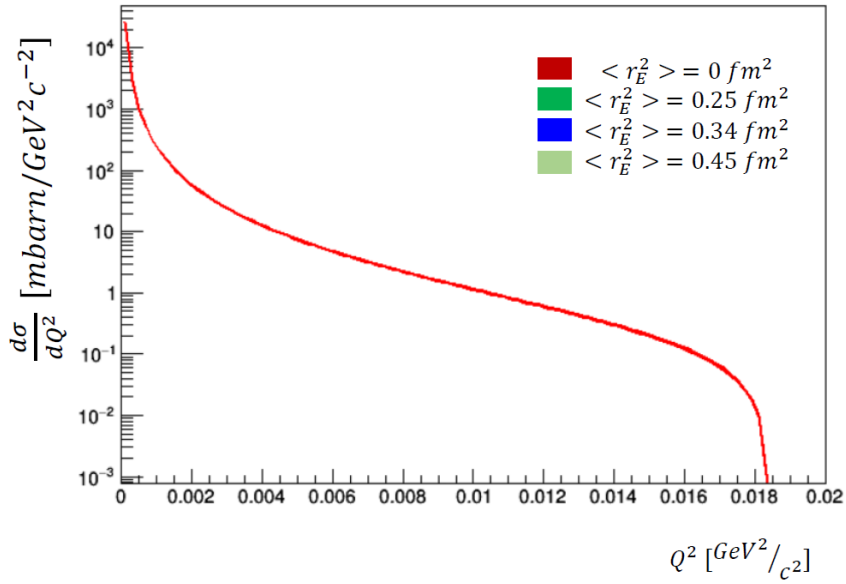


Figura 5.3.: Comparación de la sección transversal del kaón para diferentes valores arbitrarios del radio de carga. La grafica de los cuatro valores es idéntica.

De la ecuación (5.4), se observa que para obtener el valor del radio de carga cuadrado promedio, necesitamos conocer el valor de la pendiente del factor de forma para valores pequeños de Q^2. La figura 5.4 muestra el factor de forma eléctrico $G_E(Q^2)$ (ecuación (5.2) para diferentes valores arbitrarios de $\langle r^2 \rangle$.

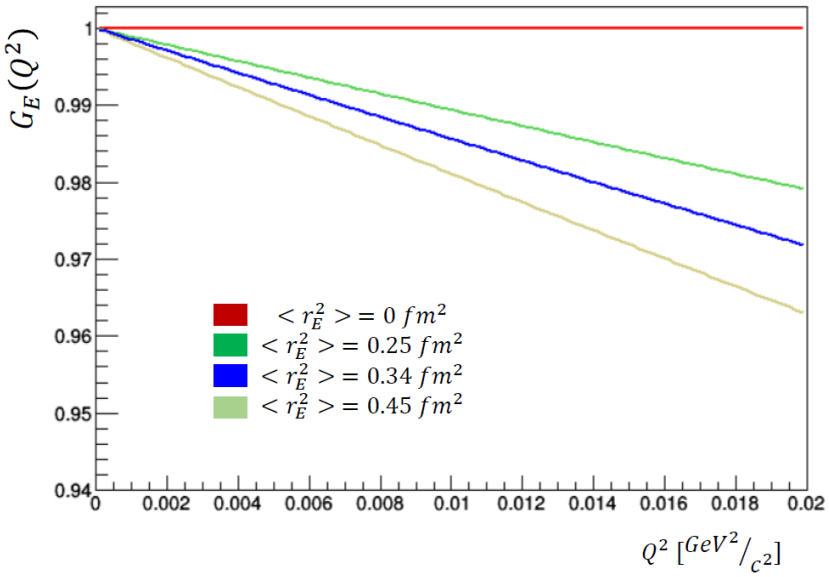


Figura 5.4.: Factor de forma eléctrico de kaón para diferentes valores de $\langle r^2 \rangle$, así como el caso de una partícula puntal.

5.1 Dispersión Pion-electrón.

Todo el estudio cinemático anterior puede ser aplicado también a otra de las partículas presentes en el haz del experimento NA62, el pion. El pion es otro de los hadrones que pertenecen al grupo de los mesones, compuestos de dos quarks. El pion cargado positivamente π^+ se compone de un quark up y un anti-down $(u\bar{d})$, el pion negativo π^- está compuesto por un quark down y un anti-up $(\bar{u}d)$ ambos con una masa de 139.57 MeV, mientras que el pion neutro π^0 compuesto de un quark up y anti-up $(u\bar{u})$ y su antipartícula $\overline{\pi^0}$ compuesta por un quark down y anti-down $(d\bar{d})$ tienen una masa de 134.97 MeV.

Aplicando todas las ecuaciones anteriores a la interacción elástica pion-electrón, obtenemos las distribuciones esperadas de Q^2 de la misma forma que con el Kaon. Para esta interacción tenemos un nuevo límite para Q^2_{max}, ecuación (4.33):

$$Q^2_{max} = \frac{(2mE_M)^2}{M^2c^2 + 2mE_M}$$
$$= 0.06193\ GeV^2/c^2 \tag{5.7}$$

A continuación, se muestran las distribuciones de las cuatro formas de obtener Q^2, ecuaciones (4.34-37).

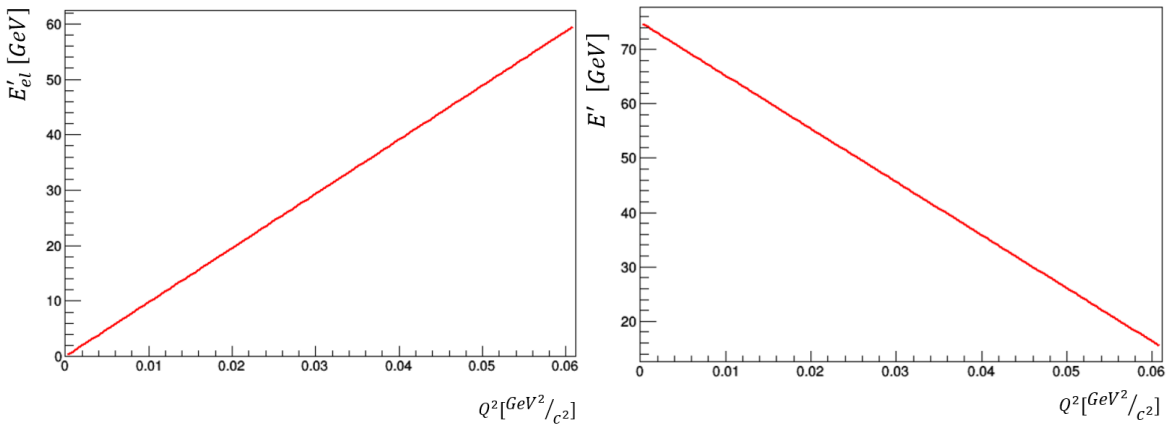


Figura 5.5.: Energías esperadas del pion y el electrón después de la dispersión en función de Q^2 (Ecuaciones 4.34 y 4.35). Energía del electrón (izquierda) comienza en cero (ignorando m_e) ya que estaba inicialmente en reposo. Energía del pion (derecha) comienza en 75 GeV que es la energía del haz.

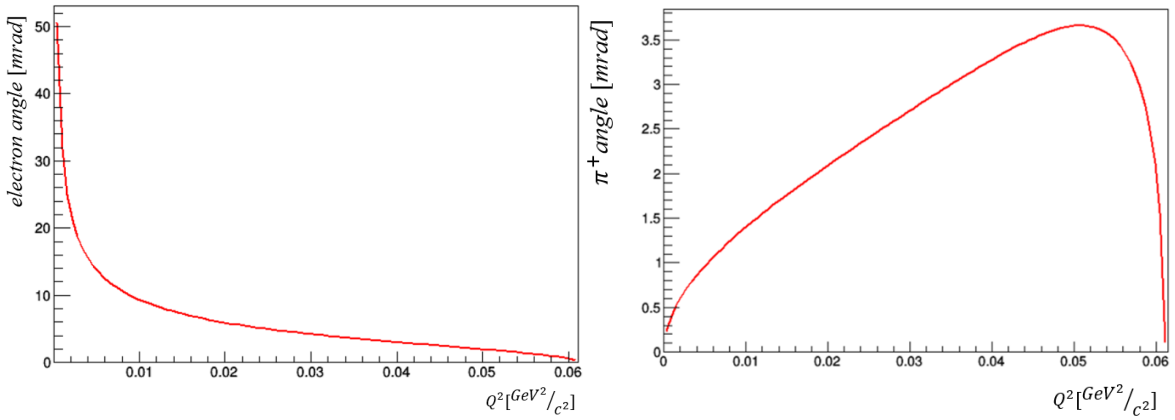


Figura 5.6. Distribuciones esperadas de los ángulos de dispersión del electrón (izquierda) y del pion (derecha) en función de Q^2 (Ecuaciones 4.36 y 4.37). Es observable el límite de transferencia de momento $Q^2_{max} = 0.06193\ GeV^2/c^2$

El pion, al ser un sistema de dos quarks, el mismo ajuste monopolar se puede aplicar para el factor de forma eléctrico y, por supuesto, el factor de forma magnético desaparece al ser un mesón igual que el kaón. Por lo que la ecuación de la sección transversal diferencial (ecuación 5.6) queda de la misma forma para el pion.

$$\frac{d\sigma}{dQ^2} = \frac{4\pi\alpha^2\hbar^2}{Q^4}\left(1 - \frac{Q^2}{Q^2_{max}}\right)\left(1 + \frac{\langle r^2\rangle Q^2}{6\hbar^2}\right)^{-2}\left(1 + \frac{Q^2}{4M^2c^2}\right)^{-1} \tag{5.8}$$

La figura 5.7 muestra como seria la sección transversal diferencial del pion para diversos valores arbitrarios de $\langle r^2\rangle$ (0.30 fm^2, 0.44 fm^2 y 0.57 fm^2), así como también el caso de una partícula puntual.

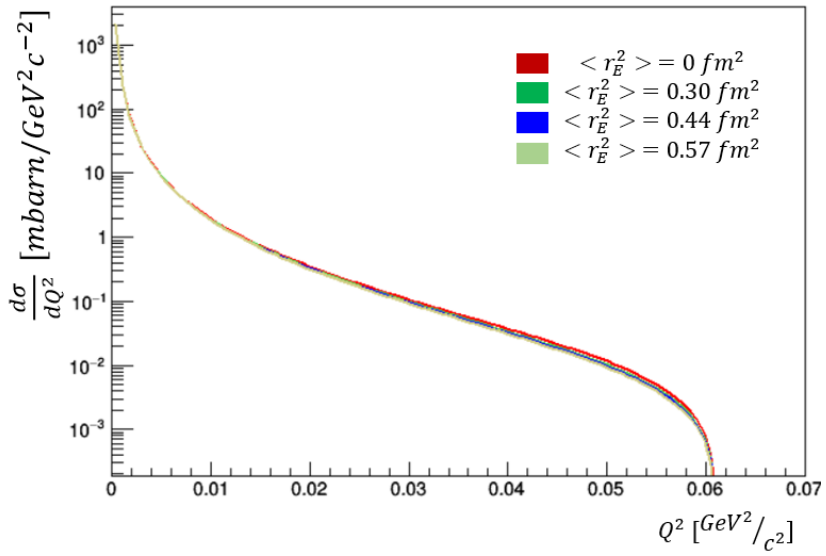


Figura 5.7.: Comparación de la sección transversal del pion para diferentes valores arbitrarios del radio de carga

5.2 Dispersión protón-electrón.

Otra de las partículas presentes en el haz del experimento NA62 es el protón. El protón, a diferencia del kaón y del pion, está constituido por tres quarks, dos quarks up y un quarks down (uud). Con una masa de 938.27 MeV se puede estudiar a si mismo la dinámica de la dispersión con electrones.

El límite de transferencia de momento cuadrado para el protón se calcula de la misma manera con la ecuación 4.33:

$$Q^2_{max} = \frac{(2mE_M)^2}{M^2c^2 + 2mE_M}$$

$$= 0.006319\ GeV^2/c^2$$

La siguiente figura muestra como serían las distribuciones angulares del protón y del electron en función de Q^2.

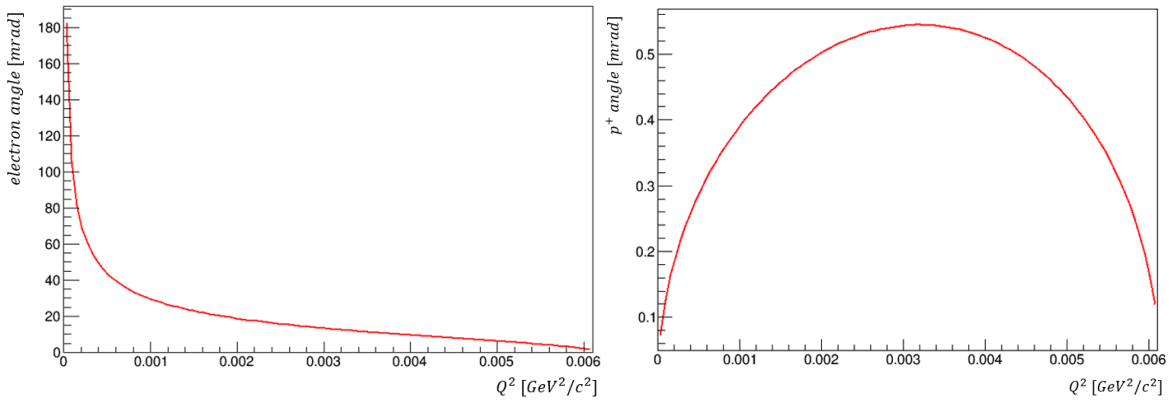


Figura 5.7. Distribuciones esperadas de los ángulos de dispersión del electrón (izquierda) y del protón (derecha) en función de Q^2 (Ecuaciones 4.36 y 4.37). Es observable el límite de transferencia de momento $Q^2_{max} = 0.006193\ GeV^2/c^2$

Capítulo 6:

Experimento NA62.

En este capítulo se hablará sobre el experimento NA62, de su objetico principal y de los detectores con los cuales está conformado, así como su distribución. Este experimento es importante para mi trabajo ya que cuenta con los detectores necesarios para realizar un análisis de dispersión. Cuenta con un haz el cual contiene un porcentaje de kaones y se usan los electrones presentes en uno de los detectores (GTK3) como blanco para la dispersión. Cuenta también con detectores especializados para la medición de ángulos y energías, así como otros diseñados para la identificación de productos. Estos últimos son de los más importantes ya que ayudaran a filtrar que partículas provienen de un decaimiento y cuáles de un proceso de dispersión.

El experimento NA62 está ubicado en el área del norte de las instalaciones del CERN, en un experimento de blanco fijo el cual su principal objetivo es realizar mediciones de decaimientos raros de kaones. Una de estas mediciones es la probabilidad de que los kaones cargados (K^+) se desintegren en un pion cargado, un neutrino y un antineutrino ($K^+ \to \pi^+ \nu\bar{\nu}$). El haz de protones con momento de 400 GeV/c proveniente del acelerador SPS del CERN permite la producción de un haz secundario de kaones con momento de 75GeV/c. La ventaja de usar un haz de protones muy energético es la reducción de la información no relacionada al kaón debido a la alta sección eficaz de la producción de kaones. La desventaja de tener protones muy energéticos y, consecuentemente, un haz secundario muy energético, es que los piones y protones no pueden ser separados eficientemente de los kaones. La razón de partículas del haz de hadrones secundario del NA62 es 750 MHz, del cual alrededor del 6% son K^+, dejando 5 MHz de decaimientos del K^+ en los 65 m de largo de la región de decaimiento. Por lo tanto, los detectores que miden el momento y dirección de los kaones son expuestos a un flujo de partículas alrededor de 16 veces más grande que el flujo de los kaones. Notemos que el 75% de los kaones no decaen antes de llegar al final del experimento.

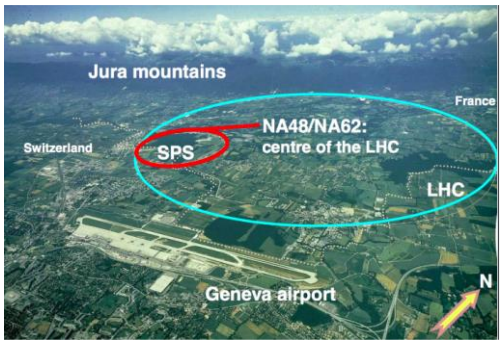


Figura 6.1: Ubicación del experimento NA62 en el CERN.

6.1 Distribución de los detectores.

La escala y el sistema de referencia para la distribución experimental se muestran la figura 6.2. La línea de haz define al eje Z con su origen en el blanco que produce a los kaones y las partículas de haz viajan en la dirección positiva, el eje Y apunta verticalmente hacia arriba, y el eje X es horizontal y direccionado con un sistema coordenado de mano derecha.

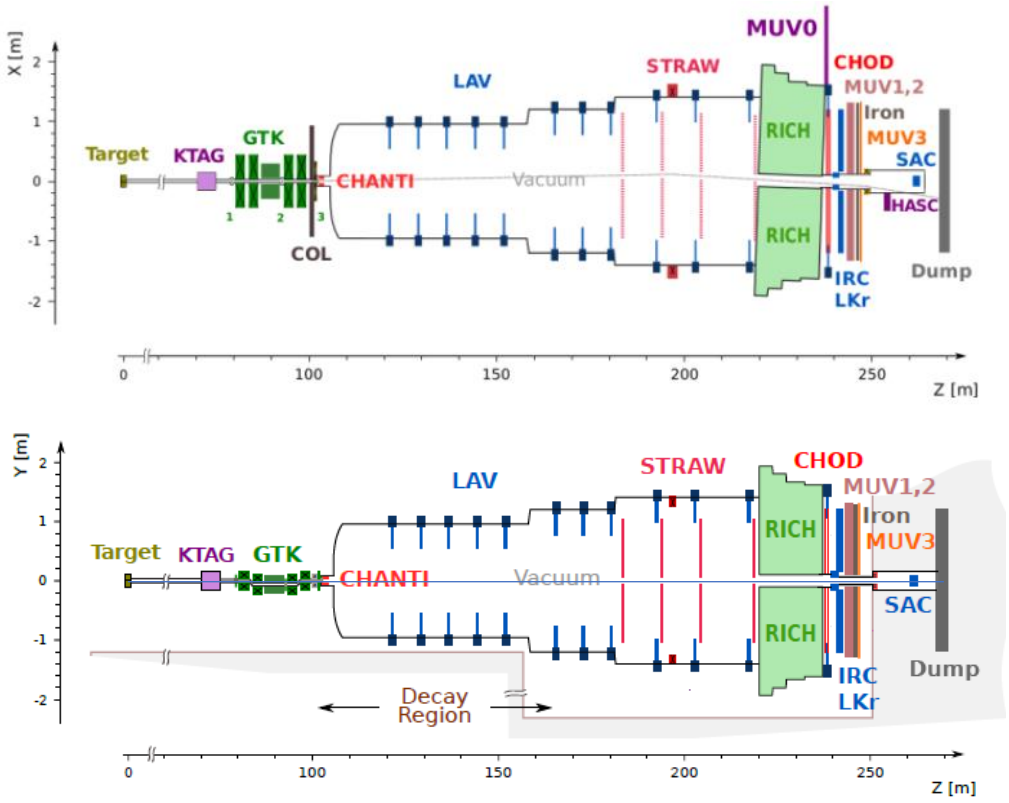


Figura 6.2. Esquema vertical (arriba) y horizontal (abajo) de la distribución de los detectores y las zonas que componen al experimento, así como la trayectoria del haz.

El espectrómetro GTK y todos los detectores alrededor de la zona de decaimiento, así como el espectrómetro que detecta los estados finales de las partículas son puestos al vacío para evitar interacciones y dispersiones de haz y para obtener una resolución mejorada para las cantidades cinemáticas medidas. La identificación de los kaones es proporcionada por un contador diferencial Cherenkov CEDAR equipada con un sistema de detección de fotones KTAG. El espectrómetro del haz GTK consiste en tres estaciones de silicio proporcionando momento y dirección de los kaones entrantes. Un sistema de seguimiento para masas pequeñas es esencial para minimizar dispersión inelástica de partículas de haz en el material del detector que podrían imitar una partícula aislada proveniente de un decaimiento. El detector de anillo de protección CHANTI, instalado en seguida del GTK, detecta dispersión inelástica en la última estación.

Después de la región de decaimiento, el rastreador STRAW mide las trayectorias y el momento de los productos cargados de los decaimientos del K^+. Un sistema de detectores photon-veto proporciona una cobertura hermética para fotones producidos en la zona de decaimiento y propagándose en ángulos arriba de 50 mrad con respecto al eje de los detectores. El detector de imágenes y anillos Cherenkov RICH situado en seguida de la última cámara del STRAW incluye un

radiador de 17 m de largo llenado con neón a presión de una atmosfera y asegura la separación de electrones, muones, piones y kaones. El detector RICH es seguido por un sistema de contadores de hodoscopio CHOD construidos con placas y tejas de centelles. Los contadores proporcionan una resolución en tiempo de alrededor de 150 ps el cual es suficientemente precisa para definir el tiempo de activación para los otros detectores. Dos calorímetros hadrónicos (MUV1, MUV2) y un plano de placas centellantes (MUV3) detrás de 80 cm de hierro, forman un sistema de identificación pion/muon.

6.2 Detectores.

6.2.1 Haz

El haz primario de protones es extraído a 400 GeV/c del acelerador SPS del CERN y es dirigido por la línea del haz P42 al blanco T10 (cilindro de berilio de 400 mm de longitud y 2 mm de diámetro) localizado en el túnel que conecta el SPS a la sala experimental subterránea. El haz impacta en el blanco T10 en pulsos de 3 segundos con intensidades que varían en un rango de 1.9 a $2.2x10^{12}$ protones por pulso. Un diseño de "línea", une el blanco T10 al centro del calorímetro LKr existente, se ha adoptado para el haz secundario cargado positivamente. Este haz secundario de alta intensidad tiene hasta 750 millones de partículas por segundo, es llamado K12 es derivado del blanco T10 a un momento central de 75 GeV/c y está constituido en un 70% de piones, 23% de protones y un 6% de kaones. La línea del haz K12 tiene una longitud de 101.3 metros.

EL blanco T10 es seguido por un colimador de cobre con longitud de 950 mm refrigerado por agua que ofrece una selección de orificios de diferentes aperturas: un orificio de 15 mm de diámetro es generalmente seleccionado para transmitir las partículas secundarias deseadas. Los primeros elementos activos del haz de alta intensidad consisten en un triplete de cuadrupolos magnéticos (Q1, Q2, Q3), los cuales recolectan una aceptancia de ángulo solido ($\pm$2.7 mrad horizontalmente, $\pm$1.5 mrad verticalmente) a un momento central de 75 GeV/c. Posteriormente sigue un acromático frontal (A1) que consiste en cuatro dipolos magnéticos de deflexión vertical. Los primeros dos dipolos magnéticos producen un desplazamiento paralelo del haz por 110 mm, mientras que los siguientes dos dipolos regresan el haz a su eje original. Entre ellos, el haz pasa a través de un conjunto de agujeros graduados en dos unidades motorizadas beam-dump refrigeradas con agua, TAX1 Y TAX2, con la finalidad de hacer la selección del momento mientras absorbe el haz primario de protones restante y las partículas secundarias no deseadas. Considerando un enfoque horizontal y vertical entre TAX1 y TAX2, un radiador que consiste en un arreglo de placas de tungsteno con un grosor de hasta 5 mm se introduce en el haz. Esta optimizado para causar que los positrones pierdan suficiente energía por Bremsstrahlung para ser posteriormente rechazados, minimizando al mismo tiempo la perdida de hadrones por dispersión.

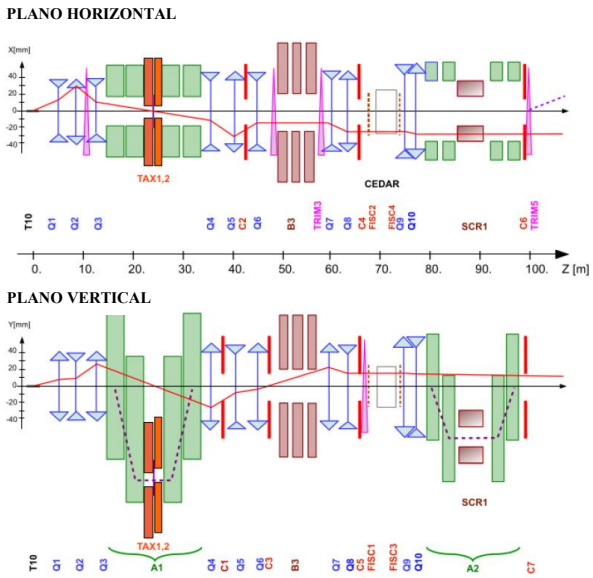


Figura 6.3 Diseño esquemático y óptico del haz secundario de K+ desde el blanco T10 hasta la entrada a la región de decaimiento. En cada vista, la línea solida corresponde a la trayectoria de una partícula que sale del blanco desde el centro a un momento nominal y a un ángulo indicado. La línea discontinua indica la trayectoria de una partícula con momento de 75 GeV/c inicialmente sobre el eje.

Un triplete de cuadrupolos (Q3, Q5, Q6) sirven para volver a enfocar el haz en el plano vertical y hacerlo paralelo con un ancho limitado en el plano horizontal. El espacio entre los cuadrupolos es ocupado por dos colimadores (C1, C2), los cuales redefinen la aceptancia vertical y horizontal del has transmitido. Un colimador subsecuente (C3) redefine el haz a un segundo foco en el plano vertical. En este punto los positrones que han sido degradados en momento por el radiador entre TAX1 y TAX2 están suficientemente separados por el haz de hadrones para que el colimador C3 los absorba. Después el haz pasa a través de un diámetro de 40 mm donde casi no hay campo, en placas de acero las cuales están insertadas entre los polos de tres dipolos magnéticos (B3) con longitud de 2 m. El campo magnético vertical en el acero que está ubicado alrededor del haz sirve para remover a los muones de ambos signos, mientras que la desviación del haz debido al pequeño campo de dispersión dentro del orificio se cancela mediante dos dipolos direccionales (TRIM 2 y TRIM 3 antes y después de B3). Un contador diferencial de Cherenkov (CEDAR) equipado con 8 arreglos de fotodetectores (KTAG) sirve para detectar a los K^+ en el haz. Esto requiere que el haz se vuelva paralelo, para lo cual el CEDAR esta seguido por dos cuadrupolos (Q7, Q8), así como también por dos colimadores horizontal y vertical (C4, C5) para que absorban las partículas que se encuentran al final del haz. Dos pares (vertical y horizontal) de filamentos de contadores de centelles (FISC 1, 3 y FISC 2,4) están instalados antes y después del CEDAR. Cuando se conectan en coincidencia respectivamente, permiten medir la divergencia promedio del haz y ajustarla a cero y verificar la divergencia intrínseca restante en cada plano. Después del CEDAR se encuentra un doblete de relativamente débiles cuadrupolos (Q9, Q10), que unen el haz a través de la etapa de seguimiento y medición del momento y determina la divergencia y el tamaño del haz a través de las aberturas de los detectores siguientes.

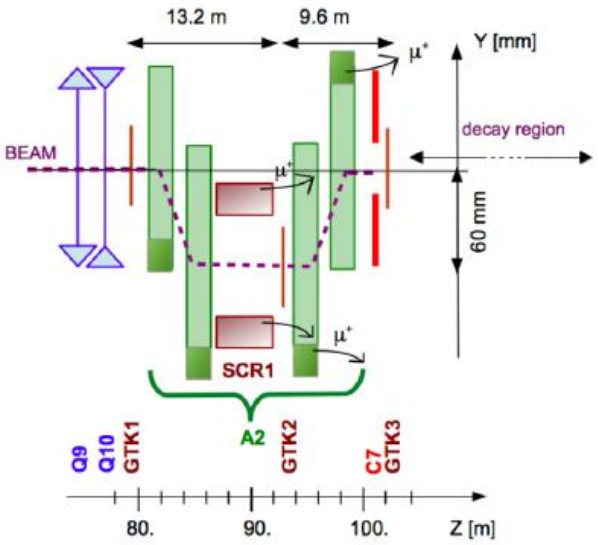


Figura 6.4 Diseño esquemático del seguimiento del haz y la medición del momento en el segundo arreglo acromático (A2). El haz es deflactado verticalmente 60 mm y regresa a su dirección nominal después de la medición del momento. Los muones son apartados por el raspador SCR1 y las abrazaderas de retorno de los dos dipolos últimos en el arreglo acromático (áreas que están más oscuras).

El sistema de seguimiento del haz GTK consiste en tres estaciones, las estaciones están arregladas de forma que el espacio entre GTK1 y GTK3 sea ocupado por un arreglo acromático (A2), compuesto por cuatro dipolos magnéticos. Las abrazaderas de retorno del tercer y cuarto dipolo, al igual que el colimador de acero magnetizado toroidalmente (SCR1), deflactan los muones que deja el haz en la sección de dispersión del momento entre el segundo y tercer dipolo de este arreglo (6.4). El GTK2 está localizado en la misma sección, justo después del colimador magnético SCR1, el GTK3 está localizado a 102.4 m del blanco T10, marca el plano de entrada al comienzo de la región de decaimiento. Los colimadores de limpieza (C6, C7) que preceden el GTk3 están destinados a interceptar el ruido fuero de la aceptancia del haz.

Adicionalmente, un imán de dirección horizontal (TRIM 5) es usado para deflactar el haz al eje X positivo por un ángulo de +1.2 mrad. Este ángulo es ajustado de manera que una deflexión subsecuente de -3.6 mrad hacia la dirección negativa del eje X, debido al imán del espectrómetro MNp33, que dirige al haz a través de la apertura central del calorímetro LKr y sus detectores subsecuentes (6.5). La región de decaimiento está ubicada en los primeros 60 m de un tanque de 117 m de longitud, comenzando a 102.4 m posterior al blanco de berilio. El tanque es evacuado a una presión residual de $\sim 10^{-6}$ mbar usando hasta siete bombas criogénicas. El tanque aloja 11 detectores LAV y cuatro cámaras de los espectrómetros STRAW y consiste en 19 secciones cilíndricas hechos de acero o acero inoxidable. El diámetro del contenedor incrementa desde 1.92m en la primera sección después del GTK3 a 2.4 m en la sección media y hasta a 2.8 m en la región del espectrómetro. El espectrómetro magnético incluye dos pares de cámaras STRAW de seguimiento, a cada uno de los lados del dipolo magnético de gran apertura (MNP33). El dipolo magnético provee un golpe horizontal en momento de 270MeV/c deflactando el haz de 75GeV/c por -3.6 mrad, para converger y posteriormente cruzar el eje sin desviar en un punto a 2.8 m posterior al centro del calorímetro LKr.

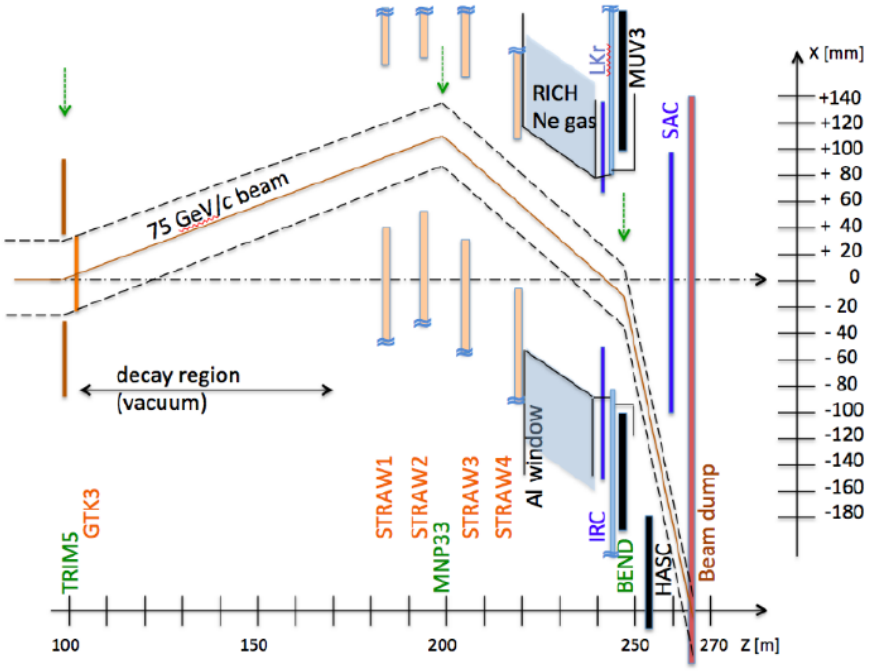


Figura 6.5 Vista más cerca de la línea del haz a través de la región de decaimiento y los detectores en el plano (X, Z). Las flechas verticales indican el centro de flexión de cada imán.

Cercano a este punto de cruce, un par de filamentos contadores de centelleo (FISC5, 6), instalados en vacío, permiten observar al haz y dirigirlo correctamente. El haz es finalmente deflactado hacia la dirección negativa del eje X a un ángulo de -13.2 mrad por un dipolo magnético (BEND). El haz es finalmente absorbido en un beam-dump que está compuesto de acero rodeado de concreto a una distancia suficiente detrás del detector para disminuir los efectos de salpicadura hacia atrás. Para monitorear el perfil e intensidad del haz, una cámara de cables con lectura analógica y una cámara de ionización están localizadas en el espacio entre la ventana de salida del vacío y el beam-dump.

6.2.2 Kaón Tagger (KTAG)

Los Kaones son el 6% del haz secundario K12 y son identificados por el detector KTAG donde la luz Cherenkov se produce en el volumen gaseoso del radiador de un CEDAR tipo W del CERN. Usa gas N2 a 1.75 bar de presión dentro de un contenedor de 5 m de largo, la presión es elegida de manera que la luz proveniente del tipo de partícula deseada pase a través de un diafragma anular de radio central fijo y apertura radial variable. La luz es enfocada a la salida del contenedor a través de ocho ventanas de cuarzo y enfocadas en 8 espejos esféricos. Los espejos reflejan la luz radialmente hacia afuera en ocho cajas de luz, la entrada de cada una de estas cajas en una guía de la luz consistente de una matriz de 64 secciones cónicas cercanamente espaciadas de 15 mm de radio externo y 4 mm de radio interno cortadas en secciones esféricas de placas de aluminio de 17 mm de grosor con un centro de curvatura en el foco virtual de la luz Cherenkov. El detector KTAG, incluye detección de fotones y sistemas para leer los datos, fue desarrollado para cumplir estos requerimientos, la óptica de este detector y su mecánica desarrollada para el experimento se muestran en la figura 6.6.

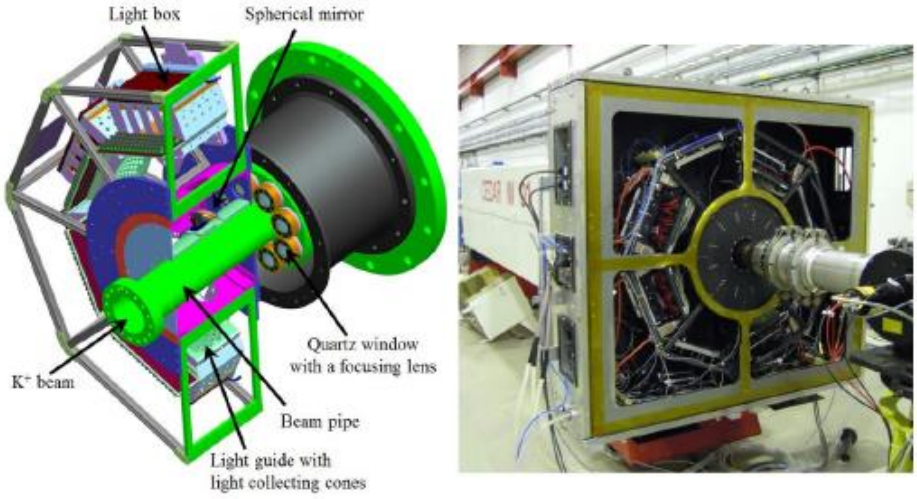


Figura 6.6 Izquierda: Dibujo de la parte anterior del CEDAR y KTAG. Derecha: Detectores KTAG y CEDAR en la línea del haz con cuatro de los 8 sectores equipados.

6.2.3 Espectrómetro de haz (GTK)

El espectrómetro del haz provee mediciones precisas del momento, tiempo y dirección del haz entrante de partículas. El espectrómetro está compuesto de tres estaciones similares instaladas alrededor del arreglo de cuatro dipolos magnéticos dispuestos de manera acromática (ver figura 6.7). El momento de la partícula puede derivarse del desplazamiento vertical de la trayectoria en la segunda estación. EL GTK está diseñado para medir el momento del haz de partículas de 75 GeV/c a una precisión de 0.2% y sus direcciones, dX/dZ y dY/dZ, a la salida del arreglo acromático a una precisión de 16 μrad. El alto flujo del haz requiere una resolución en el tiempo mejor que 200 ps. Cada estación es un detector híbrido de silicón que consiste en 18 000 pixeles de $300x300\ \mu m^2$ de área cada uno, arreglados en una matriz de 200 x 90 elementos que corresponde a un área total de 62.8 x 27 mm^2.

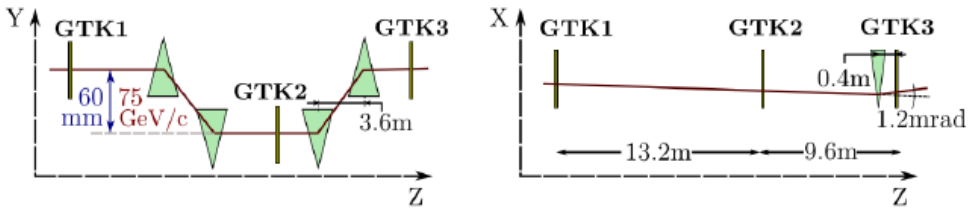


Figura 6.7 Esquema de las estaciones del GTK dentro del acromático en las vistas horizontal y vertical

6.2.4 Charged anti-coincidence detector (CHANTI)

El detector CHANTI rechaza el ruido que proviene de las interacciones inelásticas del haz con la última estación del GTK (GTK3). Detecta el halo de los muones cercanos al haz y una fracción de las partículas cargadas que son generadas antes del GTK3. El CHANTI está compuesto de seis estaciones cuadradas, la primera estación se encuentra 28 mm después del GTK3 y la distancia entre

una estación y la siguiente es aproximadamente el doble para las estaciones sucesivas, de forma que la región angular entre 49 mrad y 1.34 mrad está cubierta herméticamente por partículas generadas en el GTK3.

Figura 6.8: Derecha: imagen de una estación ensamblada en el lado del sensor. Izquierda: La estación ensamblada del lado del enfriamiento.

El GTK3 y todas las estaciones del CHANTI están ubicados dentro del mismo recipiente de vacío. Las estaciones están hechas de 48 varas centelleantes, cada una tiene dos planos de lectura, con las varas orientadas vertical y horizontalmente para formar las vistas del eje X y Y.

6.2.5 Espectrómetro STRAW

El espectrómetro STRAW mide las trayectorias y el momento de las partículas producidas en los decaimientos del kaón. Se extiende sobre una longitud de 35 m a lo largo de la línea del haz, comenzando desde 20m después de la región de decaimiento. Consiste en cuatro cámaras y un dipolo magnético de gran apertura (MNP33) el cual provee un campo de 0.9 Tm. Para minimizar la dispersión múltiple de las cámaras están construidas de material ligero y son instaladas dentro del tanque de vacío. El diseño del módulo esta optimizado para minimizar la dispersión múltiple y para dar una resolución espacial uniforme en el área activa. Cada una de estas cámaras STAW contiene dos módulos que a su vez contienen dos vistas de medición X (0°), Y (90°) y el otro modulo contiene las vistas de U (-45°) y V (+45°) (figura 6.9 izquierda). El área activa de la cámara es un círculo de 2.1 m de radio externo centrado en el eje Z longitudinal. Cada vista tiene un espacio de casi 12 cm sin tubos cerca del centro, tal que, después de superponer las cuatro vistas, un hueco en forma de octágono de 6 cm de apotema es creado para el paso del haz. Como el haz tiene un ángulo de +1.2 mrad y -3.6 mrad en el plano horizontal, arriba y abajo del imán. respectivamente (figura 6.5), este hueco no está centrado en el eje Z, pero tiene compensaciones sobre la dirección X en cada cámara. Una alta detección es proporcionada a través de un arreglo de 4 capas de tubos por vista, lo que nos garantiza al menos dos hits por vista, esto es de 8 a 12 hits por cada cámara (figura 6.9 derecha). Debido al espacio de 12 cm sin Straw en cada vista, el número de hits por cámara no se distribuyen uniformemente sobre la superficie del detector.

Cada cámara contiene 1792 straws (tubos) de 9.82 mm de diámetro y 2160 mm de longitud. El gas dentro de los straws es una mezcla de 70% de Ar y 30% de CO_2 a presión atmosférica. Los tubos son operados en el vacío del tanque de decaimiento. Están suficientemente separados uno del otro para permitir que los tubos flexibles incrementen su diámetro cuando el tanque de decaimiento esta al vacío.

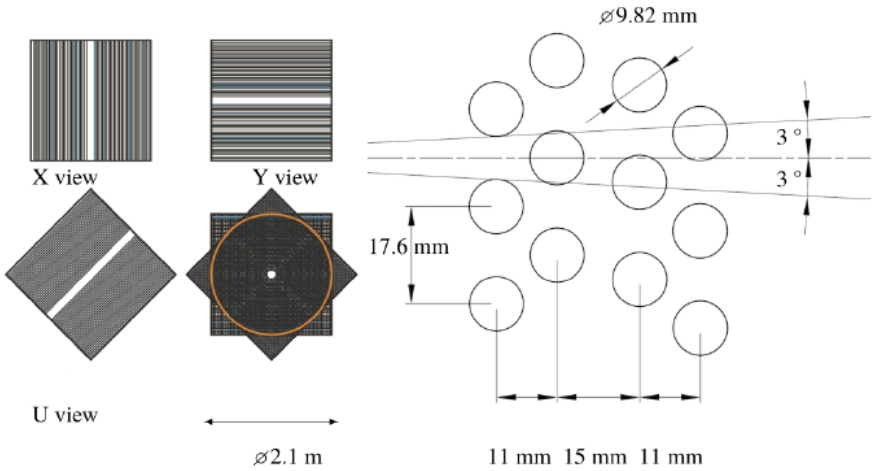


Figura 6.9: Izquierda: Una cámara STRAW está compuesta de cuatro vistas (X, Y, U, V) y cada vista mide una coordenada. Derecha: La geometría está basada en dos capas dobles de tubos por cada vista.

Cada uno de estos tubos está hecho de tereftalato de polietileno (PET), cubierto con 50 nm de cobre y 20 nm de oro en el interior. Los alambres del ánodo de tungsteno revestidos en oro tienen 30 μm de diámetro y están tensos a 80 g sin soportes y enrollados alrededor de ambas terminales de los tubos. Son diseñados para resistir la fuerza de una diferencia de presión de $\sim$ 1 bar, en caso de que uno de estos tubos se rompa. Cuando un tubo se rompe, el sistema de gas detecta una pérdida o caída de presión y cierra tanto la entrada como la salida en unos pocos milisegundos.

6.2.6 Large-angle veto system (LAV)

Estos detectores tienen forma de anillos y se encuentran en 11 posiciones dentro del volumen del vacío, mientras que la doceava estación se encuentra 3 m antes del calorímetro LKr. Los detectores LAV proveen una cobertura geométrica para los fotones de decaimientos dentro del volumen de decaimiento emitidos a ángulos de 8.5 a 50 mrad con respecto al eje Z.

Calorímetro de Kriptón liquido (LKr)

El LKr es un calorímetro cuasi homogéneo lleno con aproximadamente 9000 litros de kriptón liquido a 120 K dentro de criostato. Es usado para la identificación de partículas y detecciones de fotones con cobertura angular de 1 a 8.5 mrad. Esta segmentado en dirección transversal en 13248 celdas, donde cada celda tiene una sección eficaz de 2x2 cm^2. Las celdas están formadas por electrodos de CU-Be alineados a lo largo del eje longitudinal del experimento, tienen forma de zigzag para evitar insuficiencias cuando una lluvia de partículas está muy cerca del ánodo. La señal que se produce cuando cruzan partículas por el LKr es recolectada por preamplificadores dentro del criostato, directamente adheridos a las tiras calorimétricas. La señal se envía a las placas del transceptor a través de cables coaxiales de 50 Ω.

Small angle veto system (SAV)

Provee hermeticidad para fotones emitidos a ángulos hasta los cero grados con respecto al eje Z. Consiste en dos detectores: el IRC y el SAC. Ambos son calorímetros Shashlyk, con plomo y placas centelladoras de plástico atravesadas por fibras de cambio de longitud de onda (WLS). Los fotones que provienen del decaimiento de los kaones en el volumen de decaimiento que atraviesan los detectores SAV tienen energías mayores a 5 GeV. Para ambos detectores, el flujo de fotones que se esperan sea del orden de 1 MHZ a la intensidad nominal del haz. EL IRC esta adicionalmente expuesto a los muones que provienen de los decaimientos de las partículas del haz, están concentrados en un punto de pocos cm de diámetro a un lado (hacia la dirección X negativa) de la línea del haz; el flujo de muones de este punto incrementa el flujo de partículas en el IRC a 10 MHz.

Small angle calorimeter (SAC)

Consiste en 70 placas de plomo y 70 placas de plástico centellador, ambas con dimensión transversal de 205x205 mm^2 y un grosor de 1.5 mm. Este detector está instalado dentro del vacío del haz hacia el final de la pipa del haz. Para asegurarse que los fotones incidentes en el SAC a lo largo del eje Z no atraviese el detector, el SAC está alineado a un ángulo de 2.3 mrad con respecto al eje Z en el plano horizontal.

Intermediate-ring calorimeter (IRC)

El detector IRC es un calorímetro de plomo/centellador en forma de un cilindro excéntrico rodeando al haz antes que LKr. El detector tiene un diámetro externo de 290 mm y está centrado en el eje z. El agujero central tiene un diámetro de 120 mm con un offset de 12 mm hacia la dirección X positiva para tomar en cuenta la deflexión del haz debido al espectrómetro magnético. Está dividido en dos módulos longitudinales, ambos con módulos anterior y posterior con medidas de 89 y 154 mm de profundidad, respectivamente. Los módulos se encuentran espaciados a 40 mm. El diámetro interior del módulo siguiente es de 2.2 mm más grande que el módulo anterior, así los fotones producidos en el decaimiento de los kaones en el volumen de decaimiento no peguen en el borde de abajo del IRC, escapando así de la detección.

6.2.7 Ring Imaging Cherenkov Counter (RICH)

Este detector se diseñó como un espectrómetro de velocidad, el cual sirve para medir la velocidad y el ángulo de una partícula. Esto se puede interpretar como que separa a los piones de los muones en un momento de entre 15 y 35 GeV/c, mide el tiempo de cruce del pion con una resolución de 100 ps. Es un contenedor de 17.5 m de largo, con forma cilíndrica de acero ferro perlítico (ver figura 6.10) lleno de gas neón. El recipiente consiste en cuatro secciones de diámetro decreciente y de diferentes longitudes. En el extremo de arriba, el contenedor tiene un ancho de alrededor de 4.2 m para acomodar los bordes de los fotomultiplicadores fuera del área activa del detector. El diámetro de la última sección del contenedor es de 3.2 m, el cual es suficiente para albergar los espejos y su panel de soporte.

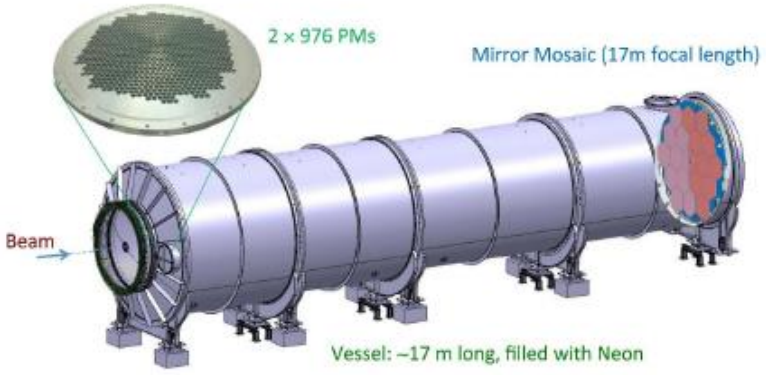


Figura 6.10: Vista esquemática del detector RICH: el haz de hadrones entra por la izquierda y viaja a lo largo del detector. Un acercamiento en uno de los arreglos de los fotomultiplicadores es mostrado en la parte superior izquierda; el mosaico de espejos es visible a través del contenedor en el lado derecho.

El área activa del detector se extiende a una distancia radial de 1.1 m desde el eje del haz a la entrada del RICH y a 1.4 m de la ventana de salida. La ventana de entrada y salida tienen una forma cónica y están hechas de aluminio con un grosor de 2 a 4 mm respectivamente. La ventada de entrada es el único separador entre el volumen de decaimiento y el RICH. un ligero tubo de aluminio, conecta el tanque de decaimiento que se encuentra al vacío, pasa por el centro del contenedor. Un mosaico de 20 espejos esféricos es usado para reflectar el cono de luz Cherenkov sobre el arreglo de los sensores de luz en el plano focal del espejo. para evitar la absorción de la luz reflectada por el haz, los espejos están divididos en 2 superficies esféricas: una con el centro de la curvatura a la izquierda y otra a la derecha del haz. Los espejos tienen un radio nominal de curvatura de 34m y por lo tanto una longitud focal de 17 m. El mosaico incluye 18 espejos de una forma hexagonal regular de 350 mm de lado y dos medios espejos. Los dos últimos se utilizan en el centro y tienen una abertura circular para acomodar el haz.

6.2.8 Charged Particle Hodoscopes.

Consiste en un sistema detector de centelleo llamado hodoscopio de partículas cargadas. Cubren la aceptancia lateral posterior al RICH y anterior del calorímetro LKR definido por el detector LAV12 con un radio interno de 1070 mm y el detector IRC con radio externo de 145 mm. Su función principal es proveer una entrada para el activador L0 cuando al menos una partícula cargada cruce la región anular con las dimensiones arriba definidas. Se encuentran expuestos a un flujo de partículas cargadas de 13 MHz. El sistema consiste en detector NA48-CHOD del experimento anterior NA48 y el detector CHOD optimizado para condiciones de alta intensidad. Ambos detectores se localizan respectivamente anterior y posterior al detector LAV12, a aproximadamente 700 mm de separación en la dirección longitudinal. El detector NA48-CHOD se basa en las coincidencias de señales en dos planos de bloques de centelleo horizontal y vertical. El detector CHOD consta de un solo plano de mosaicos de centelleo y una configuración de mosaicos más fina en el área de alta ocupación cercano al eje del haz.

6.2.9 Hadron Calorimeter (MUV1, MUV2)

El calorímetro hadrónico está hecho de capas alternantes de acero y centelladores correspondiendo alrededor de 8 longitudes de interacción. El calorímetro está dividido en dos detectores independientes: el detector frontal (MUV1) que tiene una segmentación fina transversal para una mejor separación de los componentes hadrónicos y los componentes de los chorros electromagnéticos y el detector trasero (MUV2).

6.2.10 Fast Muon Veto (MUV3)

El detector MUV3, se localiza posterior al calorímetro hadrónico detrás de una pared de acero de 80 cm de grosor y es usado para la identificación de muones. Tiene un área transversal de 2640x2640 mm^2 y esta construido de mosaicos de centelladores de 50 mm de grosor, incluyendo 140 mosaicos regulares de 220x220 mm^2 de dimensión transversa y 8 mosaicos más pequeños adyacentes al haz.

6.2.11 Peripheral Muon Veto (MUV0)

El MUV0 es un hodoscopio centellador diseñado para detectar π^- emitidos en el decaimiento $K^+ \to \pi^+\pi^+\pi^-$ con momento por debajo de 10 GeV, desviado hacia el eje X positivo por el imán del espectrómetro, abandonando la aceptancia lateral cerca del RICH ya que el MUV0 está montado en la brida posterior del RICH.

6.2.12 Hadronic Sampling Calorimeter (HASC)

El detector HASC es usado para la detección de π^+ emitidos en decaimientos $K^+ \to \pi^+\pi^+\pi^-$ con momento por arriba de los 50 GeV/c y propagándose a través de los orificios de las vigas en los centros de las cámaras STRAW. El detector está localizado subsecuente al MUV3 y al dipolo magnético BEND que barre estos piones fuera del haz de K^+ hacia la dirección negativa del eje X. Esta construido de 9 módulos idénticos. El elemento activo de un módulo es un sándwich de 60 placas de plomo de 16 mm de grosor intercaladas con 60 placas de centellador de 4 mm de ancho de 100 x 100 mm^2 en la dimensión transversal.

6.3 Trigger and Data Acquisition System. (TDAQ)

El intenso flujo del experimento NA62 dicta la necesidad de un sistema de trigger y adquisición de alto rendimiento, en el que debe minimizar el tiempo muerto mientras que maximiza la fiabilidad la colección de datos. Un sistema unificado de trigger y adquisición de data (TDAQ) fue diseñado para abordar dichos requerimientos en el experimento NA62 de una manera simple y rentable. Con un flujo estimado de decaimientos de 10MHz en el detector, el flujo máximo de salida en el L0 hardware trigger fue elegido para ser 1 MHz.

6.3.1 L0 Hardware Trigger

El L0 hardware trigger es el encargado de filtrar eventos basados en las entradas de un pequeño grupo de detectores y tienen una salida de flujo máximo de 1 MHz. Los detectores que se encuentran en este grupo son:

- CHOD: provee primitivas positivas para cualquier trayectoria cargada basado en multiplicidad de hits y tiempo de referencia.
- RICH: provee primitivas positivas para cualquier trayectoria cargada por encima del umbral de Cherenkov, basado en multiplicidad de hits.
- LAV: provee primitivas de veto de fotones (y muon halo), basado en multiplicidades de bloques adyacentes de hits.
- MUV3: provee primitivas de muon basada en multiplicidades de bloques, utilizado tanto en positivo como en veto de la lógica.
- Calorímetros (Calorímetro electromagnético LKr, así como para los calorímetros hadrónicos MUV1, MUV2), provee identificación positiva para los piones basado en depósitos de energía y primitivas de veto basado en la multiplicidad de clúster.

Las primitivas se generan de forma asincrónica en un tiempo variable, actualmente no superior de 100 μs. Cada primitiva consiste en un bloque de datos de 64-bit.

L0 trigger processor (L0TP)

La función principal del L0TP es adquirir las primitivas del trigger, organizarlas en tiempo y buscar coincidencias alineadas en tiempo con cualquiera de las máscaras trigger activas. La alineación temporal se basa en información contenida en los datos primitivos, con una marca especifica en el tiempo de 25 ns y con una precisión de 100 ps en un tiempo fino.

6.3.2 High Level Triggers (HLT)

El flujo máximo del trigger L0 es 1 MHz. Se requiere una reducción significativa del flujo y de los datos posteriores para que coincida con el ancho de banda disponible para el almacenamiento permanente de datos. El sistema TDAQ del experimento NA62 usa dos niveles de triggers basados en software para lograr la reducción necesaria:

- L1 trigger: reduce el flujo de datos por un factor de 10 a un máximo de 100 kHz, con algoritmos que utilizan información independiente de subdetectores individuales. Los calorímetros (LKr, MUV1 y MUV2) no se pueden usar para la decisión del L1 ya que solo se leen después de una decisión positiva del L1 trigger.
- L2 trigger: reduce el flujo de datos por otro factor ~ 10, hasta el almacenamiento permitido de un orden de 10 kHz. EL filtro de eventos L2 se basa en una reconstrucción parcial y explota información correlacionada de varios subdetectores.

6.4 Medición de la dispersión kaón-electrón en el experimento NA62.

En los capítulos 4 y 5 se abordó el procedimiento experimental para la obtención del radio de carga de una partícula, el cual consiste en hacer incidir un haz de la partícula a estudiar en un blanco de electrones para posteriormente analizar las distribuciones de las cantidades cinemáticas de los productos. Para la obtención de radio de carga del kaón, el experimento NA62 usa el haz secundario proveniente del SPS y utiliza los electrones de la tercera estación del espectrómetro GTK (GTK3) como blanco para la dispersión.

Como se observó en la sección 6.2.5, los detectores encargados de medir las trayectorias y los momentos de las partículas son los STRAW. Para que una partícula pueda ser medida con precisión, esta debe de atravesar por lo menos los dos primeras cámaras de STRAW, para posteriormente ser desviados por el imán MNP33 y puedan llegar al RICH y al calorímetro LKr, donde se hace la identificación de partículas, así como la energía que llevan en este punto (Figura 6.5).

La figura 6.11 muestra el rango del ángulo de dispersión para el cual se asegura un mínimo de dos hits en el primer par de STRAWs. Las partículas dispersadas con ángulos menores o mayores a este rango no son detectadas en su totalidad por todo el conjunto de detectores, por lo cual su información no nos será de utilidad.

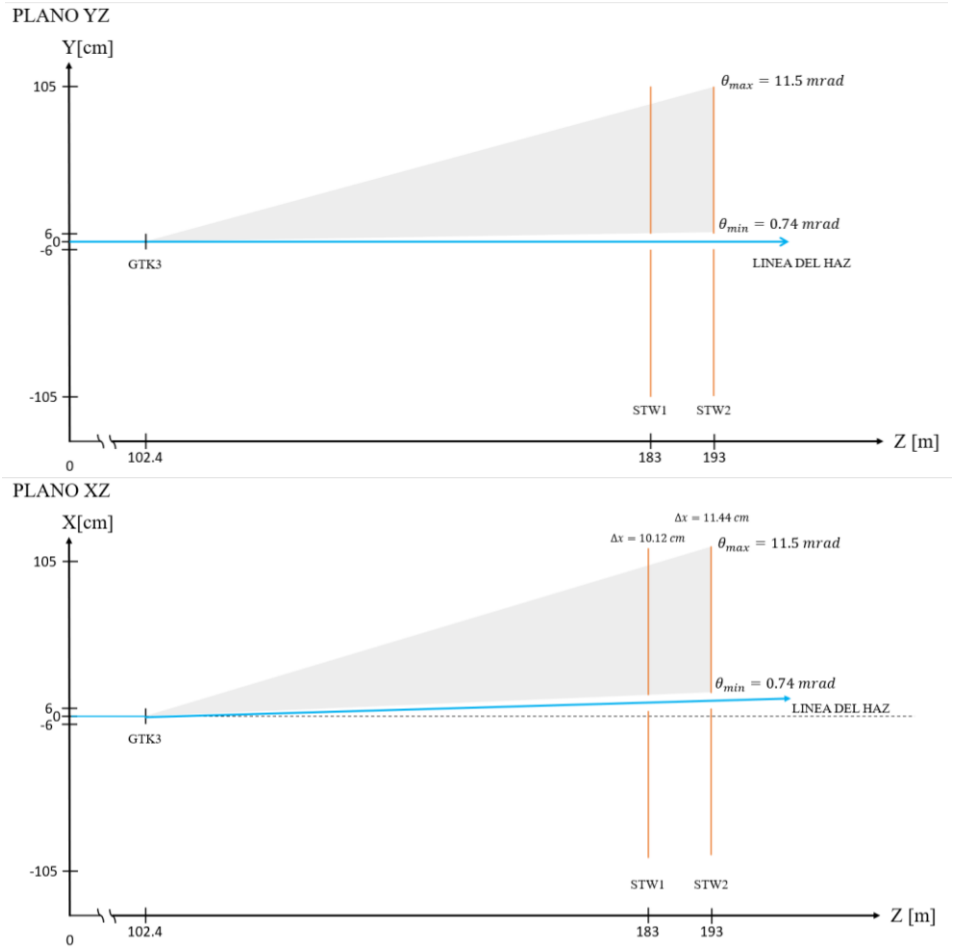


Figura 6.11 Rangos del ángulo de dispersión que aseguran el paso por el primer par de STRAW. El ángulo medido en el plano XZ (abajo) es respecto la línea del haz, no respecto al eje Z. En este plano el haz es desviado 1.2 mrad en la dirección de X positiva respecto al eje Z.

Con esta restricción del ángulo de dispersión, la obtención de Q^2 se verá también restringida por este intervalo (ec. 4.36 y 4.37). Las siguientes figuras muestran las distribuciones de los ángulos en función de la transferencia de momento para el kaón, pion y el protón, acotadas por el ángulo mínimo y máximo necesarios para una buena medición.

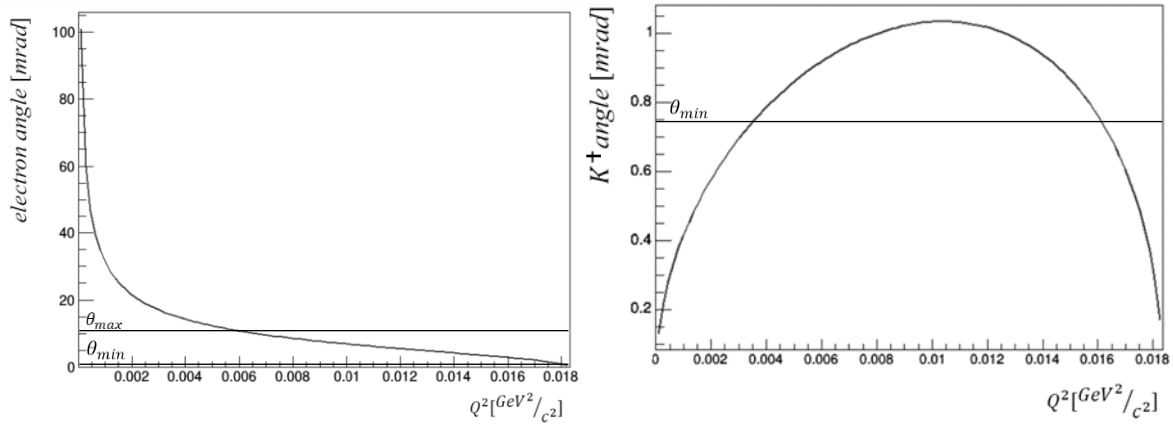


Figura 6.12. Distribuciones esperadas para la dispersión Kaon-electron acotadas por el valor mínimo y máximo del ángulo que puede ser medido por los detectores. Se observa que solo una pequeña porción de los kaones dispersados es detectada.

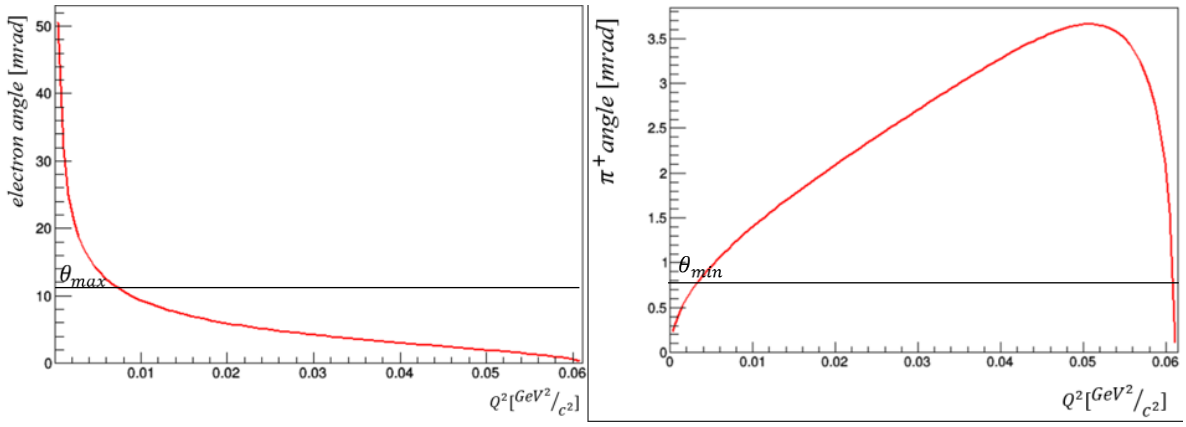


Figura 6.13. Distribuciones esperadas para la dispersión pion-electron acotadas por el valor mínimo y máximo del ángulo que puede ser medido por los detectores. A diferencia del Kaon, una mayor parte de los piones dispersados si llegan a ser detectados por los espectrómetros STRAW.

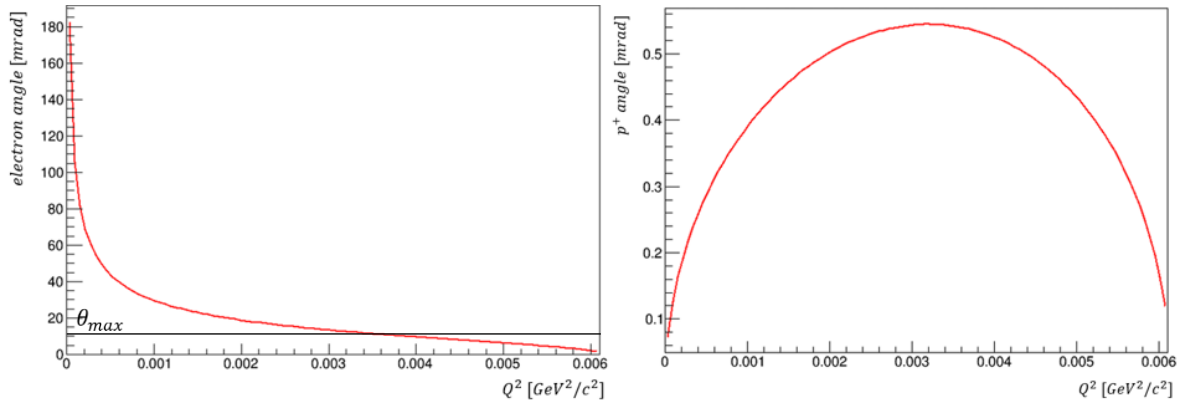


Figura 6.14. Distribuciones esperadas para la dispersión protón-electron acotadas por el valor mínimo y máximo del ángulo que puede ser medido por los detectores. El ángulo de dispersión del protón es tan pequeño que no alcanza a pasar por ningún STRAW para su detección.

En las figuras anteriores podemos observar que, si solo usamos los STRAW como detector principal para el ángulo de dispersión, solo abarcamos una pequeña fracción de eventos, además de que necesitamos valores pequeños de Q^2 y en la mayoría de los casos los eventos que podríamos detectar tienen una Q^2 alejada del origen. Como alternativa se propone el uso de los LAV para ampliar el rango de ángulo disponible para detección (vea sección 6.2.6). Con el uso de los LAV aumentamos el limite superior en el ángulo de dispersión hasta 50 mrad. A continuación, se muestran nuevamente las distribuciones de los ángulos de dispersión, pero ahora acotados con el nuevo rango de medición proporcionado por los LAVs.

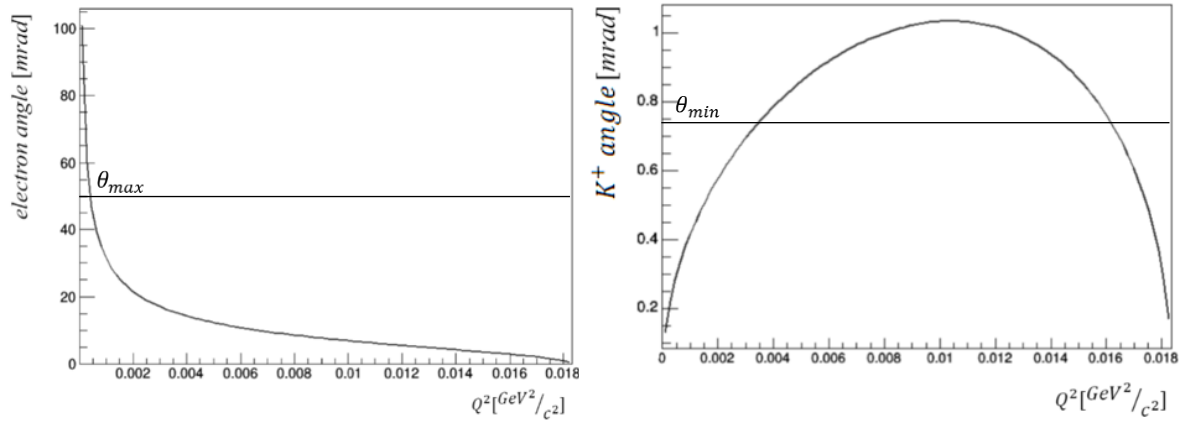


Figura 6.15. Distribuciones esperadas para la dispersión Kaon-electron acotadas con los nuevos límites al incorporar los LAVs en el análisis. El aumento en el ángulo del electron nos permite la obtención de valores mucho más pequeños de Q^2.

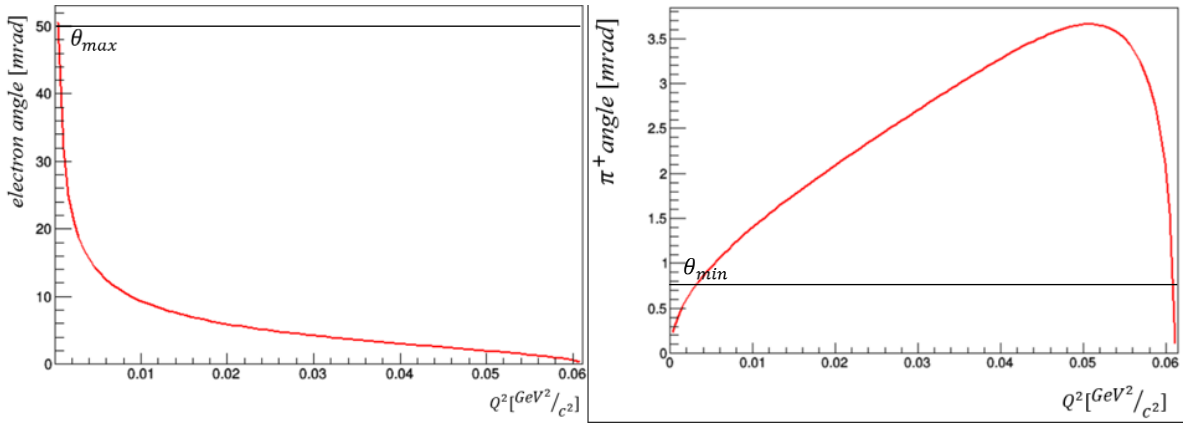


Figura 6.16. Distribuciones esperadas para la dispersión pion-electron acotadas con los nuevos límites al incorporar los LAVs en el análisis. El aumento en el ángulo del electron nos permite la obtención de valores mucho más pequeños de Q^2.

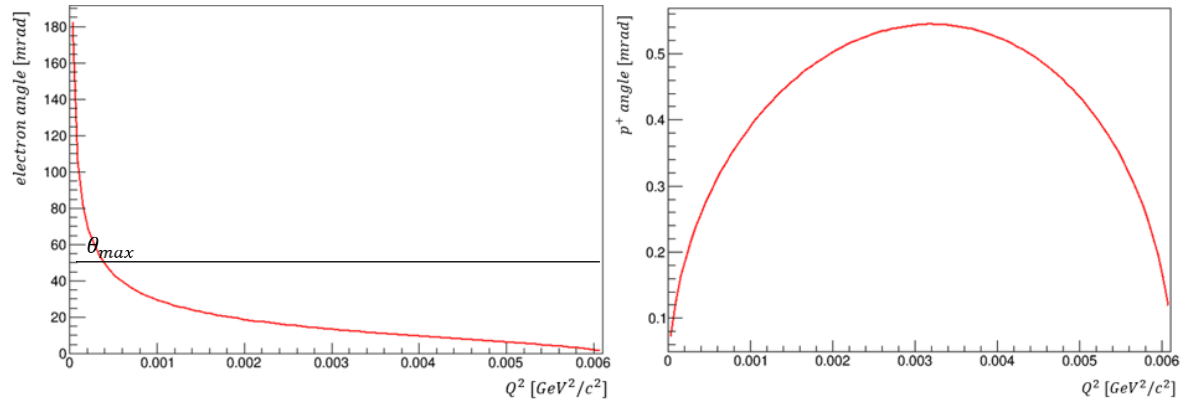


Figura 6.17. Distribuciones esperadas para la dispersión protón-electron acotadas con los nuevos límites al incorporar los LAVs en el análisis.

La incorporación de los detectores LAV nos permite acercarnos más a valores pequeños de Q^2 del lado del ángulo de dispersión de electron, por otra parte, no hay mejoría en la medición del ángulo de dispersión del hadrón, en ninguno de los tres casos. Lo que nos lleva a proponer el uso de otro detector para el ángulo y el momento de las partículas, el RICH (sección 6.2.7). El RICH detecta partículas aun cuando no pasan por los STRAW por lo que eliminaríamos la restricción del ángulo mínimo en las distribuciones de los hadrones. Pero ¿Qué tan bien lo hace?, para responder a esta pregunta recurrimos a simulaciones de Montecarlo las cuales nos permiten estimar la resolución de los detectores que usamos. En la siguiente sección se da una breve explicación de cómo funcionan estas simulaciones. Simulamos dos versiones diferentes del análisis, en la primer versión se usan los STRAW como medidores de ángulo y momento, y en la segunda usamos el RICH y los LAVs como detectores principales con la condición de que no haya señal en los STRAW. Con la incorporación del RICH y LAVs al análisis nos permite acceder a regiones de Q^2 mas pequeñas. Los LAVs nos permiten medir ángulos mayores de electrones y el RICH ángulos menores y energía de los hadrones.

Actualmente se trabaja en una tercer versión en la cual se incorpora al análisis del decaimiento K3Pi, es decir, el Kaon decayendo en tres piones. La idea principal de incorporar estos eventos al análisis es de que es uno de los decaimientos mejor medidos por el experimento. El objetivo es simular un kaón dispersado que después decae en este modo en particular, de esta manera, al detectar el

momento de los tres piones resultantes, podemos calcular el momento del kaón inicial, logrando así aumentar la estadística de kaones además de que con este método es mucho mejor precisa la medición del momento del kaón.

6.5 Simulaciones de Montecarlo.

Las simulaciones de Montecarlo utilizan GEANT4 como herramienta de simulación. GEANT4 es un software para simular la interacción de materia con las partículas que estamos estudiando. Con esta herramienta podemos simular como funcionan todos los detectores presentes en el experimento de una manera muy precisa, además de que incorpora varias interacciones presentes con sus respectivas tasas de reacción.

En GEANT4 se puede elegir qué tipo de interacción y que tipo de partículas queremos simular, así como con que probabilidad ocurre dicha interacción. Se llaman simulaciones de Montecarlo porque utilizamos un generador de números aleatorios para seleccionar al azar cuando un evento es bueno o no. En el código de simulación principal de Montecarlo ya vienen definidos bastante bien procesos como decaimientos de cualquier partícula en cualquier numero de productos, sin embargo, hasta el momento no había código que simule la dispersión hadrón-electron por lo que se tuvo que trabajar en ello.

La manera en la que se pudo simular esta interacción fue separando el proceso en dos partes. En la primera parte se simula un hadrón de $75\ GeV$ que viaja y colisiona con un blanco fijo de electrones, presentes en el GTK3. Se introdujo como probabilidad de ocurrencia la sección eficaz daba por la ecuación 5.6 con valores arbitrarios de $\langle r^2 \rangle$. En la segunda etapa se simula un hadrón y un electron provenientes de la tercera estación del GTK con energía un poco menor a $75\ GeV$ para el hadron y algunos $GeVs$ para el electron, y ángulos de dispersión con la condición de que ambos vectores formen un plano con alguna inclinación φ. Todo esto bajo las condiciones dadas por las ecuaciones 4.34 - 4.37 para las energías y ángulos permitidas después de la dispersión.

Por último, se utiliza una herramienta de GEANT que nos permite empatar estas dos etapas para formar un solo evento. Una vez que tenemos el evento reconstruido se utiliza el generador de números aleatorios para decidir si el evento es bueno o no. Este proceso se repite un número muy grande de veces para así obtener una muestra de datos simulados la cual nos ayudara a comprender como es detectado este proceso por los detectores y que tan bien lo hacen. Como se mencionó anteriormente se utilizaros dos versiones del análisis, cada una habilitando las funciones de diferentes detectores.

En la primer versión del análisis se usa como condición que el hadrón pase a través de mínimo dos estaciones de STRAW. Las siguientes figuras muestran la comparación entre la simulación y los datos tomados para el cálculo del momento y ángulo de las partículas antes y después de la dispersión.

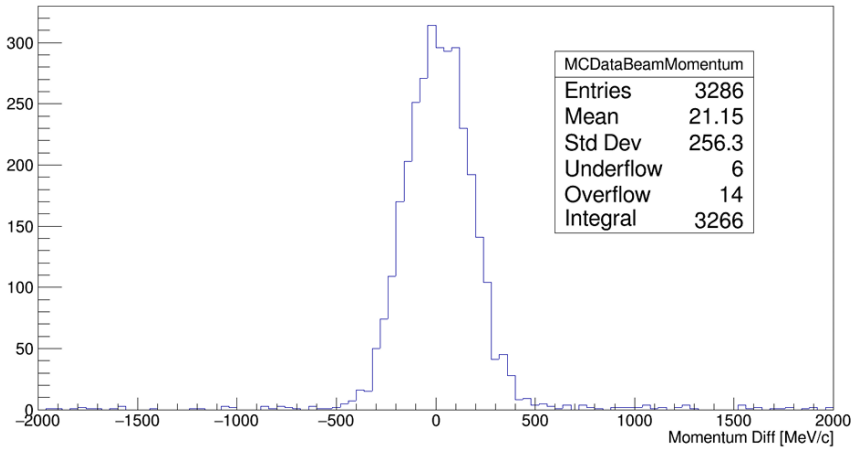


Figura 6.18: Resta de la medición simulada y la medición real con datos para el haz antes de la dispersión.

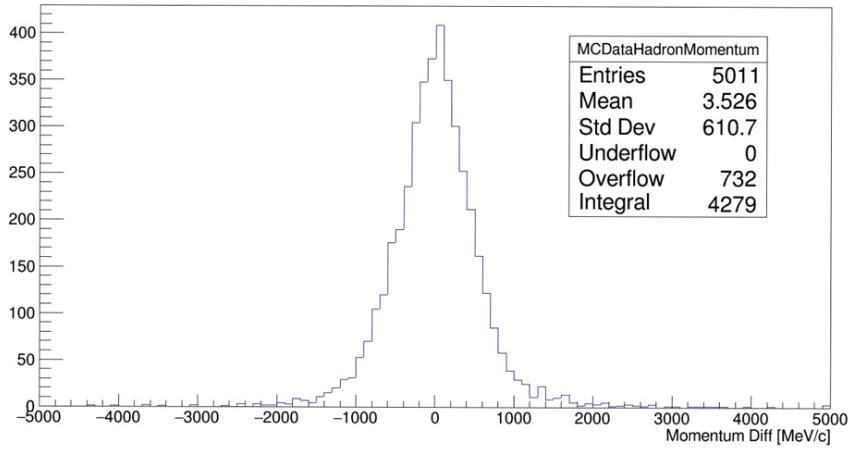


Figura 6.19: Resta de valores de momento del hadrón dispersado simulados por los detectores y los valores reales detectados en los datos.

En la segunda versión del análisis se incorpora el uso de los LAVs para medir el ángulo del electron y el uso del RICH para la medición de ángulo y momento del hadrón difractado, además de que exigimos que se cumpla la condición de que no haya trayectoria detectada en los STRAW. En las siguientes figuras se muestra la diferencia entre simulación y datos con la incorporación de estos dos diferentes detectores.

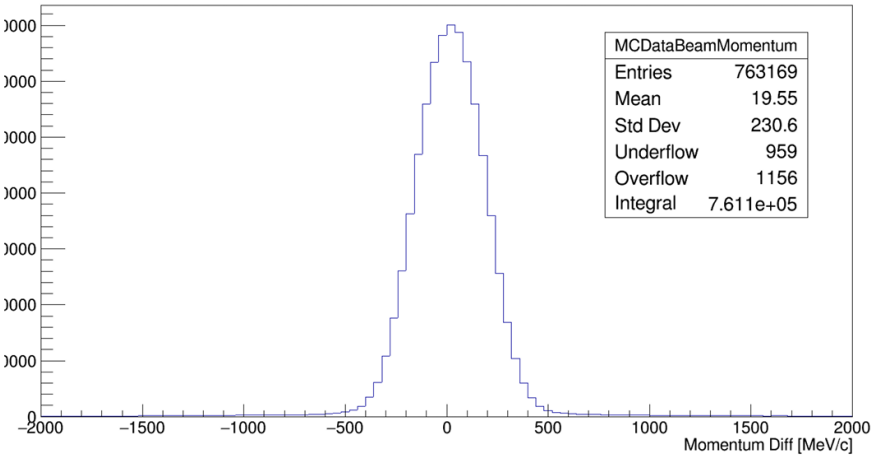


Figura 6.20: Resta de los valores simulados y datos del momento del haz en la versión dos del análisis.

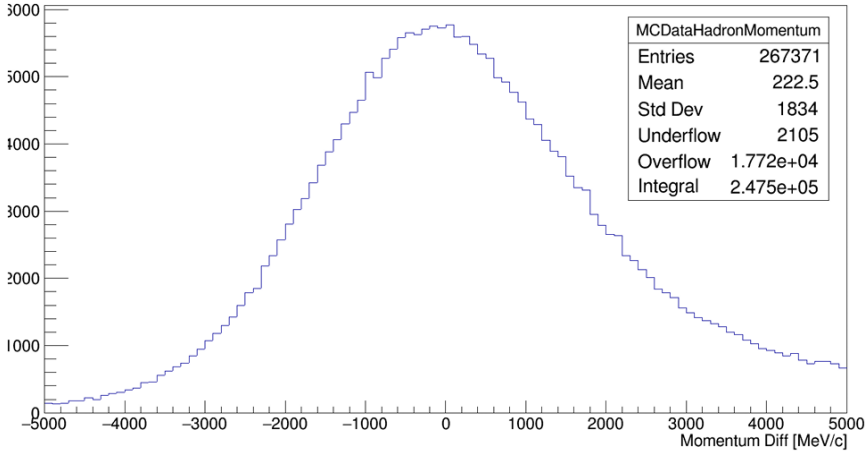


Figura 6.21: Resta entre los valores simulados y datos del momento del hadrón difractado en la versión dos del análisis.

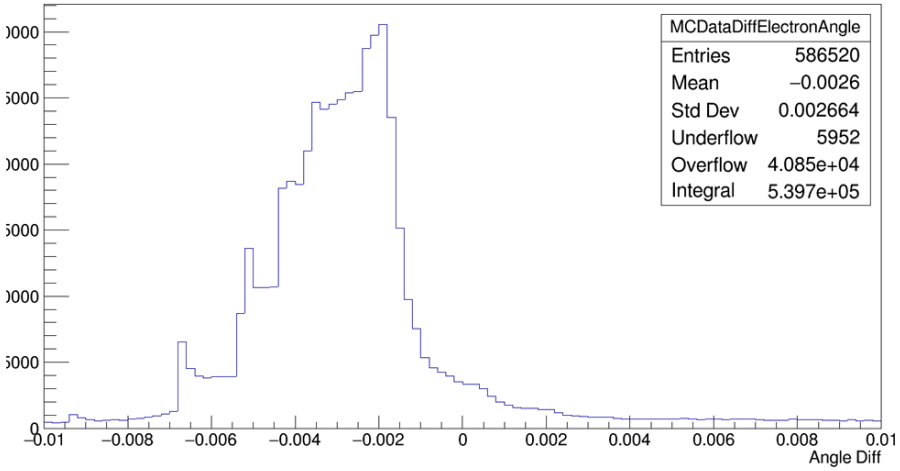


Figura 6.22: Resta de los valores simulados y datos del ángulo del electrón difractado.

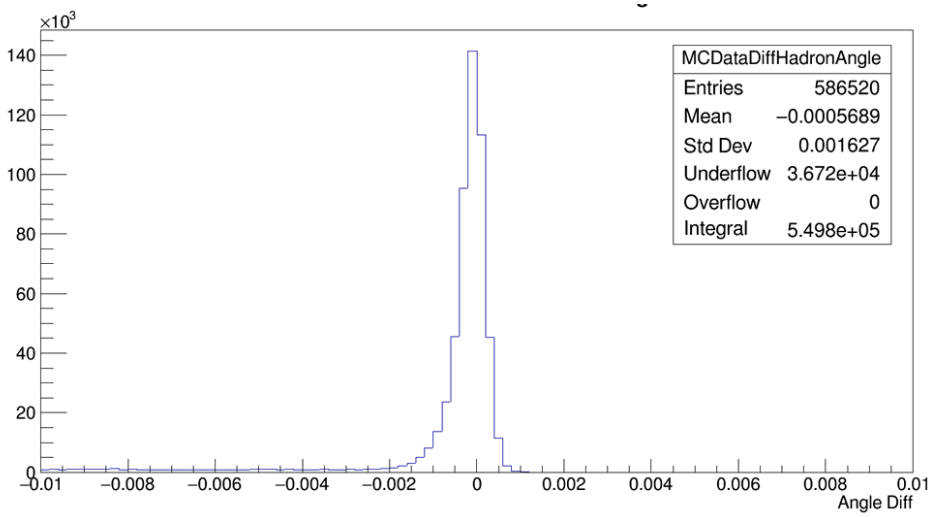


Figura 6.23: Resta de los valores simulados y datos del ángulo del hadrón difractado.

El ancho de las distribuciones en las figuras anteriores nos da una idea de que tan bien se mide una variable cinemática. Entre más delgado sea el pico, mejor es la resolución de los detectores.

Estas graficas nos indican que la resolución de la medición del momento del haz en ambas versiones es prácticamente la misma. La principal diferencia de ambas versiones será entonces en la resolución para medir ángulos y momento después de la dispersión.

La primer versión del análisis nos muestra que la medición del momento del hadrón después de la dispersión es mucho mas precisa que en la segunda versión. Esto se debe a que los STRAW miden el momento mucho mejor que el RICH, pero perdemos rango de medición en el ángulo.

En la segunda versión del análisis observamos una mejoría en la medición del angulo del hadron dispersado, esto porque el RICH sirve muy bien para medir el ángulo, no el momento. Por eso observamos un pico mas ancho en el histograma del momento del hadrón en la versión dos. Sin embargo, para el ángulo del electron observamos un pico ancho recorrido a la derecha, lo ideal es que este centrado en cero. Esto nos indica que tenemos que indagar con más profundidad como mejorar la identificación del electron en los LAV o si hay que hacer correcciones en el código debido a desviaciones del haz.

Capítulo 7:

Procedimiento experimental.

Hasta el momento solo se ha abarcado el procedimiento teórico para la obtención del radio de carga del kaón, en esta sección se dará una breve explicación de cómo se lleva a cabo el proceso experimental para la obtención de datos y el análisis de estos.

En el capítulo cinco se observa que tanto el factor de forma como el radio de carga están en función de Q^2, por lo que el primer paso será obtener las distribuciones de Q^2. En el capítulo cuatro se exponen cuatro formas diferentes para obtener Q^2; a partir de la energía final del electrón, a partir de la energía final del kaón, a partir del ángulo de dispersión del electrón y a partir del ángulo de dispersión del kaón. Una vez escogido el método a utilizar, se llena un histograma con los datos del número de eventos en función de la variable escogida. Las figuras 7.1-4 muestran los histogramas esperados del número de eventos en función de las cuatro variables que se pueden medir en el experimento.

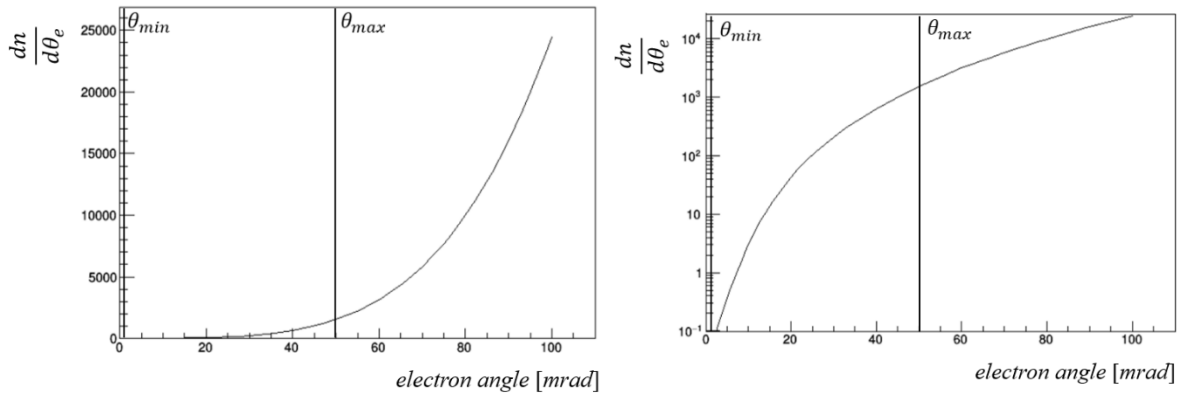


Figura 7.1 Numero de eventos en función del ángulo de dispersión del electrón en escala lineal (izquierda) y logarítmica (derecha).

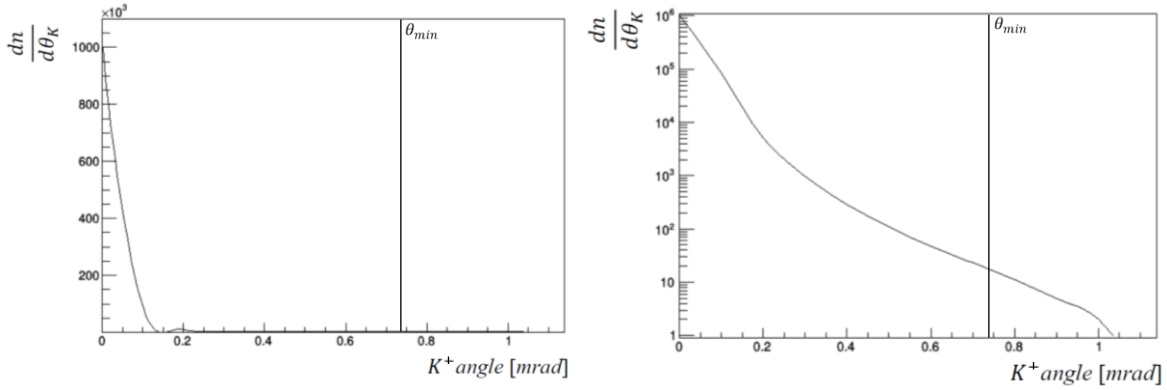


Figura 7.2 Numero de eventos en función del ángulo de dispersión del kaón en escala lineal (izquierda) y logarítmica (derecha).

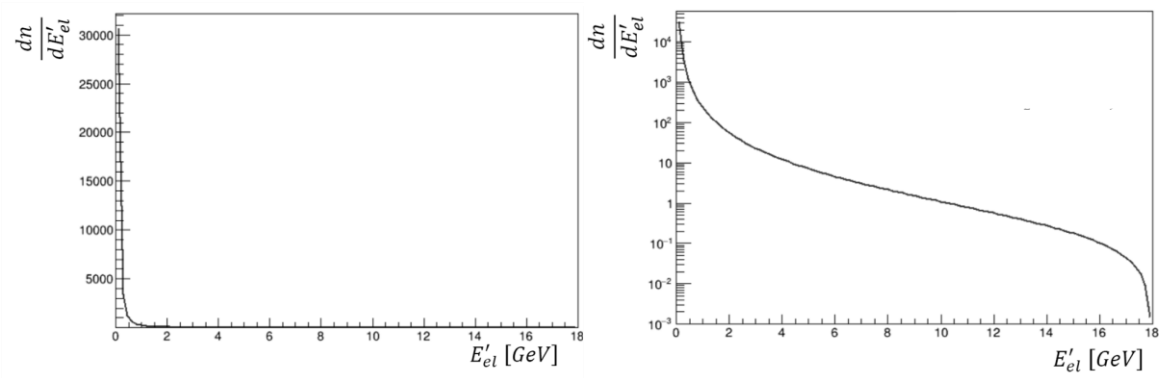
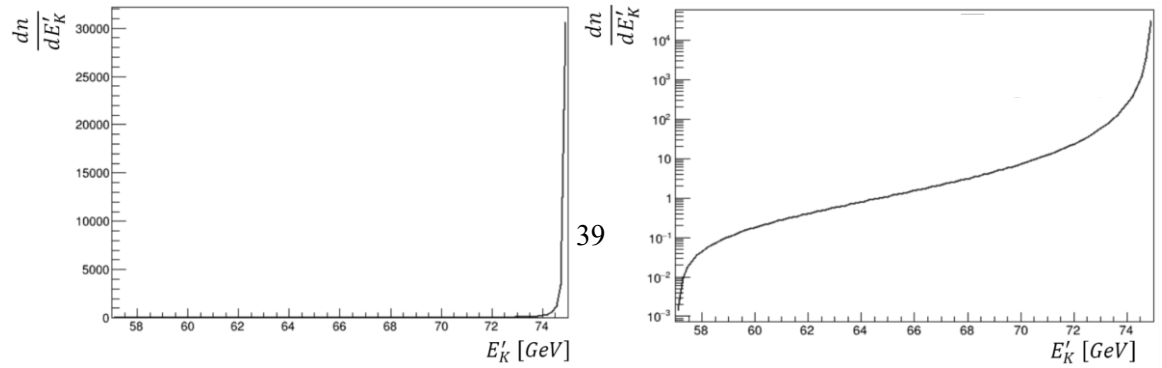


Figura 7.3 Numero de eventos en función la energía final del electrón en escala lineal (izquierda) y logarítmica (derecha).



39

Figura 7.4 Numero de eventos en función la energía final del kaón en escala lineal (izquierda) y logarítmica (derecha).

Usando las ecuaciones 4.34-37, se recalcula esta distribución para que este en función de Q^2, obteniendo de esta manera la sección transversal diferencial experimental. Una vez que se obtiene la sección transversal experimental $\frac{d\sigma}{dQ^2}$ (ecuación 5.6), esta distribución se divide de la siguiente manera, para así quedarnos solo con la parte del factor de forma.

$$\frac{\frac{d\sigma}{dQ^2}}{\frac{4\pi\alpha^2\hbar^2}{Q^4}\left(1-\frac{Q^2}{Q^2_{max}}\right)\left(1+\frac{Q^2}{4M^2c^2}\right)^{-1}} = \left(1+\frac{\langle r^2\rangle Q^2}{6\hbar^2}\right)^{-2} \tag{7.1}$$

Esto nos dará como resultado una distribución nueva la cual será el factor de forma eléctrico cuadrado $G_E^2(Q^2)$. Sacando raíz y aplicando la condición de la ecuación 5.4, el radio de carga cuadrático medio vendrá dado por la pendiente de esta distribución cerca del origen.

Capítulo 8:

Resultados y discusión.

El hecho de que el experimento NA62 está diseñado para el estudio de decaimientos, restringe demasiado las cantidades cinemáticas que podemos medir en un proceso de dispersión debido a la distribución de los detectores. Por lo que en la primer versión del análisis solo podremos ver una pequeña cantidad del total de eventos ya que los STRAW, encargados de medir el ángulo y dirección de los productos, cubren solo una pequeña superficie, permitiéndonos detectar solo las partículas con ángulo de dispersión de entre 0.74 y 11.5 mrad. En la segunda versión, existe una mejoría en el rango de medición del ángulo del electron lo que nos permite obtener valores de Q^2 más cercanos al origen, que es donde nuestro argumento de la pendiente para la obtención del radio de carga tiene validez. Lo ideal para un experimento de dispersión es que los detectores se encuentren cubriendo todo el ángulo solido al rededor del blanco, para que así el rango de ángulos que se pueden detectar sea el mayor.

La figura 8.1 muestra el rango del factor de forma que se podría medir en el experimento ya que, debido a los límites de los ángulos mostrados anteriormente, solo podemos obtener algunos valores para Q^2. (Ver figura 6.12)

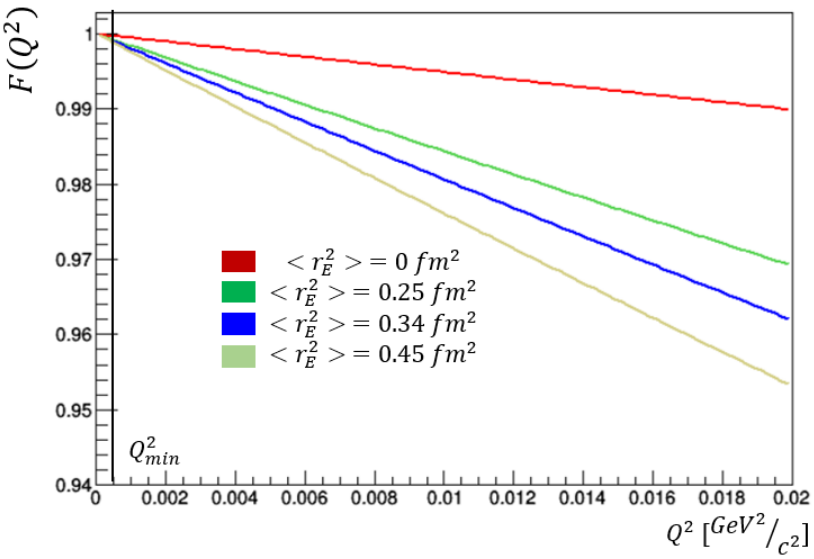


Figura 8.1: Rango de medición para el factor de forma (ecuación 5.3) para diferentes valores arbitrarios del radio de carga.

Con la incorporación de los LAVs al análisis, logramos acercarnos mucho más a las región de Q^2 cercanas a cero, esto, solo si tomamos como variable principal el ángulo de dispersión del electron que es el que nos permite llegara estas regiones de Q^2 pequeños. Tambien se puede urar cualquiera de las otras tres variables disponibles en el análisis, la decisión de cual es mejor utilizar nos se toma en base a las simulaciones.

Capítulo 9:

Conclusión.

Se propuso que esta interacción se estudiara con los datos recabados del experimento NA62 ya que se tenía lo necesario para hacerlo. Se utiliza un haz secundario de K^+, un detector que funge como blanco de electrones (GTK3), así como detectores encargados de medir ángulos y energías.

Se realizo el estudio teórico para obtener la ecuaciones que rigen la cinemática de la dispersión elástica entre el kaón y el electron y se obtuvieron límites de medición con los detectores disponibles en el experimento. Se trabajo con dos versiones de análisis de datos y simulaciones de Montecarlo.

En la primer versión se utilizaron las estaciones de STRAW para medir ángulos y momentos. Las simulaciones arrojan una buena medición en momento, pero un rango reducido para el ángulo del hadrón, además de que no hay detección de electrones. En la segunda versión se incorpora al análisis el uso del RICH para medir ángulo y momento del hadrón y los LAVs para medir ángulos de los electrones. Las simulaciones arrojan una mejoría en la medición del ángulo del hadrón, pero no en momento, se pierde resolución si usamos el RICH para medir el momento. La incorporación de los LAVs permitió la detección de electrones en ángulos mayores a que si solo usamos los STRAW permitiendo acercarnos más a las zonas de Q^2 pequeñas.

Capítulo 10:

Trabajo futuro, implementación de datos.

Hasta el momento solo se ha trabajado en su mayor parte con simulación y una pequeña muestra de datos. Como trabajo futuro, ya en una investigación doctoral, se propone mejorar el código de análisis para poder así trabajar con todos los datos disponibles en la base del experimento NA62.

Actualmente se trabaja en una tercer versión del análisis, el cual incorpora la detección de tres piones provenientes del decaimiento del kaón para el cálculo del momento de una manera más precisa, con la condición de que el kaón haya sido dispersado antes de decaer.

Se espera obtener un resultado positivo tanto para el kaón como para el pion. El protón se menciona en este trabajo solo por completes. Actualmente, el radio de carga del protón está bien medido por diversos experimentos por lo que se planteaba utilizar este resultado como base en este experimento para saber que vamos por el camino correcto. Desgraciadamente no será posible ya que debido a la distribución de los detectores el protón es casi imposible de detectar en NA62.

Anexo:

Código de análisis:

Primera versión:

```
#include <stdlib.h>
#include <iostream>
#include <TChain.h>
#include "Difraction.hh"
#include "functions.hh"
#include "Event.hh"
#include "Persistency.hh"
#include "GeometricAcceptance.hh"
#include "DownstreamTrack.hh"
#include "BeamParameters.hh"
#include "TwoLinesCDA.hh"
#include "TriggerConditions.hh"
#include "KaonDecayConstants.hh"
#include "NA62Global.hh"
#include "RICHParameters.hh"

using namespace std;
using namespace NA62Analysis;

const double MPROT = 938.272;

Difraction::Difraction(Core::BaseAnalysis *ba) : Analyzer(ba, "Difraction")
{

  RequestTree("CHOD", new TRecoCHODEvent, "Reco");
  RequestTree("RICH", new TRecoRICHEvent, "Reco");
  RequestTree("LKr",  new TRecoLKrEvent,  "Reco");
  RequestTree("Spectrometer", new TRecoSpectrometerEvent, "Reco");
  RequestTree("MUV3", new TRecoMUV3Event, "Reco");
  RequestTree("LAV", new TRecoLAVEvent, "Reco");
  RequestTree("IRC", new TRecoIRCEvent, "Reco");
  RequestTree("SAC", new TRecoSACEvent, "Reco");
  RequestTree("Cedar", new TRecoCedarEvent, "Reco");
  RequestTree("GigaTracker", new TRecoGigaTrackerEvent, "Reco");


  fSegmentsAlgo = std::make_unique<StrawSegmentAlgorithm>(ba, this, "SegmentAlgo");

  fMatchingRG = new MatchingRG(ba, this, "MatchingRG");
  fMatchingRG->InitForProcess("");
  fMatchingRG->InitForFinalSelection("");



}

void Difraction::InitOutput(){
}

void Difraction::InitHist(){



  fSegmentsAlgo->Init({"All/"});                    // initialize with default directory
  fSegmentsAlgo->SetHistoFillingFolder(true, "All/");  // enable histogram filling
```

```
  fInputFileName = "bla";

  BookHisto(new TH1F("DownstreamTracks","Number of Downstream Tracks",10,-0.5,9.5));
  BookHisto(new TH1F("NChambers","Number of Straw Stations",5,-0.5,4.5));
  BookHisto(new TH1F("BeamAxisCDA","CDA To Beam",100,0.,100.));
  BookHisto(new TH2F("GTK3Impact","Impact at GTK3",300,-30.,30.,200,-20.,20.));
  BookHisto(new TH1F("NGTKCand","Number of Original GTK Cands",10,-0.5,9.5));
  BookHisto(new TH1F("NCedarCand","Number of Original Cedar Cands",10,-0.5,9.5));
  BookHisto(new TH1F("DTrackMomentum","Momentum Downstream Tracks",400,0.,80000.));
  BookHisto(new TH1F("GTKCandInTime","Number of intime GTK Cands",20,-0.5,19.5));
  BookHisto(new TH1F("GTKHitInTime","Number of intime GTK Hits",100,-0.5,99.5));
  BookHisto(new TH1F("CedarCandInTime","Number of intime Cedar Cands",20,-0.5,19.5));
  BookHisto(new TH1F("CedarCandInTimeAfter","Number of intime Cedar Cands",20,-0.5,19.5));
  BookHisto(new TH2F("GTKCedarCandInTime","Number of intime Cedar and GTK Cands",20,-0.5,19.5,20,-0.5,19.5));
  BookHisto(new TH1F("TimeDiffGTrackCedar","Time Cedar - Time GTK Track",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffTrackCedar","Time Cedar - Time Track",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffTrackGTK","Time GTK - Time Track",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffTrackGTKAfter","Time GTK - Time Track After selection",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffTrackLAV","Time LAV - Time Track",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffCHANTITrack","Time CHANTI - Time Track",100,-5.,5.));
  BookHisto(new TH1F("AngleDiffPlaneLAV","Angle Scat Plane - Phi LAV",800,-2.*TMath::Pi(),2.*TMath::Pi()));
  BookHisto(new TH2D("TimeAngleDiffTrackLAV","Phi Diff vs Time Diff;Track LAV Time Diff [ns];Scatter Plane LAV Phi
Diff",100,-5.,5.,100.,-0.5,0.5));
  BookHisto(new TH1F("ElectronSegment","Electron in LAV or STRAW",8,-0.5,7.5));
  BookHisto(new TH1F("GoodLAV","Number of Good LAV Hits",11,-0.5,10.5));

  std::vector<TString> BinNames = {"None", "LAV", "Segment", "LAV && Segment", "Track", "Track && LAV", "Track &&
Segment", "ALL"};
  for (Int_t i = 0;i<8;i++) fHisto.GetHisto("ElectronSegment")->GetXaxis()->SetBinLabel(i + 1, BinNames[i]);

  BookHisto(new TH1F("HitsOnKaonRing","Number of Hits on Kaon Ring",20,-0.5,19.5));
  BookHisto(new TH1F("RatioHitsOnKaonRing","Ratio obs/exp Number of Hits on Kaon Ring",60,0.,3.));
  BookHisto(new TH1F("HitsOnPionRing","Number of Hits on Pion Ring",20,-0.5,19.5));
  BookHisto(new TH1F("RatioHitsOnPionRing","Ratio obs/exp Number of Hits on Pion Ring",60,0.,3.));

  BookHisto(new TH1F("MomKaonSelected","Momentum of Selected DS Kaon",400,50000.,80000.));
  BookHisto(new TH1F("RICHMomKaonSelected","RICH Momentum of Selected DS Kaon",400,50000.,80000.));
  BookHisto(new TH1F("MomPionSelected","Momentum of Selected DS Pion",400,50000.,80000.));
  BookHisto(new TH1F("RICHMomPionSelected","Momentum of Selected DS Pion",400,50000.,80000.));
  BookHisto(new TH1F("MomProtonSelected","Momentum of Selected DS Proton",400,50000.,80000.));

  BookHisto(new TH1F("AngleKaonSelected","Angle to nominal beam DS Kaon",400,0.,0.010));
  BookHisto(new TH1F("AnglePionSelected","Angle to nominal beam DS Pion",400,0.,0.010));
  BookHisto(new TH1F("AngleProtonSelected","Angle to nominal beam DS Proton",400,0.,0.010));

  BookHisto(new TH2F("AngleMomKaonSelected","Angle to nominal beam DS Kaon vs Momentum",400,50000.,80000.,400,0.,0.010));
  BookHisto(new TH2F("AngleRICHMomKaonSelected","Angle to nominal beam DS Kaon vs RICH
Momentum",400,50000.,80000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomPionSelected","Angle to nominal beam DS Pion vs Momentum",400,50000.,80000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomProtonSelected","Angle to nominal beam DS Proton vs
Momentum",400,50000.,80000.,400,0.,0.010));

  BookHisto(new TH1F("AngleKaonSelectedMatch","Angle to gtk track DS Kaon",400,0.,0.010));
  BookHisto(new TH1F("AnglePionSelectedMatch","Angle to gtk track DS Pion",400,0.,0.010));
  BookHisto(new TH1F("AngleProtonSelectedMatch","Angle to gtk track DS Proton",400,0.,0.010));

  BookHisto(new TH2F("AngleMomDiffKaonSelectedMatch","Angle to gtk track DS Kaon vs Momentum",500,-
25000.,5000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomDiffRICHKaonSelectedMatch","Angle to gtk track DS Kaon vs Momentum",500,-
25000.,5000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomDiffPionSelectedMatch","Angle to gtk track DS Pion vs Momentum",500,-
25000.,5000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomDiffRICHPionSelectedMatch","Angle to gtk track DS Pion vs Momentum",500,-
25000.,5000.,400,0.,0.010));
```

```
  BookHisto(new TH2F("AngleMomDiffProtonSelectedMatch","Angle to gtk track DS Proton vs Momentum",500,-
25000.,5000.,400,0.,0.010));


  BookHisto(new TH1F("ZPosAllTracks","Z with nominal beam of DS Track>65GeV",800,75000.,135000.));
  BookHisto(new TH1F("ZPosKaonSelected","Z with nominal beam of Selected DS Kaon",800,75000.,135000.));
  BookHisto(new TH1F("ZPosPionSelected","Z with nominal beam of Selected DS Pion",400,95000.,105000.));
  BookHisto(new TH1F("ZPosProtonSelected","Z with nominal beam of Selected DS Proton",400,95000.,105000.));
  BookHisto(new TH1F("ZPosExtraTracks","Z with nominal beam of DS extra track",800,75000.,135000.));
  BookHisto(new TH1F("ZPosKaonMatched","Z with matched GTK track of Selected DS Kaon",800,75000.,135000.));

  BookHisto(new TH1F("MomDiff","Momentum Difference GTK Downstream",500,-25000.,5000.));
  BookHisto(new TH1F("RICHMomDiff","Momentum Difference GTK Downstream RICH",500,-25000.,5000.));
  BookHisto(new TH1F("RICHMomDiffKaon","Momentum Difference GTK Downstream RICH",500,-25000.,5000.));
  BookHisto(new TH1F("RICHMomDiffPion","Momentum Difference GTK Downstream RICH",500,-25000.,5000.));


  BookHisto(new TH1F("MomExtraTrack","Momentum of Extra Track from GTK3",400,0.,25000.));
  BookHisto(new TH1F("DiffMomExtraTrack","Momentum Difference Expected Electron to Extra Track",500,-5000.,5000.));

  BookHisto(new TH2F("PosDiffTrackGTK3","Diff en Pos Beam Hadron",100,-20.,20.,100,-20.,20.));
  BookHisto(new TH2F("R2TimeDiff","Distance in GTK3 vs Time Diff",100,-5.,5.,100,0.,20.));
  BookHisto(new TH2F("PosDiffTrackGTK3After","Diff en Pos Beam Hadron After selection",100,-20.,20.,100,-20.,20.));
  BookHisto(new TH2F("R2TimeDiffAfter","Distance in GTK3 vs Time Diff After selection",100,-5.,5.,100,0.,20.));

  BookHisto(new TH2F("PosDiffExtraTrackGTK3","Diff en Pos Beam Electron?",100,-20.,20.,100,-20.,20.));

  BookHisto(new TH1F("SpatProduct","Triple product beam hadron electron",500,-100.,100.));
  BookHisto(new TH1F("AnglePlaneElectron","Angle electron to scatter plane",500,-2*TMath::Pi(),2.*TMath::Pi()));
  BookHisto(new TH1F("MCSpatProduct","Triple product beam hadron electron",500,-100.,100.));
  BookHisto(new TH1F("MCAnglePlaneElectron","Angle electron to scatter plane",500,-2*TMath::Pi(),2.*TMath::Pi()));
  BookHisto(new TH1D("MCBeamHadronElectron","0MCEvents 1Beam 2Hadron 3Electron",4,-0.5,3.5));

  BookHisto(new TH1D("Q2AMCKaon","Q^{2} from Electron Momentum
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2BMCKaon","Q^{2} from Kaon Momentum
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2CMCKaon","Q^{2} from Electron Angle
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DMCKaon","Q^{2} from Kaon Angle
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2EMCKaon","Q^{2} from both Kaon and Elctron Angles
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2FMCKaon","Q^{2} from both Electron Angle and Kaon Energy
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2MCPion","Q^{2} from Electron Momentum
MC",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2MCProton","Q^{2} from Electron Momentum
MC",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BKaon","Q^{2} from Kaon Energies Data;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DKaon","Q^{2} from Kaon momenta Data; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2EKaon","Q^{2} from Kaon and Electron Angles Data; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2FKaon","Q^{2} from Electron LAV Angle and Kaon Energies Data; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BKaonRICH","Q^{2} from Kaon Energies Data RICH;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DKaonRICH","Q^{2} from Kaon momenta Data RICH; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
```

```
  BookHisto(new TH1D("Q2EKaonRICH","Q^{2} from Kaon and Electron Angles Data RICH; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2FKaonRICH","Q^{2} from Electron LAV Angle and Kaon Energies Data RICH; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BKaonT","Q^{2} from Kaon Energies Data and Electron Track;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2BKaonS","Q^{2} from Kaon Energies Data and Electron Segment;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2BKaonL","Q^{2} from Kaon Energies Data and Electron LAV;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));


  BookHisto(new TH1D("Q2BPion","Q^{2} from Pion Energies Data; Q^{2}
[MeV^2/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DPion","Q^{2} from Pion momenta Data;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BProton","Q^{2} from Proton Energies Data; Q^{2}
[MeV^2/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DProton","Q^{2} from Proton momenta Data;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));


  BookHisto(new TH1D("Q2MCQ2BMC","MC Q2-Q2B",400,-10000.,10000.));
  BookHisto(new TH1D("Q2MCQ2CMC","MC Q2-Q2C",400,-10000.,10000.));
  BookHisto(new TH1D("Q2MCQ2DMC","MC Q2-Q2D",400,-10000.,10000.));

  BookHisto(new TH1D("ElectronAngleDiffMC","Difference Electron Angle MC - Data; [rad]",400.,-0.01,0.01));
  BookHisto(new TH1D("ElectronAngleDiffData","Difference Electron Angle Data_expected - Data",400.,-0.01,0.01));
  BookHisto(new TH1D("ElectronAngleMC","Electron Angle MC;[rad]",400.,0.,0.025));
  BookHisto(new TH1D("ElectronAngleData","Electron Angle Data;[rad]",400.,0.,0.025));
  BookHisto(new TH2D("MCElectronAnglevsHadronAngle","MC Electron Angle vs Hadron Angle; Hadron Angle [rad]; Electron Angle
[rad]",400.,0.,0.005,400.,0.,0.025));
  BookHisto(new TH2D("DataElectronSegmentAnglevsHadronAngle","Data Segment Electron Angle vs Hadron Angle; Hadron Angle
[rad]; Electron Angle [rad]",400.,0.,0.005,400.,0.,0.025));
  BookHisto(new TH2D("DataElectronTrackAnglevsHadronAngle","Data Track Electron Angle vs Hadron Angle; Hadron Angle [rad];
Electron Angle [rad]",400.,0.,0.005,400.,0.,0.025));
  BookHisto(new TH2D("DataElectronLAVAnglevsHadronAngle","Data LAV Electron Angle vs Hadron Angle; Hadron Angle [rad];
Electron Angle [rad]",400.,0.,0.015,400.,0.,0.050));
  BookHisto(new TH2D("MCHadronMomDiffvsHadronAngle","MC Electron Angle vs Hadron Angle; Hadron Angle [rad]; Hadron
Momentum Diff [MeV/c]",400.,0.,0.005,500.,-15000.,15000.));
  BookHisto(new TH2D("DataHadronMomDiffvsHadronAngle","Data Electron Angle vs Hadron Angle; Hadron Angle [rad]; Hadron
Momentum Diff [MeV/c]",400.,0.,0.005,500.,-15000.,15000.));
  BookHisto(new TH2F("ElectronAngleTrack","Electron Angle/Momenta Data from track;Angle [rad];Momentum
[MeV/c]",400.,0.,0.025,400,0.,25000.));
  BookHisto(new TH1D("ElectronAngleSegmant","Electron Angle Data from segment;[rad]",400.,0.,0.025));




  BookHisto(new TH1F("CDA","CDA",200,0.,100.));
  BookHisto(new TH1F("CDAElec","CDA with Electron",200,0.,100.));
  BookHisto(new TH1F("CDAMuon","CDA with Muon",200,0.,100.));

  BookHisto(new TH1F("ZVertex", "Z Pos Vertex Single", 2000,0.,200000.));

  BookHisto(new TH1D("MissingMassMuon","Missing Mass #mu Hypo",800,-100000.,30000.));
  BookHisto(new TH1D("MissingMassElec","Missing Mass e Hypo",1300,-100000.,30000.));
  BookHisto(new TH1D("MissingMassElecCut","Missing Mass e Hypo",1300,-100000.,30000.));
  BookHisto(new TH2F("MissingMassMuon2d","Missing Mass #mu Hypo",200,-100000.,100000.,350,-100000.,250000.));
  BookHisto(new TH2F("MissingMassElec2d","Missing Mass e Hypo",400,-100000.,100000.,700,-100000.,250000.));
  BookHisto(new TH1D("MissingMassMuonAll","Missing Mass #mu Hypo",800,-100000.,30000.));
  BookHisto(new TH1D("MissingMassElecAll","Missing Mass e Hypo",800,-100000.,30000.));
  BookHisto(new TH1D("MissingMassElecAllCut","Missing Mass e Hypo",800,-100000.,30000.));
```

```
  BookHisto(new TH2F("MissingMassMuon2dAll","Missing Mass #mu Hypo",200,-100000.,100000.,350,-100000.,250000.));
  BookHisto(new TH2F("MissingMassElec2dAll","Missing Mass e Hypo",200,-100000.,100000.,350,-100000.,250000.));

  BookHisto(new TH1F("Pi0s","Number of Pi0s selected",10,-0.5,9.5));
  BookHisto(new TH1D("Pi0Vertices","Position of Pi0 Vertices", 2000,0.,250000.));
  BookHisto(new TH1D("Pi0Energy","Energy of Pi0", 1000,40000.,80000.));
  BookHisto(new TH1D("Pi0Times","Pi0 times",500,-100.,150.));
  BookHisto(new TH1D("Pi0TimesTrack","Pi0 time vs TtackTime",500,-100.,150.));

  BookHisto(new TH1F("GTKCands","Number of original GTK Candidates",10,-0.5,9.5));
  BookHisto(new TH1F("CedarCands","Number of original Cedar Candidates",10,-0.5,9.5));

  BookHisto(new TH1D("MmissKe2","Missing Mass Ke2",800,-30000.,100000.));
  BookHisto(new TH1D("MmissPie2","Missing Mass Pie2",1600,-100000.,30000.));
  BookHisto(new TH1D("MmissKm2","Missing Mass Km2",800,-30000.,100000.));
  BookHisto(new TH1D("MmissPim2","Missing Mass Pim2",1600,-100000.,30000.));


  BookHisto(new TH1D("EndPosK","MC Decay Position Kaon",200,0.,250000.));
  BookHisto(new TH1D("MCElecMom","Electron Momentum MC",100,0.,10000.));
  BookHisto(new TH1D("MCPi0Mom","Pi0 Momentum MC",200,60000.,80000.));
  BookHisto(new TH1D("MomElectron","Electron Momentum",250,0.,75000.));
  BookHisto(new TH1D("MassPi0eKaon","Mass Pi0 Electron for Kaon Beam",500,0.,500.));
  BookHisto(new TH1D("MassPi0ePion","Mass Pi0 Electron for Pion Beam",500,0.,500.));



  BookHisto(new TH1D("RICHnRings","Numer of Standalone Rings",10,-0.5,9.5));
  BookHisto(new TH2D("RICHRingRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH1F("TimeDiffRingTimeGTK","Time GTK - Time Ring",100,-5.,5.));
  BookHisto(new TH1F("GTKCandRingInTime","Good GTK Cands inTime with Ring",10,-0.5,9.5));
  BookHisto(new TH2D("RICHRingMatchedRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingMatchedRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH1F("TimeDiffRingCedar","Time Cedar - Time Ring",100,-5.,5.));
  BookHisto(new TH1F("CedarCandRingInTime","Good Cedar Cands inTime with Ring",10,-0.5,9.5));
  BookHisto(new TH2D("RICHRingKaonRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingKaonRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingPionRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingPionRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingEPionRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingEPionRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-
500.,300.,400.,0.,220.));

  BookHisto(new TH1D("MCDataBeamMomentum","Difference MC reco Beam Mom;Momentum Diff [MeV/c]",100,-2000.,2000.));
  BookHisto(new TH1D("MCDataHadronMomentum","Difference MC reco Hadron Mom;Momentum Diff [MeV/c]",100,-
5000.,5000.));
  BookHisto(new TH2D("MCDataHadronMomentumvsMom","Difference MC reco Hadron Mom;MC Momentum [MeV/c]; Momentum
Diff [MeV/c]",100,50000.,80000.,100,-5000.,5000.));
  BookHisto(new TH1D("MCDataBeamMomentumX","Difference MC reco Beam Mom X;Momentum Diff  [MeV/c]",100,-200.,200.));
  BookHisto(new TH1D("MCDataBeamMomentumY","Difference MC reco Beam Mom Y;Momentum Diff  [MeV/c]",100,-200.,200.));
  BookHisto(new TH1D("MCDataHadronAngleX","Difference MC reco Hadron Angle X;Angle Diff  [rad]",100,-0.010,0.010));
  BookHisto(new TH1D("MCDataHadronAngleY","Difference MC reco Hadron Angle Y;Angle Diff  [rad]",100,-0.010,0.010));
  BookHisto(new TH2D("MCDataHadronAngleXvsMom","Difference MC reco Hadron Angle X; Momentum [MeV/c]; Angle Diff
[rad]",100,50000.,80000.,100,-0.002,0.002));
```

```
}

void Difraction::DefineMCSimple(){


}

void Difraction::StartOfRunUser(){

}

void Difraction::StartOfBurstUser(){

  fNominalElectronRadius = RICHParameters::GetInstance()->GetElectronRingRadius();
  fFocalLength = GeometricAcceptance::GetInstance()->GetRICHMirrorFocalLength();

}

void Difraction::ProcessSpecialTriggerUser(int iEvent, unsigned int triggerType){

}

void Difraction::Process(int iEvent){




  TRecoLAVEvent *LAVEvent = (TRecoLAVEvent*)GetEvent("LAV");

  TRecoRICHEvent *RICHEvent = (TRecoRICHEvent*)GetEvent("RICH");
  Int_t NRichHits = RICHEvent->GetNHits();

  TRecoLKrEvent* LKrEvent  = (TRecoLKrEvent*)GetEvent("LKr");
  Int_t NClusters = LKrEvent->GetNCandidates();

  TRecoCedarEvent* CedarEvent  = (TRecoCedarEvent*)GetEvent("Cedar");
  Int_t NCedarCand = CedarEvent->GetNCandidates();
  Int_t NCedarHits = CedarEvent->GetNHits();
  FillHisto("NCedarCand",NCedarCand);

  TRecoGigaTrackerEvent* GigaTrackerEvent  = (TRecoGigaTrackerEvent*)GetEvent("GigaTracker");
  Int_t NGigaTrackerCand = GigaTrackerEvent->GetNCandidates();
  FillHisto("NGTKCand",NGigaTrackerCand);

  TRecoSpectrometerEvent*  STRAWEvent  = GetEvent<TRecoSpectrometerEvent>();

  TRecoCHANTIEvent*    CHANTIEvent    = GetEvent<TRecoCHANTIEvent>();

  if (!GetWithMC()) {   // Only check trigger for data
    //  L0TPData* L0data = UserMethods::GetL0Data();
    //  UInt_t TriggerType  = L0data->GetDataType();
    //  Bool_t PhysicsTrigger = false;
    //  Bool_t ControlTrigger = false;
    //  if (TriggerType&0x1)  PhysicsTrigger = true;
    //  if (TriggerType&0x10) ControlTrigger = true;
    //  UInt_t TriggerFlags = L0data->GetTriggerFlags();
    Bool_t ControlTrigger = TriggerConditions::GetInstance()->IsControlTrigger(GetL0Data());
    //  if (!ControlTrigger) return;
  }

  Int_t RunID = GetEventHeader()->GetRunID();
  Int_t EventID = GetEventHeader()->GetEventNumber();
  Int_t BurstID = GetEventHeader()->GetBurstID();
```

```
  Double_t GigaTrackerStationPositionZ[4];
  GigaTrackerStationPositionZ[0] = GeometricAcceptance::GetInstance()->GetZGTK0();
  GigaTrackerStationPositionZ[1] = GeometricAcceptance::GetInstance()->GetZGTK1();
  GigaTrackerStationPositionZ[2] = GeometricAcceptance::GetInstance()->GetZGTK2();
  GigaTrackerStationPositionZ[3] = GeometricAcceptance::GetInstance()->GetZGTK3();

  TLorentzVector MCElectronMom,MCHadronMom,MCBeamMom;
  TLorentzVector MCElectronPos,MCHadronPos,MCBeamPos;
  Double_t Q2_MC,Q2B_MC,Q2C_MC,Q2D_MC,Theta_elec_MC,Theta_hadron_MC;

  //  cout << "BeamKineTree Entries " << fBeamKineTree->GetEntries() << " NBranches " << fBeamKineTree->GetNbranches() <<
endl;
  if (GetWithMC() && std::string(GetCurrentFile()->GetName()).find("MuonHalo") == string::npos) {   // Check true MC distributions

    if (!fInputFileName.EqualTo(GetCurrentFile()->GetName())) {
      fInputFileName = GetCurrentFile()->GetName();
      if (finputFile) delete finputFile;
      finputFile = new TFile(fInputFileName.Data());
      fTreeEvent = -1;
      fBeamKineTree = (TTree*)finputFile->Get("BeamKine/1");
      fBeamKineTree->SetBranchAddress("x", &fX);
      fBeamKineTree->SetBranchAddress("y", &fY);
      fBeamKineTree->SetBranchAddress("z", &fZ);
      fBeamKineTree->SetBranchAddress("Px", &fPx);
      fBeamKineTree->SetBranchAddress("Py", &fPy);
      fBeamKineTree->SetBranchAddress("Pz", &fPz);
      fBeamKineTree->SetBranchAddress("t", &fT);
      fBeamKineTree->SetBranchAddress("PDGid", &fPDGid);
      fBeamKineTree->SetBranchAddress("ProperTime", &fProperTime);
      fBeamKineTree->SetBranchAddress("PathLength", &fPathLength);
      fBeamKineTree->SetBranchAddress("Weight", &fWeight);
      fBeamKineTree->SetBranchAddress("InitX", &fInitX);
      fBeamKineTree->SetBranchAddress("InitY", &fInitY);
      fBeamKineTree->SetBranchAddress("InitZ", &fInitZ);
      fBeamKineTree->SetBranchAddress("InitT", &fInitT);
      fBeamKineTree->SetBranchAddress("IsBeamParticle", &fIsBeamParticle);
    }
    fTreeEvent++;

    Event *evt = GetMCEvent();
    fBeamKineTree->GetEntry(fTreeEvent);
    Bool_t FoundElectron=false,FoundHadron=false,FoundBeam=false;

    //    cout << "fX->size() " << fX->size() << " iEvent " << fTreeEvent <<endl;
    for (UInt_t i=0;i<fX->size();i++) {
      //      cout << " BeamKine " << i << " Mom " << fPx->at(i) << " " << fPy->at(i) << " " << fPz->at(i) << " PDG " << fPDGid->at(i) <<
" Pos " << fX->at(i) << " " << fY->at(i) << " " << fZ->at(i) << endl;
      if ((fPDGid->at(i)==321||fPDGid->at(i)==211||fPDGid->at(i)==2212) && (fInitZ->at(i)-GigaTrackerStationPositionZ[3])<-10000.) {
// Kaon/pion/proton before Difraction
        Double_t Mass;
        if (fPDGid->at(i)==321) Mass = MKCH;  //  Beam is Kaon+
        else if (fPDGid->at(i)==211) Mass = MPI; //  Beam is pion+
        else if (fPDGid->at(i)==2212) Mass = MPROT; //  Beam is proton+
        MCBeamMom.SetXYZM(fPx->at(i),fPy->at(i),fPz->at(i),Mass);
        MCBeamPos.SetXYZT(fX->at(i),fY->at(i),fZ->at(i),fT->at(i));
        FoundBeam = true;
      }
    }

    if (evt->GetNKineParts()) {

      for (Int_t i=0; i<evt->GetNKineParts(); i++) {
        //        cout << "Id " << i << " " << evt->GetKinePart(i)->GetMCTrackID() << " " << evt->GetKinePart(i)->GetCharge()
<< " " << evt->GetKinePart(i)->GetPDGcode() << " " << evt->GetKinePart(i)->GetParentIndex() << " " << evt->GetKinePart(i)-
```

```
>GetProdProcessName() << " " << evt->GetKinePart(i)->GetEndProcessName() << " " << (evt->GetKinePart(i)->GetProdPos()).Z() <<
" " << (evt->GetKinePart(i)->GetEndPos()).Z() << " " << evt->GetKinePart(i)->GetParentIndex() << endl;
      if (evt->GetKinePart(i)->GetParentIndex()==-1 && evt->GetKinePart(i)->GetPDGcode()==11) { // Electron from Difraction
               MCElectronMom = evt->GetKinePart(i)->GetInitial4Momentum();
               MCElectronPos = evt->GetKinePart(i)->GetProdPos();
               FoundElectron = true;
              }
      if (evt->GetKinePart(i)->GetParentIndex()==-1 && (evt->GetKinePart(i)->GetPDGcode()==321||evt->GetKinePart(i)-
>GetPDGcode()==211||evt->GetKinePart(i)->GetPDGcode()==2212) && fabs(evt->GetKinePart(i)->GetProdPos().Z()-
GigaTrackerStationPositionZ[3])<100.) { // Kaon from Difraction
               MCHadronMom = evt->GetKinePart(i)->GetInitial4Momentum();
               MCHadronPos = evt->GetKinePart(i)->GetProdPos();
               FoundHadron = true;
              }
              //       if (evt->GetKinePart(i)->GetParentIndex()==-1 && evt->GetKinePart(i)->GetPDGcode()==321 && (evt-
>GetKinePart(i)->GetProdPos().Z()-GigaTrackerStationPositionZ[3])<-100.) { // Kaon from Difraction
              //               MCBeamMom = evt->GetKinePart(i)->GetInitial4Momentum();
              //               MCBeamPos = evt->GetKinePart(i)->GetProdPos();
              //               FoundBeam = true;
              //             }
    }  // loop over kineparts

   } // are there kineparts?

   if (!(FoundElectron&&FoundHadron&&FoundBeam)) return;  // bad MC event

   Q2_MC = 2.*MEL*(MCElectronMom.E()-MEL);
   Q2B_MC = 2.*MEL*(MCBeamMom.E()-MCHadronMom.E());
   Q2C_MC =
2.*MEL*MCBeamMom.P()*MCElectronMom.P()*cos((MCElectronMom.Vect()).Angle(MCBeamMom.Vect()))/(MEL+MCBeamMom
.E());
   Q2D_MC = 2.*MEL*MCBeamMom.P()*(MCBeamMom.P()-
MCHadronMom.P()*cos((MCHadronMom.Vect()).Angle(MCBeamMom.Vect())))/(MEL+MCBeamMom.E());
   Theta_elec_MC = (MCBeamMom.Vect()).Angle(MCElectronMom.Vect());
   Theta_hadron_MC = (MCBeamMom.Vect()).Angle(MCHadronMom.Vect());

   //    cout << "Q2 MC "  << Q2_MC << " " << Q2B_MC << " " << Q2C_MC << " " << Q2D_MC << " " << Theta_elec_MC << " Spat
Product " << ((MCBeamMom.Vect()).Cross(MCHadronMom.Vect())).Dot(MCElectronMom.Vect()) << " " <<
((MCBeamMom.Vect()).Cross(MCHadronMom.Vect())).Angle(MCElectronMom.Vect()) << endl;

   FillHisto("MCSpatProduct",((MCBeamMom.Vect()).Cross(MCHadronMom.Vect())).Dot(MCElectronMom.Vect()));
   FillHisto("MCAnglePlaneElectron",((MCBeamMom.Vect()).Cross(MCHadronMom.Vect())).Angle(MCElectronMom.Vect()));


   FillHisto("Q2MCKaon",Q2_MC);
   FillHisto("Q2MCPion",Q2_MC);
   FillHisto("Q2MCProton",Q2_MC);

   FillHisto("Q2MCQ2BMC",Q2_MC-Q2B_MC);
   FillHisto("Q2MCQ2CMC",Q2_MC-Q2C_MC);
   FillHisto("Q2MCQ2DMC",Q2_MC-Q2D_MC);
   FillHisto("ElectronAngleMC",Theta_elec_MC);
   FillHisto("MCElectronAnglevsHadronAngle",Theta_hadron_MC,Theta_elec_MC);
   FillHisto("MCHadronMomDiffvsHadronAngle",Theta_hadron_MC,MCBeamMom.P()-MCHadronMom.P());

  } // end of MC code


  OutputState DownstreamState;
  std::vector<DownstreamTrack> DTracks = *(std::vector<DownstreamTrack>*)GetOutput("DownstreamTrackBuilder.Output",
DownstreamState);
  if (DownstreamState!=kOValid) return;

  FillHisto("DownstreamTracks",DTracks.size());
```

```
  if (DTracks.size()<1) return;


  for (UInt_t it=0;it<DTracks.size();it++) {   //  big loop over tracks
    Bool_t HadronIsKaon=false,HadronIsPion=false,HadronIsProton=false;
    //    cout << "Track " << it << " "  << DTracks[it].GetMomentum() << " " << DTracks[it].RICHAssociationSuccessful() << " " <<
DTracks[it].GetRICHMostLikelyHypothesis() << endl;

    FillHisto("DTrackMomentum",DTracks[it].GetMomentum());
    if (DTracks[it].GetMomentum()<50000.) continue;
    if (DTracks[it].GetCharge() != +1) continue;
    if (DTracks[it].GetChi2() > 40.0) continue;
    FillHisto("NChambers",DTracks[it].GetNChambers());
    if (DTracks[it].GetNChambers() < 3) continue;
    if (!GeometricAcceptance::GetInstance()->InAcceptance(&DTracks[it], NA62::kMUV3)) continue;
    if (DTracks[it].MUV3AssociationExists()) continue;
    if (!GeometricAcceptance::GetInstance()->InAcceptance(&DTracks[it], NA62::kLKr)) continue;
    if (!DTracks[it].LKrAssociationExists()) continue;
    if (DTracks[it].GetLKrEoP()<0.1 || DTracks[it].GetLKrEoP()>0.9) continue;  // remove muons and electrons
    if (!GeometricAcceptance::GetInstance()->InAcceptance(&DTracks[it], NA62::kRICH)) continue;
    FillHisto ("BeamAxisCDA", DTracks[it].GetNominalBeamAxisCDA());
    if (DTracks[it].GetNominalBeamAxisCDA() > 60.0) continue;
    FillHisto("ZPosAllTracks",DTracks[it].GetBeamAxisVertex().Z());
    if (fabs((DTracks[it].GetNominalBeamAxisVertex()).Z()-GigaTrackerStationPositionZ[3])>2000.) continue;

FillHisto("GTK3Impact",DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),DTracks[it].yAtBeforeMagnet(GigaTrackerSta
tionPositionZ[3]));
    if (fabs((DTracks[it].GetNominalBeamAxisVertex()).X())>GeometricAcceptance::GetInstance()->GetGTKStationHalfWidthX())
continue;
    if (fabs((DTracks[it].GetNominalBeamAxisVertex()).Y())>GeometricAcceptance::GetInstance()->GetGTKStationHalfWidthY())
continue;


    Double_t TrackTime = -99999999999.;
    if (!DTracks[it].RICHAssociationSuccessful()) continue;
    if (DTracks[it].GetRICHRingPredictedNHits(NA62::kRICHHypothesisKaon)>2 &&
DTracks[it].GetRICHMostLikelyHypothesis()==NA62::kRICHHypothesisKaon) {
      FillHisto("HitsOnKaonRing",DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisKaon));

FillHisto("RatioHitsOnKaonRing",DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisKaon)/DTracks[it].GetRICHRingPredicted
NHits(NA62::kRICHHypothesisKaon));
      if (DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisKaon) > 2 &&
DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisKaon)/DTracks[it].GetRICHRingPredictedNHits(NA62::kRICHHypothesisK
aon) > 0.4) {
                HadronIsKaon = true;
                if (DTracks[it].RICHRingTimeExists(NA62::kRICHHypothesisKaon)) TrackTime =
DTracks[it].GetRICHRingTime(NA62::kRICHHypothesisKaon);
                else if (DTracks[it].CHODTimeExists()) TrackTime = DTracks[it].GetCHODTime();
                else TrackTime = DTracks[it].GetTrackTime();
      }
    }
    //    else if (DTracks[it].GetRICHMostLikelyHypothesis()==NA62::kRICHHypothesisPion) {
    else if (DTracks[it].GetRICHLikelihoodPion()>0.95) {  // possible overlap with muon and electron hypo
      FillHisto("HitsOnPionRing",DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisPion));

FillHisto("RatioHitsOnPionRing",DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisPion)/DTracks[it].GetRICHRingPredictedN
Hits(NA62::kRICHHypothesisPion));
      if (DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisPion) > 2 &&
DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisPion)/DTracks[it].GetRICHRingPredictedNHits(NA62::kRICHHypothesisPio
n) > 0.4) {
                HadronIsPion = true;
                if (DTracks[it].RICHRingTimeExists(NA62::kRICHHypothesisPion)) TrackTime =
DTracks[it].GetRICHRingTime(NA62::kRICHHypothesisPion);
                else if (DTracks[it].CHODTimeExists()) TrackTime = DTracks[it].GetCHODTime();
                else TrackTime = DTracks[it].GetTrackTime();
```

```
    }
   }
   else if (DTracks[it].GetRICHMostLikelyHypothesis()==NA62::kRICHHypothesisBackground) {
    HadronIsProton = true;
    if (DTracks[it].CHODTimeExists()) TrackTime = DTracks[it].GetCHODTime();
    else TrackTime = DTracks[it].GetTrackTime();
   }
   if (!HadronIsKaon && !HadronIsPion && !HadronIsProton) continue;

   Double_t HadronGamma = 1.;
   if (HadronIsKaon || HadronIsPion) {
    Double_t HadronBeta = (2.*fFocalLength*fFocalLength - fNominalElectronRadius*fNominalElectronRadius)/
          (2.*fFocalLength*fFocalLength - DTracks[it].GetRICHRingRadius()*DTracks[it].GetRICHRingRadius());
    HadronGamma = 1./sqrt(1.-HadronBeta*HadronBeta);
   }

   TLorentzVector HadronMom,HadronMomRICH;
   if (HadronIsKaon) {
    HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MKCH);
    HadronMomRICH.SetVectM((DTracks[it].GetMomentumBeforeMagnet().Unit())*MKCH*sqrt(HadronGamma*HadronGamma-
1.),MKCH);
    FillHisto("MomKaonSelected",DTracks[it].GetMomentum());
    FillHisto("RICHMomKaonSelected",HadronMomRICH.Vect().Mag());
    FillHisto("ZPosKaonSelected",DTracks[it].GetBeamAxisVertex().Z());
    FillHisto("AngleKaonSelected",(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    FillHisto("AngleMomKaonSelected",DTracks[it].GetMomentum(),(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    FillHisto("AngleRICHMomKaonSelected",HadronMomRICH.Vect().Mag(),(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
   }
   if (HadronIsPion) {
    HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MPI);
    HadronMomRICH.SetVectM((DTracks[it].GetMomentumBeforeMagnet().Unit())*MPI*sqrt(HadronGamma*HadronGamma-
1.),MPI);
    FillHisto("MomPionSelected",DTracks[it].GetMomentum());
    FillHisto("RICHMomPionSelected",HadronMomRICH.Vect().Mag());
    FillHisto("ZPosPionSelected",DTracks[it].GetBeamAxisVertex().Z());
    FillHisto("AnglePionSelected",(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    FillHisto("AngleMomPionSelected",DTracks[it].GetMomentum(),(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
   }
   if (HadronIsProton) {
    HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MPROT);
    HadronMomRICH = HadronMom;
    FillHisto("MomProtonSelected",DTracks[it].GetMomentum());
    FillHisto("ZPosProtonSelected",DTracks[it].GetBeamAxisVertex().Z());
    FillHisto("AngleProtonSelected",(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    FillHisto("AngleMomProtonSelected",DTracks[it].GetMomentum(),(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
   }

   if (GetWithMC()) {
        FillHisto("MCDataHadronMomentum",MCHadronMom.P()-(DTracks[it].GetMomentum()));
   }

   /*
   // try some sophisticated matching
   std::vector<TVector3>  matchedGTKMomenta;
   std::vector<int> matchedGTKIDs;
   std::vector<double> matchedGTKTimes;
   std::vector<TVector3> matchedVertices;
```

```
    TRecoSpectrometerCandidate *STRAWCand = static_cast<TRecoSpectrometerCandidate*>(STRAWEvent-
>GetCandidate(GoodTrack[0]));
    Double_t tKTAG = TrackTime;
    fMatchingRG->Process(GigaTrackerEvent, STRAWCand, tKTAG, tKTAG, TrackTime, 1, "");
    fMatchingRG->FinalSelection(tKTAG, TrackTime, 1, "");
    matchedGTKIDs = fMatchingRG->GetMatchedGTKIDs();
    if (matchedGTKIDs.at(0)==-1) return; // GTK matching failed
    //    if (matchedGTKIDs.size()>1) continue; // we only want one match.  If commented, we get the "best" match
    matchedGTKMomenta = fMatchingRG->GetGTKMomentaAtVertices();
    matchedVertices = fMatchingRG->GetVertices();
    FillHisto("ZPosKaonMatched",(matchedVertices.at(0)).Z());
    if (fabs((matchedVertices.at(0)).Z()-GigaTrackerStationPositionZ[3])>600*2.5) return; // sigma=600, 2.5sigma
    */

    std::vector<UInt_t> GoodGTK;
    FillHisto("GTKCands",NGigaTrackerCand);
    Double_t R2;
    Double_t Chi2=9999999999999.;
    Int_t iigtk=-1;
    for (Int_t igtk=0;igtk<GigaTrackerEvent->GetNCandidates();igtk++) {
      TRecoGigaTrackerCandidate* GTKCand = (TRecoGigaTrackerCandidate*) GigaTrackerEvent->GetCandidate(igtk);
      FillHisto("TimeDiffTrackGTK",GTKCand->GetTime()-TrackTime);
      FillHisto("PosDiffTrackGTK3",GTKCand->GetPosition(3).X()-
DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),GTKCand->GetPosition(3).Y()-
DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]));
      R2 = TMath::Power(GTKCand->GetPosition(3).X()-DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),2) +
TMath::Power(GTKCand->GetPosition(3).Y()-DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]),2);
      FillHisto("R2TimeDiff",GTKCand->GetTime()-TrackTime,R2);
      if((GTKCand->GetType()==15 || GTKCand->GetType()==14) &&
                fabs(GTKCand->GetTime()-TrackTime)<0.5 &&
                fabs(DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3])-GTKCand->GetPosition(3).X())<3.0 &&
                fabs(DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3])-GTKCand->GetPosition(3).Y())<3.0) {
               GoodGTK.push_back(igtk);
               if (GTKCand->GetChi2()<Chi2) {
                 iigtk = igtk;
                 Chi2 = GTKCand->GetChi2();
               }
      }
      //      cout << "GTKMatch " << it << " " << igtk << " " << HadronIsPion << " " << HadronIsKaon << " " << HadronIsProton << "
Time " << TrackTime << " " << GTKCand->GetTime() << " Pos " << DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3])
<< " " << GTKCand->GetPosition(3).X() << " " << DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]) << " " <<
GTKCand->GetPosition(3).Y() << " " << " Prop " << GTKCand->GetType() << " " << GTKCand->GetChi2() << " " <<
GoodGTK.size() << endl;
    }
    FillHisto("GTKCandInTime",GoodGTK.size());
    if (GoodGTK.size()!=1) continue;
    TRecoGigaTrackerCandidate* GTKCand = (TRecoGigaTrackerCandidate*) GigaTrackerEvent->GetCandidate(iigtk);

    FillHisto("MomDiff",DTracks[it].GetMomentum()-GTKCand->GetMomentum().Mag());
    FillHisto("RICHMomDiff",HadronMomRICH.Vect().Mag()-GTKCand->GetMomentum().Mag());

    if (HadronIsKaon) {
      FillHisto("RICHMomDiffKaon",HadronMomRICH.Vect().Mag()-GTKCand->GetMomentum().Mag());
      FillHisto("AngleKaonSelectedMatch",(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
      FillHisto("AngleMomDiffKaonSelectedMatch",DTracks[it].GetMomentum()-GTKCand->GetMomentum().Mag(),(GTKCand-
>GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
      FillHisto("AngleMomDiffRICHKaonSelectedMatch",HadronMomRICH.Vect().Mag()-GTKCand-
>GetMomentum().Mag(),(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));

    }
    if (HadronIsPion) {
      FillHisto("RICHMomDiffPion",HadronMomRICH.Vect().Mag()-GTKCand->GetMomentum().Mag());
      FillHisto("AnglePionSelectedMatch",(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
      FillHisto("AngleMomDiffPionSelectedMatch",DTracks[it].GetMomentum()-GTKCand->GetMomentum().Mag(),(GTKCand-
>GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
```

```
      FillHisto("AngleMomDiffRICHPionSelectedMatch",HadronMomRICH.Vect().Mag()-GTKCand-
>GetMomentum().Mag(),(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    }
    if (HadronIsProton) {
      FillHisto("AngleProtonSelectedMatch",(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
      FillHisto("AngleMomDiffProtonSelectedMatch",DTracks[it].GetMomentum()-GTKCand->GetMomentum().Mag(),(GTKCand-
>GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    }

    if (GetWithMC()) {
       FillHisto("MCDataDiffHadronAngle",(MCHadronMom.Vect()).Angle(MCBeamMom.Vect())-(GTKCand-
>GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
     }

    if (GetWithMC()) {
       FillHisto("MCDataBeamMomentum",MCBeamMom.P()-GTKCand->GetMomentum().Mag());
       FillHisto("MCDataBeamMomentumX",MCBeamMom.Px()-GTKCand->GetMomentum().X());
       FillHisto("MCDataBeamMomentumY",MCBeamMom.Py()-GTKCand->GetMomentum().Y());
     }

    FillHisto("TimeDiffTrackGTKAfter",GTKCand->GetTime()-TrackTime);
    FillHisto("PosDiffTrackGTK3After",GTKCand->GetPosition(3).X()-
DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),GTKCand->GetPosition(3).Y()-
DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]));
    R2 = TMath::Power(GTKCand->GetPosition(3).X()-DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),2) +
TMath::Power(GTKCand->GetPosition(3).Y()-DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]),2);
    FillHisto("R2TimeDiffAfter",GTKCand->GetTime()-TrackTime,R2);

    std::vector<UInt_t> GoodCedar;
    FillHisto("CedarCands",NCedarCand);
    for (UInt_t icedar=0;icedar<NCedarCand;icedar++) {
      TRecoCedarCandidate* CedarCand = (TRecoCedarCandidate*) CedarEvent->GetCandidate(icedar);
      FillHisto("TimeDiffGTrackCedar",CedarCand->GetTime()-GTKCand->GetTime());
      FillHisto("TimeDiffTrackCedar",CedarCand->GetTime()-TrackTime);
      if (fabs(CedarCand->GetTime()-GTKCand->GetTime())<0.5 && CedarCand->GetNSectors()>5) GoodCedar.push_back(icedar);
    }
    Int_t nh=0;
    for (Int_t icedar=0;icedar<NCedarHits;icedar++) {
      TRecoCedarHit* CedarHit = (TRecoCedarHit*) CedarEvent->GetHit(icedar);
      FillHisto("TimeDiffPositronCedarHit",TrackTime-CedarHit->GetTime());
      if (fabs(TrackTime-CedarHit->GetTime())<1.) nh++;
    }
    FillHisto("CedarCandInTime",GoodCedar.size());
    if (GoodCedar.size()>1) continue;

    Bool_t BeamisKaon = false;
    Bool_t BeamisPion = true;
    Bool_t BeamisExtremePion = false;
    if (GoodCedar.size()==1) {
      TRecoCedarCandidate* CedarCand = (TRecoCedarCandidate*) CedarEvent->GetCandidate(GoodCedar[0]);
      if (CedarCand->GetNSectors()>5) BeamisKaon=true;
      if (CedarCand->GetNSectors()>2) BeamisPion=false;
      if (CedarCand->GetNSectors()<3) BeamisExtremePion=true;
    }
    if (nh<5) BeamisPion=true;

    Bool_t ExtraGTKHits=false;
    Int_t nGTKHitInTime[5]={0,0,0,0,0};
    for (Int_t ih=0;ih<GigaTrackerEvent->GetNHits();ih++) {
      TRecoGigaTrackerHit* GTKHit = (TRecoGigaTrackerHit*) GigaTrackerEvent->GetHit(ih);
      if (fabs(GTKHit->GetTime()-GTKCand->GetTime())<1.0) {
                nGTKHitInTime[4]++;
                for (Int_t igtk=0;igtk<=3;igtk++) {
                  if (fabs((GTKHit->GetPosition()).Z()-GigaTrackerStationPositionZ[igtk])<100.) nGTKHitInTime[igtk]++;
                }
```

```
    }
   }
   FillHisto("GTKHitInTime",nGTKHitInTime[4]);
   if ((nGTKHitInTime[1]>1&&nGTKHitInTime[2]>1) || (nGTKHitInTime[1]>1&&nGTKHitInTime[3]>1) ||
(nGTKHitInTime[2]>1&&nGTKHitInTime[3]>1) || (nGTKHitInTime[0]>1&&nGTKHitInTime[1]>1) ||
(nGTKHitInTime[0]>1&&nGTKHitInTime[2]>1) || (nGTKHitInTime[0]>1&&nGTKHitInTime[3]>1)) {
    ExtraGTKHits = true;
   }


   TLorentzVector BeamMom;
   if (BeamisKaon) {
    BeamMom.SetVectM(GTKCand->GetMomentum(),MKCH);
    HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MKCH);
   }
   else if (BeamisPion) {
    if (HadronIsPion) {
            BeamMom.SetVectM(GTKCand->GetMomentum(),MPI);
            HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MPI);
    }
    if (HadronIsProton) {
            BeamMom.SetVectM(GTKCand->GetMomentum(),MPROT);
            HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MPROT);
    }
   }
   else continue;

   Int_t ExtraCHANTIHit=0;
   for (Int_t i=0;i<CHANTIEvent->GetNHits();i++) {
    TRecoCHANTIHit* Hit = (TRecoCHANTIHit*) CHANTIEvent->GetHit(i);
    FillHisto("TimeDiffCHANTITrack",Hit->GetTime()-TrackTime);
    if (fabs(Hit->GetTime()-TrackTime)<3.) ExtraCHANTIHit++;
   }
   if (ExtraCHANTIHit>1) continue;

   Double_t Phi_Plane = (GTKCand->GetMomentum().Cross(DTracks[it].GetMomentumBeforeMagnet())).Phi()-TMath::Pi()/2.;
   if (Phi_Plane < -TMath::Pi()) Phi_Plane = 2.*TMath::Pi() + Phi_Plane;


   Double_t Q2B, Q2D;

   Q2B = 2.*MEL*(BeamMom.E()-HadronMom.E());
   Q2D = 2.*MEL*BeamMom.P()*(BeamMom.P()-
HadronMom.P()*cos((HadronMom.Vect()).Angle(BeamMom.Vect())))/(MEL+BeamMom.E());
   //  cout << "Beam " << BeamisKaon << " " << BeamisPion << " " << BeamMom.E() << " " << BeamMom.P() << " " <<
BeamMom.M() << " Hadron " << HadronIsKaon << " " << HadronIsPion << " " << HadronIsProton << " " << HadronMom.E() << " "
<< HadronMom.P() << " " << HadronMom.M() << " Q2 " << Q2B << " " << Q2D << endl;
   Double_t Theta_elec_data_expected =
acos((MEL+BeamMom.E())*Q2B/2./MEL/BeamMom.P()/sqrt(Q2B*Q2B/4./MEL/MEL+Q2B));
   Double_t E_elec_data_expected = Q2B/(2.*MEL)+MEL;

   Double_t Theta_hadron_data = (BeamMom.Vect()).Angle(HadronMom.Vect());


   Int_t ExtraTrack=0;
   Int_t GoodElectronTrack = -1;
   for (UInt_t it2=0;it2<DTracks.size();it2++) {
    if (it==it2) continue;
    if (fabs(DTracks[it2].GetTrackTime()-TrackTime)<5.) {
            ExtraTrack++;
            FillHisto("ZPosExtraTracks",DTracks[it2].GetBeamAxisVertex().Z());
            if (fabs(DTracks[it2].GetBeamAxisVertex().Z()-GigaTrackerStationPositionZ[3]) < 2000.) {
              FillHisto("PosDiffExtraTrackGTK3",GTKCand->GetPosition(3).X()-
DTracks[it2].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),GTKCand->GetPosition(3).Y()-
DTracks[it2].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]));
```

```
                FillHisto("MomExtraTrack",DTracks[it2].GetMomentum());
                FillHisto("DiffMomExtraTrack",DTracks[it2].GetMomentum()-E_elec_data_expected);

FillHisto("DataElectronTrackAnglevsHadronAngle",Theta_hadron_data,(BeamMom.Vect()).Angle(DTracks[it2].GetMomentumBefore
Magnet()));
                if (fabs(DTracks[it2].GetMomentum()-E_elec_data_expected)<1500.) {
                  GoodElectronTrack = it2;
                  ExtraTrack--;
                }
              }
    }

   }
   if (ExtraTrack>0) continue;

   // LAV check:  Is there a hit from the electron in the LAV?
   Int_t GoodLAV = 0;
   //    std::cout << "LAV NCands " << LAVEvent->GetNCandidates() << std::endl;
   for (UInt_t il=0;il<LAVEvent->GetNCandidates();il++) {
    TRecoLAVCandidate* LAVCand = (TRecoLAVCandidate*)LAVEvent->GetCandidate(il);
    //      cout << il << " Time " << TrackTime << " " << LAVCand->GetTime() << " Phi Plane " << Phi_Plane << " " << " Pos " <<
LAVCand->GetPosition().X() << " " << LAVCand->GetPosition().Y() << " " << LAVCand->GetPosition().Z() << " " << LAVCand-
>GetPosition().Phi() << " Energy " << LAVCand->GetEnergy() << " Type " << LAVCand->GetClusterType() << endl;
     TVector3 ElectronCand = LAVCand->GetPosition()-GTKCand->GetPosition(3);
     FillHisto("TimeDiffTrackLAV",LAVCand->GetTime()-TrackTime);
     if (fabs(LAVCand->GetTime()-TrackTime+0.2)>4.) continue;
     FillHisto("DataElectronLAVAnglevsHadronAngle",Theta_hadron_data,(BeamMom.Vect()).Angle(ElectronCand));
     FillHisto("AngleDiffPlaneLAV",Phi_Plane-LAVCand->GetPosition().Phi());
     FillHisto("TimeAngleDiffTrackLAV",LAVCand->GetTime()-TrackTime,Phi_Plane-LAVCand->GetPosition().Phi());
     if (
                fabs(Phi_Plane-LAVCand->GetPosition().Phi()-0.087)<0.08 ||
                fabs(Phi_Plane-LAVCand->GetPosition().Phi()+TMath::Pi()-0.087)<0.08 ||
                fabs(Phi_Plane-LAVCand->GetPosition().Phi()-TMath::Pi()-0.087)<0.08
                ) GoodLAV++;; // for now
     if (GetWithMC()) {
     FillHisto("MCDataDiffElectronAngle",(MCBeamMom.Vect()).Angle(MCElectronMom.Vect())-(GTKCand-
>GetMomentum().Angle(ElectronCand)));
     }
   }

   FillHisto("GoodLAV",GoodLAV);
   if (GoodLAV>2) continue;

   // multi-track segments evaluation
   TVector3 position,direction;
   Bool_t GoodSegment = false;
   std::vector<Int_t> trackIDs = {(int)it};
   if (GoodElectronTrack > -1) trackIDs.push_back(GoodElectronTrack);
   fSegmentsAlgo->Process(GetEvent<TRecoSpectrometerEvent>(), trackIDs, GTKCand->GetPosition(3),TrackTime);
   fIsSegments = fSegmentsAlgo->GetIsSegment();
   //  cout << "Segments " << fIsSegments << endl;
   if (fIsSegments) {
    const std::vector<StrawSegmentAlgorithm::TrackSegment>& segments_before_magnet =
                fSegmentsAlgo->GetTrackSegmentsBeforeMagnet();
    for (const auto & segment : segments_before_magnet) {
                position = segment.GetPosition();
                direction = segment.GetDirection();
                //      cout << " Pos " << position.X() << " " << position.Y() << " " << position.Z() << " Dir " << direction.X() << " " <<
direction.Y() <<  " " << direction.Z() << endl;
     }
     if (segments_before_magnet.size() == 1) {
                FillHisto("ElectronAngleSegmant",(BeamMom.Vect()).Angle(direction));
                Double_t Theta_elec_data = (BeamMom.Vect()).Angle(direction);
                if (GetWithMC()) {
                  FillHisto("ElectronAngleMC",Theta_elec_MC);
```

```
                FillHisto("ElectronAngleDiffMC",Theta_elec_MC-Theta_elec_data);
              }
              FillHisto("ElectronAngleData",Theta_elec_data);
              FillHisto("DataElectronSegmentAnglevsHadronAngle",Theta_hadron_data,Theta_elec_data);
              FillHisto("DataHadronMomDiffvsHadronAngle",Theta_hadron_data,BeamMom.P()-HadronMom.P());
              FillHisto("ElectronAngleDiffData",Theta_elec_data_expected-Theta_elec_data);
              if (fabs(Theta_elec_data_expected-Theta_elec_data)<0.002) GoodSegment = true;
     }
     if (!GoodSegment) continue;
    }

    if ((GoodLAV>0) && GoodSegment && (GoodElectronTrack>-1)) FillHisto("ElectronSegment",7.);
    else if ((GoodLAV>0) && GoodSegment) FillHisto("ElectronSegment",3.);
    else if ((GoodLAV>0) && (GoodElectronTrack>-1)) FillHisto("ElectronSegment",5.);
    else if (GoodSegment && (GoodElectronTrack>-1)) FillHisto("ElectronSegment",6.);
    else if (GoodLAV>0) FillHisto("ElectronSegment",1.);
    else if (GoodSegment) FillHisto("ElectronSegment",2.);
    else if (GoodElectronTrack>-1) FillHisto("ElectronSegment",4.);
    else FillHisto("ElectronSegment",0.);
    if ((GoodLAV==0) && !GoodSegment && (GoodElectronTrack==-1)) continue;  // It has to be one...
    if ( ((GoodLAV>0) && GoodSegment) || ((GoodLAV>0) && (GoodElectronTrack>-1)) || (GoodSegment && (GoodElectronTrack>-
1)) || ((GoodLAV>0) && GoodSegment && (GoodElectronTrack>-1))) continue;    // but only one...

    if (BeamisKaon && HadronIsKaon) {
     FillHisto("Q2BKaon",Q2B);
     FillHisto("Q2DKaon",Q2D);
     if (GoodElectronTrack>0) FillHisto("Q2BKaonT",Q2B);
     if (GoodSegment) FillHisto("Q2BKaonS",Q2B);
     if (GoodLAV>0) FillHisto("Q2BKaonL",Q2B);
    }
    if (BeamisPion) {
     if (HadronIsPion) {
              FillHisto("Q2BPion",Q2B);
              FillHisto("Q2DPion",Q2D);
     }
     if (HadronIsProton) {
              FillHisto("Q2BProton",Q2B);
              FillHisto("Q2DProton",Q2D);
     }
    }

    //  cout << "Q2 Data " << Q2_MC << " " << Q2B << " " << Q2D << " Electron Angle Exp " << Theta_elec_MC << " " <<
Theta_elec_data_expected << " Meas " << Theta_elec_data << endl;

  }  //   loop over tracks
}


void Difraction::PostProcess(){

  //  if (GigaTrackerEvent) delete GigaTrackerEvent;

}

void Difraction::EndOfBurstUser(){
              /// \MemberDescr
              /// This method is called when a new file is opened in the ROOT TChain (corresponding to a start/end of burst in the normal
NA62 data taking) + at the end of the last file\n
              /// Do here your start/end of burst processing if any.
              /// Be careful: this is called after the event/file has changed.
              /// \EndMemberDescr
}

void Difraction::EndOfRunUser(){
              /// \MemberDescr
```

```
	/// This method is called at the end of the processing (corresponding to a end of run in the normal NA62 data taking)\n
	/// Do here your end of run processing if any\n
	/// \EndMemberDescr

}

void Difraction::EndOfJobUser(){
  fSegmentsAlgo->SaveAllPlots();
  SaveAllPlots();
}

void Difraction::DrawPlot(){
	/// \MemberDescr
	/// This method is called at the end of processing to draw plots when the -g option is used.\n
	/// If you want to draw all the plots, just call\n
	/// \code
	///		DrawAllPlots();
	/// \endcode
	/// Or get the pointer to the histogram with\n
	/// \code
	///		fHisto.GetTH1("histoName");// for TH1
	///		fHisto.GetTH2("histoName");// for TH2
	///		fHisto.GetGraph("graphName");// for TGraph and TGraphAsymmErrors
	///	fHisto.GetHisto("histoName");// for TH1 or TH2 (returns a TH1 pointer)
	/// \endcode
	/// and manipulate it as usual (TCanvas, Draw, ...)\n
	/// \EndMemberDescr
}

Difraction::~Difraction(){
	/// \MemberDescr
	/// Destructor of the Analyzer. If you allocated any memory for class
	/// members, delete them here.
	/// \EndMemberDescr
}
```

Segunda versión:

```
#include <stdlib.h>
#include <iostream>
#include <TChain.h>
#include "Difraction2.hh"
#include "functions.hh"
#include "Event.hh"
#include "Persistency.hh"
#include "GeometricAcceptance.hh"
#include "DownstreamTrack.hh"
#include "BeamParameters.hh"
#include "TwoLinesCDA.hh"
#include "TriggerConditions.hh"
#include "KaonDecayConstants.hh"
#include "NA62Global.hh"
#include "RICHParameters.hh"

using namespace std;
using namespace NA62Analysis;

const double MPROT = 938.272;

Difraction2::Difraction2(Core::BaseAnalysis *ba) : Analyzer(ba, "Difraction2")
{

  RequestTree("CHOD", new TRecoCHODEvent, "Reco");
  RequestTree("RICH", new TRecoRICHEvent, "Reco");
  RequestTree("LKr",  new TRecoLKrEvent,  "Reco");
```

```
  RequestTree("Spectrometer", new TRecoSpectrometerEvent, "Reco");
  RequestTree("MUV3", new TRecoMUV3Event, "Reco");
  RequestTree("LAV", new TRecoLAVEvent, "Reco");
  RequestTree("IRC", new TRecoIRCEvent, "Reco");
  RequestTree("SAC", new TRecoSACEvent, "Reco");
  RequestTree("Cedar", new TRecoCedarEvent, "Reco");
  RequestTree("GigaTracker", new TRecoGigaTrackerEvent, "Reco");


  //  fSegmentsAlgo = std::make_unique<StrawSegmentAlgorithm>(ba, this, "SegmentAlgo");

  fMatchingRG = new MatchingRG(ba, this, "MatchingRG");
  fMatchingRG->InitForProcess("");
  fMatchingRG->InitForFinalSelection("");



}

void Difraction2::InitOutput()
{
}

void Difraction2::InitHist()
{


  /*
  fSegmentsAlgo->Init({"All/"});                    // initialize with default directory
  fSegmentsAlgo->SetHistoFillingFolder(true, "All/");  // enable histogram filling
  */

  fInputFileName = "bla";

  BookHisto(new TH1F("DownstreamTracks","Number of Downstream Tracks",10,-0.5,9.5));
  BookHisto(new TH1F("NChambers","Number of Straw Stations",5,-0.5,4.5));
  BookHisto(new TH1F("BeamAxisCDA","CDA To Beam",100,0.,100.));
  BookHisto(new TH2F("GTK3Impact","Impact at GTK3",300,-30.,30.,200,-20.,20.));
  BookHisto(new TH1F("NGTKCand","Number of Original GTK Cands",10,-0.5,9.5));
  BookHisto(new TH1F("NCedarCand","Number of Original Cedar Cands",10,-0.5,9.5));
  BookHisto(new TH1F("DTrackMomentum","Momentum Downstream Tracks",400,0.,80000.));
  BookHisto(new TH1F("GTKCandInTime","Number of intime GTK Cands",20,-0.5,19.5));
  BookHisto(new TH1F("GTKHitInTime","Number of intime GTK Hits",100,-0.5,99.5));
  BookHisto(new TH1F("CedarCandInTime","Number of intime Cedar Cands",20,-0.5,19.5));
  BookHisto(new TH1F("CedarCandInTimeAfter","Number of intime Cedar Cands",20,-0.5,19.5));
  BookHisto(new TH2F("GTKCedarCandInTime","Number of intime Cedar and GTK Cands",20,-0.5,19.5,20,-0.5,19.5));
  BookHisto(new TH1F("TimeDiffGTrackCedar","Time Cedar - Time GTK Track",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffTrackCedar","Time Cedar - Time Track",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffTrackGTK","Time GTK - Time Track",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffTrackGTKAfter","Time GTK - Time Track After selection",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffTrackLAV","Time LAV - Time Track",100,-5.,5.));
  BookHisto(new TH1F("TimeDiffCHANTITrack","Time CHANTI - Time Track",100,-5.,5.));
  BookHisto(new TH1F("AngleDiffPlaneLAV","Angle Scat Plane - Phi LAV",800,-2.*TMath::Pi(),2.*TMath::Pi()));
  BookHisto(new TH2D("TimeAngleDiffTrackLAV","Phi Diff vs Time Diff;Track LAV Time Diff [ns];Scatter Plane LAV Phi
Diff",100,-5.,5.,100.,-0.5,0.5));
  BookHisto(new TH1F("ElectronSegment","Electron in LAV or STRAW",8,-0.5,7.5));
  BookHisto(new TH1F("GoodLAV","Number of Good LAV Hits",11,-0.5,10.5));

  std::vector<TString> BinNames = {"None", "LAV", "Segment", "LAV && Segment", "Track", "Track && LAV", "Track &&
Segment", "ALL"};
  for (Int_t i = 0;i<8;i++) fHisto.GetHisto("ElectronSegment")->GetXaxis()->SetBinLabel(i + 1, BinNames[i]);

  BookHisto(new TH1F("HitsOnKaonRing","Number of Hits on Kaon Ring",20,-0.5,19.5));
  BookHisto(new TH1F("RatioHitsOnKaonRing","Ratio obs/exp Number of Hits on Kaon Ring",60,0.,3.));
  BookHisto(new TH1F("HitsOnPionRing","Number of Hits on Pion Ring",20,-0.5,19.5));
```

```
  BookHisto(new TH1F("RatioHitsOnPionRing","Ratio obs/exp Number of Hits on Pion Ring",60,0.,3.));

  BookHisto(new TH1F("MomKaonSelected","Momentum of Selected DS Kaon",400,50000.,80000.));
  BookHisto(new TH1F("RICHMomKaonSelected","RICH Momentum of Selected DS Kaon",400,50000.,80000.));
  BookHisto(new TH1F("MomPionSelected","Momentum of Selected DS Pion",400,50000.,80000.));
  BookHisto(new TH1F("RICHMomPionSelected","Momentum of Selected DS Pion",400,50000.,80000.));
  BookHisto(new TH1F("MomProtonSelected","Momentum of Selected DS Proton",400,50000.,80000.));

  BookHisto(new TH1F("AngleKaonSelected","Angle to nominal beam DS Kaon",400,0.,0.010));
  BookHisto(new TH1F("AnglePionSelected","Angle to nominal beam DS Pion",400,0.,0.010));
  BookHisto(new TH1F("AngleProtonSelected","Angle to nominal beam DS Proton",400,0.,0.010));

  BookHisto(new TH2F("AngleMomKaonSelected","Angle to nominal beam DS Kaon vs Momentum",400,50000.,80000.,400,0.,0.010));
  BookHisto(new TH2F("AngleRICHMomKaonSelected","Angle to nominal beam DS Kaon vs RICH
Momentum",400,50000.,80000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomPionSelected","Angle to nominal beam DS Pion vs Momentum",400,50000.,80000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomProtonSelected","Angle to nominal beam DS Proton vs
Momentum",400,50000.,80000.,400,0.,0.010));

  BookHisto(new TH1F("AngleKaonSelectedMatch","Angle to gtk track DS Kaon",400,0.,0.010));
  BookHisto(new TH1F("AnglePionSelectedMatch","Angle to gtk track DS Pion",400,0.,0.010));
  BookHisto(new TH1F("AngleProtonSelectedMatch","Angle to gtk track DS Proton",400,0.,0.010));

  BookHisto(new TH2F("AngleMomDiffKaonSelectedMatch","Angle to gtk track DS Kaon vs Momentum",500,-
25000.,5000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomDiffRICHKaonSelectedMatch","Angle to gtk track DS Kaon vs Momentum",500,-
25000.,5000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomDiffPionSelectedMatch","Angle to gtk track DS Pion vs Momentum",500,-
25000.,5000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomDiffRICHPionSelectedMatch","Angle to gtk track DS Pion vs Momentum",500,-
25000.,5000.,400,0.,0.010));
  BookHisto(new TH2F("AngleMomDiffProtonSelectedMatch","Angle to gtk track DS Proton vs Momentum",500,-
25000.,5000.,400,0.,0.010));


  BookHisto(new TH1F("ZPosAllTracks","Z with nominal beam of DS Track>65GeV",800,75000.,135000.));
  BookHisto(new TH1F("ZPosKaonSelected","Z with nominal beam of Selected DS Kaon",800,75000.,135000.));
  BookHisto(new TH1F("ZPosPionSelected","Z with nominal beam of Selected DS Pion",400,95000.,105000.));
  BookHisto(new TH1F("ZPosProtonSelected","Z with nominal beam of Selected DS Proton",400,95000.,105000.));
  BookHisto(new TH1F("ZPosExtraTracks","Z with nominal beam of DS extra track",800,75000.,135000.));
  BookHisto(new TH1F("ZPosKaonMatched","Z with matched GTK track of Selected DS Kaon",800,75000.,135000.));

  BookHisto(new TH1F("MomDiff","Momentum Difference GTK Downstream",500,-25000.,5000.));
  BookHisto(new TH1F("RICHMomDiff","Momentum Difference GTK Downstream RICH",500,-25000.,5000.));
  BookHisto(new TH1F("RICHMomDiffKaon","Momentum Difference GTK Downstream RICH",500,-25000.,5000.));
  BookHisto(new TH1F("RICHMomDiffPion","Momentum Difference GTK Downstream RICH",500,-25000.,5000.));


  BookHisto(new TH1F("MomExtraTrack","Momentum of Extra Track from GTK3",400,0.,25000.));
  BookHisto(new TH1F("DiffMomExtraTrack","Momentum Difference Expected Electron to Extra Track",500,-5000.,5000.));

  BookHisto(new TH2F("PosDiffTrackGTK3","Diff en Pos Beam Hadron",100,-20.,20.,100,-20.,20.));
  BookHisto(new TH2F("R2TimeDiff","Distance in GTK3 vs Time Diff",100,-5.,5.,100,0.,20.));
  BookHisto(new TH2F("PosDiffTrackGTK3After","Diff en Pos Beam Hadron After selection",100,-20.,20.,100,-20.,20.));
  BookHisto(new TH2F("R2TimeDiffAfter","Distance in GTK3 vs Time Diff After selection",100,-5.,5.,100,0.,20.));

  BookHisto(new TH2F("PosDiffExtraTrackGTK3","Diff en Pos Beam Electron?",100,-20.,20.,100,-20.,20.));

  BookHisto(new TH1F("SpatProduct","Triple product beam hadron electron",500,-100.,100.));
  BookHisto(new TH1F("AnglePlaneElectron","Angle electron to scatter plane",500,-2*TMath::Pi(),2.*TMath::Pi()));
  BookHisto(new TH1F("MCSpatProduct","Triple product beam hadron electron",500,-100.,100.));
  BookHisto(new TH1F("MCAnglePlaneElectron","Angle electron to scatter plane",500,-2*TMath::Pi(),2.*TMath::Pi()));
  BookHisto(new TH1D("MCBeamHadronElectron","0MCEvents 1Beam 2Hadron 3Electron",4,-0.5,3.5));
```

```
  BookHisto(new TH1D("Q2AMCKaon","Q^{2} from Electron Momentum
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2BMCKaon","Q^{2} from Kaon Momentum
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2CMCKaon","Q^{2} from Electron Angle
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DMCKaon","Q^{2} from Kaon Angle
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2EMCKaon","Q^{2} from both Kaon and Elctron Angles
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2FMCKaon","Q^{2} from both Electron Angle and Kaon Energy
MC",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2MCPion","Q^{2} from Electron Momentum
MC",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2MCProton","Q^{2} from Electron Momentum
MC",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BKaon","Q^{2} from Kaon Energies Data;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DKaon","Q^{2} from Kaon momenta Data; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2EKaon","Q^{2} from Kaon and Electron Angles Data; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2FKaon","Q^{2} from Electron LAV Angle and Kaon Energies Data; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BKaonRICH","Q^{2} from Kaon Energies Data RICH;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DKaonRICH","Q^{2} from Kaon momenta Data RICH; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2EKaonRICH","Q^{2} from Kaon and Electron Angles Data RICH; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2FKaonRICH","Q^{2} from Electron LAV Angle and Kaon Energies Data RICH; Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BKaonT","Q^{2} from Kaon Energies Data and Electron Track;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2BKaonS","Q^{2} from Kaon Energies Data and Electron Segment;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2BKaonL","Q^{2} from Kaon Energies Data and Electron LAV;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MKCH*MKCH+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BPion","Q^{2} from Pion Energies Data; Q^{2}
[MeV^2/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DPion","Q^{2} from Pion momenta Data;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));

  BookHisto(new TH1D("Q2BProton","Q^{2} from Proton Energies Data; Q^{2}
[MeV^2/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));
  BookHisto(new TH1D("Q2DProton","Q^{2} from Proton momenta Data;Q^{2}
[MeV^{2}/c^{2}];d#sigma/dQ^{2}",400,0.,4.*MEL*MEL*75000.*75000./(MPI*MPI+2.*MEL*75000.)));


  BookHisto(new TH1D("Q2MCQ2BMC","MC Q2-Q2B",400,-10000.,10000.));
  BookHisto(new TH1D("Q2MCQ2CMC","MC Q2-Q2C",400,-10000.,10000.));
  BookHisto(new TH1D("Q2MCQ2DMC","MC Q2-Q2D",400,-10000.,10000.));

  BookHisto(new TH1D("ElectronAngleDiffMC","Difference Electron Angle MC - Data; [rad]",400.,-0.01,0.01));
  BookHisto(new TH1D("ElectronAngleDiffData","Difference Electron Angle Data_expected - Data",400.,-0.01,0.01));
  BookHisto(new TH1D("ElectronAngleMC","Electron Angle MC;[rad]",400.,0.,0.025));
  BookHisto(new TH1D("ElectronAngleData","Electron Angle Data;[rad]",400.,0.,0.025));
  BookHisto(new TH2D("MCElectronAnglevsHadronAngle","MC Electron Angle vs Hadron Angle; Hadron Angle [rad]; Electron Angle
[rad]",400.,0.,0.005,400.,0.,0.025));
```

```
  BookHisto(new TH2D("DataElectronSegmentAnglevsHadronAngle","Data Segment Electron Angle vs Hadron Angle; Hadron Angle [rad]; Electron Angle [rad]",400.,0.,0.005,400.,0.,0.025));
  BookHisto(new TH2D("DataElectronTrackAnglevsHadronAngle","Data Track Electron Angle vs Hadron Angle; Hadron Angle [rad]; Electron Angle [rad]",400.,0.,0.005,400.,0.,0.025));
  BookHisto(new TH2D("DataElectronLAVAnglevsHadronAngle","Data LAV Electron Angle vs Hadron Angle; Hadron Angle [rad]; Electron Angle [rad]",400.,0.,0.015,400.,0.,0.050));
  BookHisto(new TH2D("MCHadronMomDiffvsHadronAngle","MC Electron Angle vs Hadron Angle; Hadron Angle [rad]; Hadron Momentum Diff [MeV/c]",400.,0.,0.005,500.,-15000.,15000.));
  BookHisto(new TH2D("DataHadronMomDiffvsHadronAngle","Data Electron Angle vs Hadron Angle; Hadron Angle [rad]; Hadron Momentum Diff [MeV/c]",400.,0.,0.005,500.,-15000.,15000.));
  BookHisto(new TH2F("ElectronAngleTrack","Electron Angle/Momenta Data from track;Angle [rad];Momentum [MeV/c]",400.,0.,0.025,400,0.,25000.));
  BookHisto(new TH1D("ElectronAngleSegmant","Electron Angle Data from segment;[rad]",400.,0.,0.025));

  BookHisto(new TH1F("CDA","CDA",200,0.,100.));
  BookHisto(new TH1F("CDAElec","CDA with Electron",200,0.,100.));
  BookHisto(new TH1F("CDAMuon","CDA with Muon",200,0.,100.));

  BookHisto(new TH1F("ZVertex", "Z Pos Vertex Single", 2000,0.,200000.));

  BookHisto(new TH1D("MissingMassMuon","Missing Mass #mu Hypo",800,-100000.,30000.));
  BookHisto(new TH1D("MissingMassElec","Missing Mass e Hypo",1300,-100000.,30000.));
  BookHisto(new TH1D("MissingMassElecCut","Missing Mass e Hypo",1300,-100000.,30000.));
  BookHisto(new TH2F("MissingMassMuon2d","Missing Mass #mu Hypo",200,-100000.,100000.,350,-100000.,250000.));
  BookHisto(new TH2F("MissingMassElec2d","Missing Mass e Hypo",400,-100000.,100000.,700,-100000.,250000.));
  BookHisto(new TH1D("MissingMassMuonAll","Missing Mass #mu Hypo",800,-100000.,30000.));
  BookHisto(new TH1D("MissingMassElecAll","Missing Mass e Hypo",800,-100000.,30000.));
  BookHisto(new TH1D("MissingMassElecAllCut","Missing Mass e Hypo",800,-100000.,30000.));
  BookHisto(new TH2F("MissingMassMuon2dAll","Missing Mass #mu Hypo",200,-100000.,100000.,350,-100000.,250000.));
  BookHisto(new TH2F("MissingMassElec2dAll","Missing Mass e Hypo",200,-100000.,100000.,350,-100000.,250000.));

  BookHisto(new TH1F("Pi0s","Number of Pi0s selected",10,-0.5,9.5));
  BookHisto(new TH1D("Pi0Vertices","Position of Pi0 Vertices", 2000,0.,250000.));
  BookHisto(new TH1D("Pi0Energy","Energy of Pi0", 1000,40000.,80000.));
  BookHisto(new TH1D("Pi0Times","Pi0 times",500,-100.,150.));
  BookHisto(new TH1D("Pi0TimesTrack","Pi0 time vs TtackTime",500,-100.,150.));

  BookHisto(new TH1F("GTKCands","Number of original GTK Candidates",10,-0.5,9.5));
  BookHisto(new TH1F("CedarCands","Number of original Cedar Candidates",10,-0.5,9.5));

  BookHisto(new TH1D("MmissKe2","Missing Mass Ke2",800,-30000.,100000.));
  BookHisto(new TH1D("MmissPie2","Missing Mass Pie2",1600,-100000.,30000.));
  BookHisto(new TH1D("MmissKm2","Missing Mass Km2",800,-30000.,100000.));
  BookHisto(new TH1D("MmissPim2","Missing Mass Pim2",1600,-100000.,30000.));

  BookHisto(new TH1D("EndPosK","MC Decay Position Kaon",200,0.,250000.));
  BookHisto(new TH1D("MCElecMom","Electron Momentum MC",100,0.,10000.));
  BookHisto(new TH1D("MCPi0Mom","Pi0 Momentum MC",200,60000.,80000.));
  BookHisto(new TH1D("MomElectron","Electron Momentum",250,0.,75000.));
  BookHisto(new TH1D("MassPi0eKaon","Mass Pi0 Electron for Kaon Beam",500,0.,500.));
  BookHisto(new TH1D("MassPi0ePion","Mass Pi0 Electron for Pion Beam",500,0.,500.));

  BookHisto(new TH1D("RICHnRings","Numer of Standalone Rings",10,-0.5,9.5));
  BookHisto(new TH2D("RICHRingRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH1F("TimeDiffRingTimeGTK","Time GTK - Time Ring",100,-5.,5.));
  BookHisto(new TH1F("GTKCandRingInTime","Good GTK Cands inTime with Ring",10,-0.5,9.5));
  BookHisto(new TH2D("RICHRingMatchedRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingMatchedRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH1F("TimeDiffRingCedar","Time Cedar - Time Ring",100,-5.,5.));
  BookHisto(new TH1F("CedarCandRingInTime","Good Cedar Cands inTime with Ring",10,-0.5,9.5));
```

```
  BookHisto(new TH2D("RICHRingKaonRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingKaonRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingPionRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingPionRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingEPionRadiusvsCenterX","Single RICH Radius vs X Center; X Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));
  BookHisto(new TH2D("RICHRingEPionRadiusvsCenterY","Single RICH Radius vs Y Center; Y Center [mm]; Ring Radius",400.,-500.,300.,400.,0.,220.));

  BookHisto(new TH1D("RICHBeamAngleDiffX","Angle Diff Beam RICH X",100,-0.02,0.02));
  BookHisto(new TH1D("RICHBeamAngleDiffY","Angle Diff Beam RICH Y",100,-0.02,0.02));
  BookHisto(new TH1D("RICHStrawCenterDiffX","X Center Diff Ring Predicted",100,-50.,50.));
  BookHisto(new TH1D("RICHStrawCenterDiffY","Y Center Diff Ring Predicted",100,-50.,50.));
  BookHisto(new TH1D("RICHStrawTimeDiff","Time Diff Ring Track",100,-20.,20.));

  BookHisto(new TH2D("RingRadiusvsRICHAngleMom","Single RICH Radius vs Momentum from Center All Beam; Momentum [MeV/c]; Ring Radius [mm]",400,50000.,80000.,400,0.,220.));
  BookHisto(new TH2D("RingRadiusvsRICHAngleMomKaon","Single RICH Radius vs Momentum from Center Kaon; Momentum [MeV/c]; Ring Radius [mm]",400,50000.,80000.,400,0.,220.));
  BookHisto(new TH2D("RingRadiusvsRICHAngleMomPion","Single RICH Radius vs Momentum from Center Pion; Momentum [MeV/c]; Ring Radius [mm]",400,50000.,80000.,400,0.,220.));
  BookHisto(new TH2D("RingRadiusvsRICHAngleMomExPion","Single RICH Radius vs Momentum from Center Extreme Pion; Momentum [MeV/c]; Ring Radius [mm]",400,50000.,80000.,400,0.,220.));

  BookHisto(new TH1D("DiffRadiusBeamRadiusRICHKaon","Diff Radius Beam RICH Kaon; Difference [mm]",100,-20.,30.));
  BookHisto(new TH1D("DiffRadiusBeamRadiusRICHPion","Diff Radius Beam RICH Pion; Difference [mm]",100,-20.,30.));
  BookHisto(new TH1D("DiffRadiusBeamRadiusRICHExPion","Diff Radius Beam RICH Extreme Pion; Difference [mm]",100,-20.,30.));

  BookHisto(new TH2D("DataElectronLAVAnglevsHadronAngleRICH","Data LAV Electron Angle vs Hadron Angle; Hadron Angle [rad]; Electron Angle [rad]",400.,0.,0.015,400.,0.,0.050));
  BookHisto(new TH1F("TimeDiffRingLAV","Time LAV - Time RICH Ring",100,-5.,5.));
  BookHisto(new TH1F("AngleDiffPlaneRICHLAV","Angle Scat Plane - Phi LAV",800,-2.*TMath::Pi(),2.*TMath::Pi()));
  BookHisto(new TH2D("TimeAngleDiffRingLAV","Phi Diff vs Time Diff;Track LAV Time Diff [ns];Scatter Plane LAV Phi Diff",100,-5.,5.,100.,-0.5,0.5));

  BookHisto(new TH1D("MCDataDiffElectronAngle","Difference MC reco Electron Angle;Angle Diff",100,-0.010,0.010));
  BookHisto(new TH1D("MCDataDiffHadronAngle","Difference MC reco Hadron Angle;Angle Diff",100,-0.010,0.010));

  BookHisto(new TH1D("MCDataBeamMomentum","Difference MC reco Beam Mom;Momentum Diff [MeV/c]",100,-2000.,2000.));
  BookHisto(new TH1D("MCDataHadronMomentum","Difference MC reco Hadron Mom;Momentum Diff [MeV/c]",100,-5000.,5000.));
  BookHisto(new TH2D("MCDataHadronMomentumvsMom","Difference MC reco Hadron Mom;MC Momentum [MeV/c]; Momentum Diff [MeV/c]",100,50000.,80000.,100,-5000.,5000.));
  BookHisto(new TH1D("MCDataBeamMomentumX","Difference MC reco Beam Mom X;Momentum Diff  [MeV/c]",100,-200.,200.));
  BookHisto(new TH1D("MCDataBeamMomentumY","Difference MC reco Beam Mom Y;Momentum Diff  [MeV/c]",100,-200.,200.));
  BookHisto(new TH1D("MCDataHadronAngleX","Difference MC reco Hadron Angle X;Angle Diff  [rad]",100,-0.010,0.010));
  BookHisto(new TH1D("MCDataHadronAngleY","Difference MC reco Hadron Angle Y;Angle Diff  [rad]",100,-0.010,0.010));
  BookHisto(new TH2D("MCDataHadronAngleXvsMom","Difference MC reco Hadron Angle X; Momentum [MeV/c]; Angle Diff [rad]",100,50000.,80000.,100,-0.002,0.002));

}

void Difraction2::DefineMCSimple(){


}

void Difraction2::StartOfRunUser(){

}
```

```
void Difraction2::StartOfBurstUser(){

  fNominalElectronRadius = RICHParameters::GetInstance()->GetElectronRingRadius();
  fFocalLength = GeometricAcceptance::GetInstance()->GetRICHMirrorFocalLength();

  fRefIndex = 1.0 / (cos(atan(fNominalElectronRadius / fFocalLength)));
}

void Difraction2::ProcessSpecialTriggerUser(int iEvent, unsigned int triggerType){

}

void Difraction2::Process(int iEvent){




  TRecoLAVEvent *LAVEvent = (TRecoLAVEvent*)GetEvent("LAV");

  TRecoRICHEvent *RICHEvent = (TRecoRICHEvent*)GetEvent("RICH");
  Int_t NRichHits = RICHEvent->GetNHits();

  TRecoLKrEvent* LKrEvent  = (TRecoLKrEvent*)GetEvent("LKr");
  Int_t NClusters = LKrEvent->GetNCandidates();

  TRecoCedarEvent* CedarEvent  = (TRecoCedarEvent*)GetEvent("Cedar");
  Int_t NCedarCand = CedarEvent->GetNCandidates();
  FillHisto("NCedarCand",NCedarCand);

  TRecoGigaTrackerEvent* GigaTrackerEvent  = (TRecoGigaTrackerEvent*)GetEvent("GigaTracker");
  Int_t NGigaTrackerCand = GigaTrackerEvent->GetNCandidates();
  FillHisto("NGTKCand",NGigaTrackerCand);

  TRecoSpectrometerEvent*  STRAWEvent  = GetEvent<TRecoSpectrometerEvent>();

  TRecoCHANTIEvent*    CHANTIEvent    = GetEvent<TRecoCHANTIEvent>();

  L0TPData* L0data = UserMethods::GetL0Data();
  if (!GetWithMC()) {   // Only check trigger for data
    //  UInt_t TriggerType  = L0data->GetDataType();
    //  Bool_t PhysicsTrigger = false;
    //  Bool_t ControlTrigger = false;
    //  if (TriggerType&0x1)  PhysicsTrigger = true;
    //  if (TriggerType&0x10) ControlTrigger = true;
    //  UInt_t TriggerFlags = L0data->GetTriggerFlags();
    Bool_t ControlTrigger = TriggerConditions::GetInstance()->IsControlTrigger(GetL0Data());
    //  if (!ControlTrigger) return;
  }
  Double_t TriggerTime = L0data->GetReferenceFineTime()*NA62ConditionsService::GetInstance()->GetTdcCalib();
  //  cout << "TriggerTime " << TriggerTime << endl;

  Int_t RunID = GetEventHeader()->GetRunID();
  Int_t EventID = GetEventHeader()->GetEventNumber();
  Int_t BurstID = GetEventHeader()->GetBurstID();

  Double_t GigaTrackerStationPositionZ[4];
  GigaTrackerStationPositionZ[0] = GeometricAcceptance::GetInstance()->GetZGTK0();
  GigaTrackerStationPositionZ[1] = GeometricAcceptance::GetInstance()->GetZGTK1();
  GigaTrackerStationPositionZ[2] = GeometricAcceptance::GetInstance()->GetZGTK2();
  GigaTrackerStationPositionZ[3] = GeometricAcceptance::GetInstance()->GetZGTK3();

  OutputState DownstreamState;
```

```
  std::vector<DownstreamTrack> DTracks = *(std::vector<DownstreamTrack>*)GetOutput("DownstreamTrackBuilder.Output",
DownstreamState);
  if (DownstreamState!=kOValid) return;

  FillHisto("DownstreamTracks",DTracks.size());

  TLorentzVector MCElectronMom,MCHadronMom,MCBeamMom;
  TLorentzVector MCElectronPos,MCHadronPos,MCBeamPos;
  Double_t Q2_MC,Q2B_MC,Q2C_MC,Q2D_MC,Q2E_MC,Q2F_MC,ElectronMomA_MC,
ElectronMomB_MC,Theta_elec_MC,Theta_hadron_MC;

  //  cout << "BeamKineTree Entries " << fBeamKineTree->GetEntries() << " NBranches " << fBeamKineTree->GetNbranches() <<
endl;
  //  if (GetWithMC() && std::string(GetCurrentFile()->GetName()).find("MuonHalo") == string::npos && std::string(GetCurrentFile()-
>GetName()).find("kmu2") == string::npos  && false) {   // Check true MC distributions
  if (GetWithMC() && std::string(GetCurrentFile()->GetName()).find("MuonHalo") == string::npos && std::string(GetCurrentFile()-
>GetName()).find("kmu2") == string::npos) {   // Check true MC distributions

    if (!fInputFileName.EqualTo(GetCurrentFile()->GetName())) {
      fInputFileName = GetCurrentFile()->GetName();
      if (finputFile) delete finputFile;
      finputFile = new TFile(fInputFileName.Data());
      fTreeEvent = -1;
      fBeamKineTree = (TTree*)finputFile->Get("BeamKine/1");
      fBeamKineTree->SetBranchAddress("x", &fX);
      fBeamKineTree->SetBranchAddress("y", &fY);
      fBeamKineTree->SetBranchAddress("z", &fZ);
      fBeamKineTree->SetBranchAddress("Px", &fPx);
      fBeamKineTree->SetBranchAddress("Py", &fPy);
      fBeamKineTree->SetBranchAddress("Pz", &fPz);
      fBeamKineTree->SetBranchAddress("t", &fT);
      fBeamKineTree->SetBranchAddress("PDGid", &fPDGid);
      fBeamKineTree->SetBranchAddress("ProperTime", &fProperTime);
      fBeamKineTree->SetBranchAddress("PathLength", &fPathLength);
      fBeamKineTree->SetBranchAddress("Weight", &fWeight);
      fBeamKineTree->SetBranchAddress("InitX", &fInitX);
      fBeamKineTree->SetBranchAddress("InitY", &fInitY);
      fBeamKineTree->SetBranchAddress("InitZ", &fInitZ);
      fBeamKineTree->SetBranchAddress("InitT", &fInitT);
      fBeamKineTree->SetBranchAddress("IsBeamParticle", &fIsBeamParticle);
    }
    fTreeEvent++;

    Event *evt = GetMCEvent();
    fBeamKineTree->GetEntry(fTreeEvent);
    Bool_t FoundElectron=false,FoundHadron=false,FoundBeam=false;

    FillHisto("MCBeamHadronElectron",0.);

    //    cout << "fX->size() " << fX->size() << " iEvent " << fTreeEvent <<endl;
    for (UInt_t i=0;i<fX->size();i++) {
      //      cout << " BeamKine " << i << " Mom " << fPx->at(i) << " " << fPy->at(i) << " " << fPz->at(i) << " PDG " << fPDGid->at(i) <<
" Pos " << fX->at(i) << " " << fY->at(i) << " " << fZ->at(i) << endl;
      if ((fPDGid->at(i)==321||fPDGid->at(i)==211||fPDGid->at(i)==2212) && (fInitZ->at(i)-GigaTrackerStationPositionZ[3])<-10000.) {
// Kaon/pion/proton before Difraction2
        Double_t Mass;
        if (fPDGid->at(i)==321) Mass = MKCH;  //  Beam is Kaon+
        else if (fPDGid->at(i)==211) Mass = MPI; //  Beam is pion+
        else if (fPDGid->at(i)==2212) Mass = MPROT; //  Beam is proton+
        MCBeamMom.SetXYZM(fPx->at(i),fPy->at(i),fPz->at(i),Mass);
        MCBeamPos.SetXYZT(fX->at(i),fY->at(i),fZ->at(i),fT->at(i));
        FoundBeam = true;
              FillHisto("MCBeamHadronElectron",1.);
      }
    }
```

```
   if (evt->GetNKineParts()) {

     for (Int_t i=0; i<evt->GetNKineParts(); i++) {
       //      cout << "Id " << i << " " << evt->GetKinePart(i)->GetMCTrackID() << " " << evt->GetKinePart(i)->GetCharge() << " " <<
evt->GetKinePart(i)->GetPDGcode() << " " << evt->GetKinePart(i)->GetParentIndex() << " " << evt->GetKinePart(i)-
>GetProdProcessName() << " " << evt->GetKinePart(i)->GetEndProcessName() << " " << (evt->GetKinePart(i)->GetProdPos()).Z() <<
" " << (evt->GetKinePart(i)->GetEndPos()).Z() << " " << evt->GetKinePart(i)->GetParentIndex() << endl;
       if (evt->GetKinePart(i)->GetParentIndex()==-1 && evt->GetKinePart(i)->GetPDGcode()==11) { // Electron from Difraction2
         MCElectronMom = evt->GetKinePart(i)->GetInitial4Momentum();
         MCElectronPos = evt->GetKinePart(i)->GetProdPos();
         FoundElectron = true;
         FillHisto("MCBeamHadronElectron",3.);
       }
             if (evt->GetKinePart(i)->GetParentIndex()==-1 && (evt->GetKinePart(i)->GetPDGcode()==321||evt->GetKinePart(i)-
>GetPDGcode()==211||evt->GetKinePart(i)->GetPDGcode()==2212) && fabs(evt->GetKinePart(i)->GetProdPos().Z()-
GigaTrackerStationPositionZ[3])<100.) { // Kaon from Difraction2
         MCHadronMom = evt->GetKinePart(i)->GetInitial4Momentum();
         MCHadronPos = evt->GetKinePart(i)->GetProdPos();
         FoundHadron = true;
         FillHisto("MCBeamHadronElectron",2.);
       }
       //       if (evt->GetKinePart(i)->GetParentIndex()==-1 && evt->GetKinePart(i)->GetPDGcode()==321 && (evt->GetKinePart(i)-
>GetProdPos().Z()-GigaTrackerStationPositionZ[3])<-100.) { // Kaon from Difraction2
       //       MCBeamMom = evt->GetKinePart(i)->GetInitial4Momentum();
       //       MCBeamPos = evt->GetKinePart(i)->GetProdPos();
       //       FoundBeam = true;
       //     }
     }  // loop over kineparts

   } // are there kineparts?

   if (!(FoundElectron&&FoundHadron&&FoundBeam)) return;  // bad MC event
   Q2_MC = 2.*MEL*(MCElectronMom.E()-MEL);
   Q2B_MC = 2.*MEL*(MCBeamMom.E()-MCHadronMom.E());
   Q2C_MC =
2.*MEL*MCBeamMom.P()*MCElectronMom.P()*cos((MCElectronMom.Vect()).Angle(MCBeamMom.Vect()))/(MEL+MCBeamMom
.E());
   Q2D_MC = 2.*MEL*MCBeamMom.P()*(MCBeamMom.P()-
MCHadronMom.P()*cos((MCHadronMom.Vect()).Angle(MCBeamMom.Vect())))/(MEL+MCBeamMom.E());

   ElectronMomA_MC = (MCBeamMom.P()-
MCHadronMom.P()*cos((MCHadronMom.Vect()).Angle(MCBeamMom.Vect())))/(cos((MCElectronMom.Vect()).Angle(MCBeamMo
m.Vect())));
   ElectronMomB_MC = (MCBeamMom.E()-
MCHadronMom.E())*(MCBeamMom.E()+MEL)/(MCBeamMom.P()*cos((MCElectronMom.Vect()).Angle(MCBeamMom.Vect())));

   Q2E_MC = 2.*MEL*(sqrt(ElectronMomA_MC*ElectronMomA_MC+MEL*MEL)-MEL);
   Q2F_MC = 2.*MEL*(sqrt(ElectronMomB_MC*ElectronMomB_MC+MEL*MEL)-MEL);

   Theta_elec_MC = (MCBeamMom.Vect()).Angle(MCElectronMom.Vect());
   Theta_hadron_MC = (MCBeamMom.Vect()).Angle(MCHadronMom.Vect());



   //    cout << "Q2 MC "  << Q2_MC << " " << Q2B_MC << " " << Q2C_MC << " " << Q2D_MC << " " << Theta_elec_MC << " Spat
Product " << ((MCBeamMom.Vect()).Cross(MCHadronMom.Vect())).Dot(MCElectronMom.Vect()) << " " <<
((MCBeamMom.Vect()).Cross(MCHadronMom.Vect())).Angle(MCElectronMom.Vect()) << endl;

   FillHisto("MCSpatProduct",((MCBeamMom.Vect()).Cross(MCHadronMom.Vect())).Dot(MCElectronMom.Vect()));
   FillHisto("MCAnglePlaneElectron",((MCBeamMom.Vect()).Cross(MCHadronMom.Vect())).Angle(MCElectronMom.Vect()));


   FillHisto("Q2AMCKaon",Q2_MC);
   FillHisto("Q2BMCKaon",Q2B_MC);
```

```
    FillHisto("Q2CMCKaon",Q2C_MC);
    FillHisto("Q2DMCKaon",Q2D_MC);
    FillHisto("Q2EMCKaon",Q2E_MC);
    FillHisto("Q2FMCKaon",Q2F_MC);

    FillHisto("Q2MCPion",Q2_MC);
    FillHisto("Q2MCProton",Q2_MC);

    FillHisto("Q2MCQ2BMC",Q2_MC-Q2B_MC);
    FillHisto("Q2MCQ2CMC",Q2_MC-Q2C_MC);
    FillHisto("Q2MCQ2DMC",Q2_MC-Q2D_MC);
    FillHisto("ElectronAngleMC",Theta_elec_MC);
    FillHisto("MCElectronAnglevsHadronAngle",Theta_hadron_MC,Theta_elec_MC);
    FillHisto("MCHadronMomDiffvsHadronAngle",Theta_hadron_MC,MCBeamMom.P()-MCHadronMom.P());

  } // end of MC code

  Int_t RICHnRings = RICHEvent->GetNRingCandidates();
  FillHisto("RICHnRings",RICHnRings);
  //  if (RICHnRings>0 && RICHnRings<3) {
  if (RICHnRings==1) {
    for (Int_t iring=0;iring<RICHnRings;iring++) {
      TRecoRICHCandidate *RingCandidate = RICHEvent->GetRingCandidate(iring);
      Double_t RingTime = RingCandidate->GetRingTime();
      //      Double_t RingTime = RingCandidate->GetAverageTime();
      Double_t radius  = RingCandidate->GetRingRadius();
      TVector2 center = RingCandidate->GetRingCenter();
      //      cout << "Single Ring " << RingTime << " Time " << RingCandidate->GetTime() << " Radius " << radius << " " << center.X()
<< " " << center.Y();
      Int_t TCI = RingCandidate->GetTimeCandidateIndex();
      //      cout << " Single Single Ring Time " << RingCandidate->GetRingTimeSingleRing();
      //      cout << " TriggerTime " << TriggerTime;
      //      cout << " RICH Hit Times " << NRichHits ;
      for(Int_t iHit = 0; iHit < NRichHits; ++iHit) {
        TRecoRICHHit *RecoHit = static_cast<TRecoRICHHit *>(RICHEvent->GetHit(iHit));
        if(!RecoHit->GetOrSuperCelllD()) {  // PM hit (not a supercell hit)
          Double_t hTime = RecoHit->GetTime();
          //      cout << " " << hTime;
                }
      }
      FillHisto("RICHRingRadiusvsCenterX",center.X(),radius);
      FillHisto("RICHRingRadiusvsCenterY",center.Y(),radius);

      // verify that that ring is NOT from a track which passed thru Straws
      Bool_t FoundOne=false;
      for (UInt_t it=0;it<DTracks.size();it++) {   //  loop over tracks
        if (!DTracks[it].RICHAssociationSuccessful() || !DTracks[it].RICHTimeExists()) continue;
        //      cout << " TrackTime " << DTracks[it].GetTrackTime() << " LikeliTime " << DTracks[it].GetRICHTime() << " NStations " <<
DTracks[it].GetNChambers();
        FillHisto("RICHStrawCenterDiffX",center.X()-DTracks[it].GetRICHRingPredictedCentrePosition().X());
        FillHisto("RICHStrawCenterDiffY",center.Y()-DTracks[it].GetRICHRingPredictedCentrePosition().Y());
        if (fabs(center.X()-DTracks[it].GetRICHRingPredictedCentrePosition().X())<10. &&
            fabs(center.Y()-DTracks[it].GetRICHRingPredictedCentrePosition().Y())<10. &&
            DTracks[it].CHODTimeExists())
                FillHisto("RICHStrawTimeDiff",RingTime-DTracks[it].GetCHODTime());
        if (fabs(center.X()-DTracks[it].GetRICHRingPredictedCentrePosition().X())<10. &&
            fabs(center.Y()-DTracks[it].GetRICHRingPredictedCentrePosition().Y())<10. &&
            fabs(RingTime-DTracks[it].GetCHODTime())<5.) FoundOne = true;
      }
      //      cout << endl;
      if (FoundOne) continue;

      std::vector<UInt_t> GoodGTK;
      Double_t Chi2=9999999999999.;
      Int_t iigtk=-1;
```

```
    for (Int_t igtk=0;igtk<GigaTrackerEvent->GetNCandidates();igtk++) {
      TRecoGigaTrackerCandidate* GTKCand = (TRecoGigaTrackerCandidate*) GigaTrackerEvent->GetCandidate(igtk);
      FillHisto("TimeDiffRingTimeGTK",GTKCand->GetTime()-RingTime);
            if ((GTKCand->GetType()==15 || GTKCand->GetType()==14) &&
         fabs(GTKCand->GetTime()-RingTime)<0.5) {
        GoodGTK.push_back(igtk);
        if (GTKCand->GetChi2()<Chi2) {
          iigtk = igtk;
          Chi2 = GTKCand->GetChi2();
        }
      }
      //      cout << "GTKMatch " << it << " " << igtk << " " << HadronIsPion << " " << HadronIsKaon << " " << HadronIsProton << "
Time " << TrackTime << " " << GTKCand->GetTime() << " Pos " << DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3])
<< " " << GTKCand->GetPosition(3).X() << " " << DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]) << " " <<
GTKCand->GetPosition(3).Y() << " " << " Prop " << GTKCand->GetType() << " " << GTKCand->GetChi2() << " " <<
GoodGTK.size() << endl;
    }
    FillHisto("GTKCandRingInTime",GoodGTK.size());
    //      if (GoodGTK.size()>0) {   // not much difference.  Nearly all have 0 or one GTK in time.
    if (GoodGTK.size()==1) {
      TRecoGigaTrackerCandidate* GTKCand = (TRecoGigaTrackerCandidate*) GigaTrackerEvent->GetCandidate(iigtk);   // use the
one with the best chi2

            FillHisto("RICHRingMatchedRadiusvsCenterX",center.X(),radius);
      FillHisto("RICHRingMatchedRadiusvsCenterY",center.Y(),radius);


      //  here we do some calcs trying to get the hadron angle compensating for the kick of MNP33
      Double_t MNP33Kick = -270.; //  MeV.  Should come from some DB.
      TVector3 BeamMomentum = GTKCand->GetMomentum();
      BeamMomentum.SetX(BeamMomentum.X()+MNP33Kick);
      TVector2 AngleRICH = center/fFocalLength;
      FillHisto("RICHBeamAngleDiffX",BeamMomentum.X()/BeamMomentum.Z()-AngleRICH.X());   // both distr are centered in 0 ->
calc is correct. x has long tail to pos diffs.
      FillHisto("RICHBeamAngleDiffY",BeamMomentum.Y()/BeamMomentum.Z()-AngleRICH.Y());

      // try to guess the momentum of the scattered hadron.  angle difference in horizontal plane due tto MNP33 field is inverse
proporcional to total momentum
      // angle diif in X corresponds to 75GeV (or better the momentum of this hadron).  Extra angle diff (<>0) means hadron has less.
      Double_t ExtraAngleX = BeamMomentum.X()/BeamMomentum.Z()-AngleRICH.X();
      Double_t ScatteredMom = BeamMomentum.X()/AngleRICH.X();
      FillHisto("RingRadiusvsRICHAngleMom",ScatteredMom,radius);

            if (GetWithMC()) {
        FillHisto("MCDataBeamMomentum",MCBeamMom.P()-GTKCand->GetMomentum().Mag());
        FillHisto("MCDataBeamMomentumX",MCBeamMom.Px()-GTKCand->GetMomentum().X());
        FillHisto("MCDataBeamMomentumY",MCBeamMom.Py()-GTKCand->GetMomentum().Y());
      }

      std::vector<UInt_t> GoodCedar;
      for (UInt_t icedar=0;icedar<NCedarCand;icedar++) {
        TRecoCedarCandidate* CedarCand = (TRecoCedarCandidate*) CedarEvent->GetCandidate(icedar);
        FillHisto("TimeDiffRingCedar",CedarCand->GetTime()-RingTime);
        if (fabs(CedarCand->GetTime()-GTKCand->GetTime())<0.5) GoodCedar.push_back(icedar);
              }
      FillHisto("CedarCandRingInTime",GoodCedar.size());
      if (GoodCedar.size()<2) {

        Bool_t BeamisKaon = false;
        Bool_t BeamisPion = true;
        Bool_t BeamisExtremePion = false;
        if (GoodCedar.size()==1) {
          TRecoCedarCandidate* CedarCand = (TRecoCedarCandidate*) CedarEvent->GetCandidate(GoodCedar[0]);
          if (CedarCand->GetNSectors()>5) BeamisKaon=true;
          if (CedarCand->GetNSectors()>2) BeamisPion=false;
```

```
      if (CedarCand->GetNSectors()<3) BeamisExtremePion=true;
     }

          Double_t RadiusBeam, DataHadronMom;
          if (BeamisKaon) {
           FillHisto("RICHRingKaonRadiusvsCenterX",center.X(),radius);
           FillHisto("RICHRingKaonRadiusvsCenterY",center.Y(),radius);
           FillHisto("RingRadiusvsRICHAngleMomKaon",ScatteredMom,radius);
           Double_t Beta = GTKCand->GetMomentum().Mag() / sqrt(GTKCand->GetMomentum().Mag()*GTKCand-
>GetMomentum().Mag() + MKCH*MKCH);
           Double_t ThetaC = acos(1.0 / fRefIndex / Beta);
           RadiusBeam = fFocalLength * ThetaC;
           FillHisto("DiffRadiusBeamRadiusRICHKaon",RadiusBeam-radius);
           Beta = 1./cos(radius/fFocalLength)/fRefIndex;
           DataHadronMom = MKCH * Beta / sqrt(1.-Beta*Beta);

          }

          if (BeamisPion) {
           FillHisto("RICHRingPionRadiusvsCenterX",center.X(),radius);
           FillHisto("RICHRingPionRadiusvsCenterY",center.Y(),radius);
           FillHisto("RingRadiusvsRICHAngleMomPion",ScatteredMom,radius);
           Double_t Beta = GTKCand->GetMomentum().Mag() / sqrt(GTKCand->GetMomentum().Mag()*GTKCand-
>GetMomentum().Mag() + MPI*MPI);
           Double_t ThetaC = acos(1.0 / fRefIndex / Beta);
           RadiusBeam = fFocalLength * ThetaC;
           FillHisto("DiffRadiusBeamRadiusRICHPion",RadiusBeam-radius);
           Beta = 1./cos(radius/fFocalLength)/fRefIndex;
           DataHadronMom = MPI * Beta / sqrt(1.-Beta*Beta);
          }

          if (BeamisExtremePion) {
           FillHisto("RICHRingEPionRadiusvsCenterX",center.X(),radius);
           FillHisto("RICHRingEPionRadiusvsCenterY",center.Y(),radius);
           FillHisto("RingRadiusvsRICHAngleMomExPion",ScatteredMom,radius);
           Double_t Beta = GTKCand->GetMomentum().Mag() / sqrt(GTKCand->GetMomentum().Mag()*GTKCand-
>GetMomentum().Mag() + MPI*MPI);
           Double_t ThetaC = acos(1.0 / fRefIndex / Beta);
           RadiusBeam = fFocalLength * ThetaC;
           FillHisto("DiffRadiusBeamRadiusRICHExPion",RadiusBeam-radius);
           Beta = 1./cos(radius/fFocalLength)/fRefIndex;
           DataHadronMom = MPI * Beta / sqrt(1.-Beta*Beta);
          }

          if (RadiusBeam-radius<0. || RadiusBeam-radius>15.) continue;     //  This cut comes from MC


          TVector3 HadronDirec;
          //              HadronDirec.SetZ(BeamMomentum.Mag());
          HadronDirec.SetZ(DataHadronMom);
          HadronDirec.SetX(AngleRICH.X()*HadronDirec.Z());
          HadronDirec.SetY(AngleRICH.Y()*HadronDirec.Z());
          HadronDirec.SetX(HadronDirec.X()-MNP33Kick);
          HadronDirec.Unit();

          if (GetWithMC()) {
           FillHisto("MCDataHadronMomentum",MCHadronMom.P()-DataHadronMom);
           FillHisto("MCDataHadronMomentumvsMom",MCHadronMom.P(),MCHadronMom.P()-DataHadronMom);
           FillHisto("MCDataHadronAngleX",MCHadronMom.Px()/MCHadronMom.Pz()-HadronDirec.Px()/HadronDirec.Pz());
           FillHisto("MCDataHadronAngleXvsMom",MCHadronMom.P(),MCHadronMom.Px()/MCHadronMom.Pz()-
HadronDirec.Px()/HadronDirec.Pz());
           FillHisto("MCDataHadronAngleY",MCHadronMom.Py()/MCHadronMom.Pz()-HadronDirec.Py()/HadronDirec.Pz());
          }
```

```
            Double_t RQ2A, RQ2B, RQ2C, RQ2D, RQ2E, RQ2F, RElectronMomA, RElectronMomB;

        //RQ2B = 2.*MEL*(BeamMom2.E()-HadronMomRICH2.E());
        //RQ2D = 2.*MEL*BeamMom2.P()*(BeamMom2.P()-
HadronMomRICH2.P()*cos((HadronMomRICH2.Vect()).Angle(BeamMom2.Vect())))/(MEL+BeamMom2.E());



            Double_t Theta_hadron_data = GTKCand->GetMomentum().Angle(HadronDirec);
            Double_t Phi_Plane = (GTKCand->GetMomentum().Cross(HadronDirec).Phi()) - TMath::Pi()/2.; // vector is now in
scattering plane
            if (Phi_Plane < -TMath::Pi()) Phi_Plane += 2.*TMath::Pi(); // Phi is between -pi and +pi.

            // LAV check:  Is there a hit from the electron in the LAV?
            Int_t GoodLAV = 0;
            //    std::cout << "LAV NCands " << LAVEvent->GetNCandidates() << std::endl;
            for (UInt_t il=0;il<LAVEvent->GetNCandidates();il++) {
              TRecoLAVCandidate* LAVCand = (TRecoLAVCandidate*)LAVEvent->GetCandidate(il);
              //      cout << il << " Time " << TrackTime << " " << LAVCand->GetTime() << " Phi Plane " << Phi_Plane << " " << " Pos
" << LAVCand->GetPosition().X() << " " << LAVCand->GetPosition().Y() << " " << LAVCand->GetPosition().Z() << " " <<
LAVCand->GetPosition().Phi() << " Energy " << LAVCand->GetEnergy() << " Type " << LAVCand->GetClusterType() << endl;
              TVector3 ElectronCand = LAVCand->GetPosition()-GTKCand->GetPosition(3);
              FillHisto("TimeDiffRingLAV",LAVCand->GetTime()-RingTime);
              if (fabs(LAVCand->GetTime()-RingTime)>4.) continue;
              FillHisto("DataElectronLAVAnglevsHadronAngleRICH",Theta_hadron_data,(GTKCand-
>GetMomentum()).Angle(ElectronCand));
              FillHisto("AngleDiffPlaneRICHLAV",Phi_Plane-LAVCand->GetPosition().Phi());
              FillHisto("TimeAngleDiffRingLAV",LAVCand->GetTime()-RingTime,Phi_Plane-LAVCand->GetPosition().Phi());
              if (
                  fabs(Phi_Plane-LAVCand->GetPosition().Phi()-0.087)<0.08 ||
                  fabs(Phi_Plane-LAVCand->GetPosition().Phi()+TMath::Pi()-0.087)<0.08 ||
                  fabs(Phi_Plane-LAVCand->GetPosition().Phi()-TMath::Pi()-0.087)<0.08
                  ) GoodLAV++; // for now

                RElectronMomA = (GTKCand->GetMomentum().Mag()-DataHadronMom*cos(GTKCand-
>GetMomentum().Angle(HadronDirec)))/(cos(GTKCand->GetMomentum().Angle(ElectronCand)));

          RQ2E = 2.*MEL*(sqrt(RElectronMomA*RElectronMomA+MEL*MEL)-MEL);

                  if (GetWithMC()) {
           FillHisto("MCDataDiffElectronAngle",(MCBeamMom.Vect()).Angle(MCElectronMom.Vect())-(GTKCand-
>GetMomentum().Angle(ElectronCand)));
            FillHisto("MCDataDiffHadronAngle",(MCHadronMom.Vect()).Angle(MCBeamMom.Vect())-(GTKCand-
>GetMomentum().Angle(HadronDirec)));
            }
              }

              FillHisto("GoodLAV",GoodLAV);
              if (GoodLAV>2) continue;
              //FillHisto("Q2AKaon",RQ2A);
        FillHisto("Q2BKaonRICH",RQ2B);
        //FillHisto("Q2CKaon",Q2C);
        FillHisto("Q2DKaonRICH",RQ2D);
        FillHisto("Q2EKaonRICH",RQ2E);
        FillHisto("Q2FKaonRICH",RQ2F);


            } // correct number of good cedar cand

    } // correct number of good gtk tracks

  } // end loop over single rings

 }  // end of single ring condition
```

```
  if (DTracks.size()<1) return;

  for (UInt_t it=0;it<DTracks.size();it++) {   //  big loop over tracks
    Bool_t HadronIsKaon=false,HadronIsPion=false,HadronIsProton=false;
    //    cout << "Track " << it << " "  << DTracks[it].GetMomentum() << " " << DTracks[it].RICHAssociationSuccessful() << " " <<
DTracks[it].GetRICHMostLikelyHypothesis() << endl;

    FillHisto("DTrackMomentum",DTracks[it].GetMomentum());
    if (DTracks[it].GetMomentum()<50000.) continue;
    if (DTracks[it].GetCharge() != +1) continue;
    if (DTracks[it].GetChi2() > 40.0) continue;
    FillHisto("NChambers",DTracks[it].GetNChambers());
    if (DTracks[it].GetNChambers() < 3) continue;
    if (!GeometricAcceptance::GetInstance()->InAcceptance(&DTracks[it], NA62::kMUV3)) continue;
    if (DTracks[it].MUV3AssociationExists()) continue;
    if (!GeometricAcceptance::GetInstance()->InAcceptance(&DTracks[it], NA62::kLKr)) continue;
    if (!DTracks[it].LKrAssociationExists()) continue;
    if (DTracks[it].GetLKrEoP()<0.1 || DTracks[it].GetLKrEoP()>0.9) continue;  // remove muons and electrons
    if (!GeometricAcceptance::GetInstance()->InAcceptance(&DTracks[it], NA62::kRICH)) continue;
    FillHisto ("BeamAxisCDA", DTracks[it].GetNominalBeamAxisCDA());
    if (DTracks[it].GetNominalBeamAxisCDA() > 60.0) continue;
    FillHisto("ZPosAllTracks",DTracks[it].GetBeamAxisVertex().Z());
    if (fabs((DTracks[it].GetNominalBeamAxisVertex()).Z()-GigaTrackerStationPositionZ[3])>2000.) continue;

FillHisto("GTK3Impact",DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),DTracks[it].yAtBeforeMagnet(GigaTrackerSta
tionPositionZ[3]));
    if (fabs((DTracks[it].GetNominalBeamAxisVertex()).X())>GeometricAcceptance::GetInstance()->GetGTKStationHalfWidthX())
continue;
    if (fabs((DTracks[it].GetNominalBeamAxisVertex()).Y())>GeometricAcceptance::GetInstance()->GetGTKStationHalfWidthY())
continue;


    Double_t TrackTime = -99999999999.;
    if (!DTracks[it].RICHAssociationSuccessful()) continue;
    if (DTracks[it].GetRICHRingPredictedNHits(NA62::kRICHHypothesisKaon)>2 &&
DTracks[it].GetRICHMostLikelyHypothesis()==NA62::kRICHHypothesisKaon) {
      FillHisto("HitsOnKaonRing",DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisKaon));

FillHisto("RatioHitsOnKaonRing",DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisKaon)/DTracks[it].GetRICHRingPredicted
NHits(NA62::kRICHHypothesisKaon));
      if (DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisKaon) > 2 &&
DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisKaon)/DTracks[it].GetRICHRingPredictedNHits(NA62::kRICHHypothesisK
aon) > 0.4) {
        HadronIsKaon = true;
        if (DTracks[it].RICHRingTimeExists(NA62::kRICHHypothesisKaon)) TrackTime =
DTracks[it].GetRICHRingTime(NA62::kRICHHypothesisKaon);
        else if (DTracks[it].CHODTimeExists()) TrackTime = DTracks[it].GetCHODTime();
        else TrackTime = DTracks[it].GetTrackTime();
      }
    }
    //    else if (DTracks[it].GetRICHMostLikelyHypothesis()==NA62::kRICHHypothesisPion) {
    else if (DTracks[it].GetRICHLikelihoodPion()>0.95) {  // possible overlap with muon and electron hypo
      FillHisto("HitsOnPionRing",DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisPion));

FillHisto("RatioHitsOnPionRing",DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisPion)/DTracks[it].GetRICHRingPredictedN
Hits(NA62::kRICHHypothesisPion));
      if (DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisPion) > 2 &&
DTracks[it].GetRICHRingNHits(NA62::kRICHHypothesisPion)/DTracks[it].GetRICHRingPredictedNHits(NA62::kRICHHypothesisPio
n) > 0.4) {
        HadronIsPion = true;
        if (DTracks[it].RICHRingTimeExists(NA62::kRICHHypothesisPion)) TrackTime =
DTracks[it].GetRICHRingTime(NA62::kRICHHypothesisPion);
        else if (DTracks[it].CHODTimeExists()) TrackTime = DTracks[it].GetCHODTime();
              else TrackTime = DTracks[it].GetTrackTime();
```

```
    }
   }
   else if (DTracks[it].GetRICHMostLikelyHypothesis()==NA62::kRICHHypothesisBackground) {
    HadronIsProton = true;
    if (DTracks[it].CHODTimeExists()) TrackTime = DTracks[it].GetCHODTime();
    else TrackTime = DTracks[it].GetTrackTime();
   }
   if (!HadronIsKaon && !HadronIsPion && !HadronIsProton) continue;

   Double_t HadronGamma = 1.;
   if (HadronIsKaon || HadronIsPion) {
    Double_t HadronBeta = (2.*fFocalLength*fFocalLength - fNominalElectronRadius*fNominalElectronRadius)/
     (2.*fFocalLength*fFocalLength - DTracks[it].GetRICHRingRadius()*DTracks[it].GetRICHRingRadius());
    HadronGamma = 1./sqrt(1.-HadronBeta*HadronBeta);
   }

   TLorentzVector HadronMom,HadronMomRICH;
   if (HadronIsKaon) {
    HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MKCH);
    HadronMomRICH.SetVectM((DTracks[it].GetMomentumBeforeMagnet().Unit())*MKCH*sqrt(HadronGamma*HadronGamma-
1.),MKCH);
    FillHisto("MomKaonSelected",DTracks[it].GetMomentum());
    FillHisto("RICHMomKaonSelected",HadronMomRICH.Vect().Mag());
    FillHisto("ZPosKaonSelected",DTracks[it].GetBeamAxisVertex().Z());
    FillHisto("AngleKaonSelected",(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    FillHisto("AngleMomKaonSelected",DTracks[it].GetMomentum(),(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    FillHisto("AngleRICHMomKaonSelected",HadronMomRICH.Vect().Mag(),(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
   }
   if (HadronIsPion) {
    HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MPI);
    HadronMomRICH.SetVectM((DTracks[it].GetMomentumBeforeMagnet().Unit())*MPI*sqrt(HadronGamma*HadronGamma-
1.),MPI);
    FillHisto("MomPionSelected",DTracks[it].GetMomentum());
    FillHisto("RICHMomPionSelected",HadronMomRICH.Vect().Mag());
    FillHisto("ZPosPionSelected",DTracks[it].GetBeamAxisVertex().Z());
    FillHisto("AnglePionSelected",(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    FillHisto("AngleMomPionSelected",DTracks[it].GetMomentum(),(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
   }
   if (HadronIsProton) {
    HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MPROT);
    HadronMomRICH = HadronMom;
    FillHisto("MomProtonSelected",DTracks[it].GetMomentum());
    FillHisto("ZPosProtonSelected",DTracks[it].GetBeamAxisVertex().Z());
    FillHisto("AngleProtonSelected",(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    FillHisto("AngleMomProtonSelected",DTracks[it].GetMomentum(),(BeamParameters::GetInstance()-
>GetBeamThreeMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
   }


   /*
   // try some sophisticated matching
   std::vector<TVector3>  matchedGTKMomenta;
   std::vector<int> matchedGTKIDs;
   std::vector<double> matchedGTKTimes;
   std::vector<TVector3> matchedVertices;
   TRecoSpectrometerCandidate *STRAWCand = static_cast<TRecoSpectrometerCandidate*>(STRAWEvent-
>GetCandidate(GoodTrack[0]));
   Double_t tKTAG = TrackTime;
   fMatchingRG->Process(GigaTrackerEvent, STRAWCand, tKTAG, tKTAG, TrackTime, 1, "");
```

```
    fMatchingRG->FinalSelection(tKTAG, TrackTime, 1, "");
    matchedGTKIDs = fMatchingRG->GetMatchedGTKIDs();
    if (matchedGTKIDs.at(0)==-1) return; // GTK matching failed
    //    if (matchedGTKIDs.size()>1) continue; // we only want one match.  If commented, we get the "best" match
    matchedGTKMomenta = fMatchingRG->GetGTKMomentaAtVertices();
    matchedVertices = fMatchingRG->GetVertices();
    FillHisto("ZPosKaonMatched",(matchedVertices.at(0)).Z());
    if (fabs((matchedVertices.at(0)).Z()-GigaTrackerStationPositionZ[3])>600*2.5) return; // sigma=600, 2.5sigma
    */

    std::vector<UInt_t> GoodGTK;
    FillHisto("GTKCands",NGigaTrackerCand);
    Double_t R2;
    Double_t Chi2=9999999999999.;
    Int_t iigtk=-1;
    for (Int_t igtk=0;igtk<GigaTrackerEvent->GetNCandidates();igtk++) {
      TRecoGigaTrackerCandidate* GTKCand = (TRecoGigaTrackerCandidate*) GigaTrackerEvent->GetCandidate(igtk);
      FillHisto("TimeDiffTrackGTK",GTKCand->GetTime()-TrackTime);
      FillHisto("PosDiffTrackGTK3",GTKCand->GetPosition(3).X()-
DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),GTKCand->GetPosition(3).Y()-
DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]));
      R2 = TMath::Power(GTKCand->GetPosition(3).X()-DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),2) +
TMath::Power(GTKCand->GetPosition(3).Y()-DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]),2);
      FillHisto("R2TimeDiff",GTKCand->GetTime()-TrackTime,R2);
      if((GTKCand->GetType()==15 || GTKCand->GetType()==14) &&
         fabs(GTKCand->GetTime()-TrackTime)<0.5 &&
         fabs(DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3])-GTKCand->GetPosition(3).X())<3.0 &&
         fabs(DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3])-GTKCand->GetPosition(3).Y())<3.0) {
        GoodGTK.push_back(igtk);
        if (GTKCand->GetChi2()<Chi2) {
          iigtk = igtk;
          Chi2 = GTKCand->GetChi2();
        }
      }
      //      cout << "GTKMatch " << it << " " << igtk << " " << HadronIsPion << " " << HadronIsKaon << " " << HadronIsProton << "
Time " << TrackTime << " " << GTKCand->GetTime() << " Pos " << DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3])
<< " " << GTKCand->GetPosition(3).X() << " " << DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]) << " " <<
GTKCand->GetPosition(3).Y() << " " << " Prop " << GTKCand->GetType() << " " << GTKCand->GetChi2() << " " <<
GoodGTK.size() << endl;
    }
    FillHisto("GTKCandInTime",GoodGTK.size());
    if (GoodGTK.size()<1) continue;
    TRecoGigaTrackerCandidate* GTKCand = (TRecoGigaTrackerCandidate*) GigaTrackerEvent->GetCandidate(iigtk);

    FillHisto("MomDiff",DTracks[it].GetMomentum()-GTKCand->GetMomentum().Mag());
    FillHisto("RICHMomDiff",HadronMomRICH.Vect().Mag()-GTKCand->GetMomentum().Mag());

    if (HadronIsKaon) {
      FillHisto("RICHMomDiffKaon",HadronMomRICH.Vect().Mag()-GTKCand->GetMomentum().Mag());
      FillHisto("AngleKaonSelectedMatch",(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
      FillHisto("AngleMomDiffKaonSelectedMatch",DTracks[it].GetMomentum()-GTKCand->GetMomentum().Mag(),(GTKCand-
>GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
      FillHisto("AngleMomDiffRICHKaonSelectedMatch",HadronMomRICH.Vect().Mag()-GTKCand-
>GetMomentum().Mag(),(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    }
    if (HadronIsPion) {
      FillHisto("RICHMomDiffPion",HadronMomRICH.Vect().Mag()-GTKCand->GetMomentum().Mag());
      FillHisto("AnglePionSelectedMatch",(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
      FillHisto("AngleMomDiffPionSelectedMatch",DTracks[it].GetMomentum()-GTKCand->GetMomentum().Mag(),(GTKCand-
>GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
      FillHisto("AngleMomDiffRICHPionSelectedMatch",HadronMomRICH.Vect().Mag()-GTKCand-
>GetMomentum().Mag(),(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
    }
    if (HadronIsProton) {
      FillHisto("AngleProtonSelectedMatch",(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
```

```
    FillHisto("AngleMomDiffProtonSelectedMatch",DTracks[it].GetMomentum()-GTKCand->GetMomentum().Mag(),(GTKCand->GetMomentum()).Angle(DTracks[it].GetMomentumBeforeMagnet()));
   }


   FillHisto("TimeDiffTrackGTKAfter",GTKCand->GetTime()-TrackTime);
   FillHisto("PosDiffTrackGTK3After",GTKCand->GetPosition(3).X()-DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),GTKCand->GetPosition(3).Y()-DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]));
   R2 = TMath::Power(GTKCand->GetPosition(3).X()-DTracks[it].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),2) + TMath::Power(GTKCand->GetPosition(3).Y()-DTracks[it].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]),2);
   FillHisto("R2TimeDiffAfter",GTKCand->GetTime()-TrackTime,R2);

   std::vector<UInt_t> GoodCedar;
   FillHisto("CedarCands",NCedarCand);
   for (UInt_t icedar=0;icedar<NCedarCand;icedar++) {
    TRecoCedarCandidate* CedarCand = (TRecoCedarCandidate*) CedarEvent->GetCandidate(icedar);
    FillHisto("TimeDiffGTrackCedar",CedarCand->GetTime()-GTKCand->GetTime());
    FillHisto("TimeDiffTrackCedar",CedarCand->GetTime()-TrackTime);
    if (fabs(CedarCand->GetTime()-GTKCand->GetTime())<0.5) GoodCedar.push_back(icedar);
   }
   FillHisto("CedarCandInTime",GoodCedar.size());
   if (GoodCedar.size()>1) continue;

   Bool_t BeamisKaon = false;
   Bool_t BeamisPion = true;
   Bool_t BeamisExtremePion = false;
   if (GoodCedar.size()==1) {
    TRecoCedarCandidate* CedarCand = (TRecoCedarCandidate*) CedarEvent->GetCandidate(GoodCedar[0]);
    if (CedarCand->GetNSectors()>5) BeamisKaon=true;
    if (CedarCand->GetNSectors()>2) BeamisPion=false;
    if (CedarCand->GetNSectors()<3) BeamisExtremePion=true;
   }


   Bool_t ExtraGTKHits=false;
   Int_t nGTKHitInTime[5]={0,0,0,0,0};
   for (Int_t ih=0;ih<GigaTrackerEvent->GetNHits();ih++) {
    TRecoGigaTrackerHit* GTKHit = (TRecoGigaTrackerHit*) GigaTrackerEvent->GetHit(ih);
    if (fabs(GTKHit->GetTime()-GTKCand->GetTime())<1.0) {
     nGTKHitInTime[4]++;
     for (Int_t igtk=0;igtk<=3;igtk++) {
      if (fabs((GTKHit->GetPosition()).Z()-GigaTrackerStationPositionZ[igtk])<100.) nGTKHitInTime[igtk]++;
     }
    }
   }
   FillHisto("GTKHitInTime",nGTKHitInTime[4]);
   if ((nGTKHitInTime[1]>1&&nGTKHitInTime[2]>1) || (nGTKHitInTime[1]>1&&nGTKHitInTime[3]>1) || (nGTKHitInTime[2]>1&&nGTKHitInTime[3]>1) || (nGTKHitInTime[0]>1&&nGTKHitInTime[1]>1) || (nGTKHitInTime[0]>1&&nGTKHitInTime[2]>1) || (nGTKHitInTime[0]>1&&nGTKHitInTime[3]>1)) {
    ExtraGTKHits = true;
   }


   TLorentzVector BeamMom;
   if (BeamisKaon) {
    BeamMom.SetVectM(GTKCand->GetMomentum(),MKCH);
    HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MKCH);
   }
   else if (BeamisPion) {
    if (HadronIsPion) {
     BeamMom.SetVectM(GTKCand->GetMomentum(),MPI);
     HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MPI);
    }
    if (HadronIsProton) {
```

```
      BeamMom.SetVectM(GTKCand->GetMomentum(),MPROT);
      HadronMom.SetVectM(DTracks[it].GetMomentumBeforeMagnet(),MPROT);
     }
    }
    else continue;

    Int_t ExtraCHANTIHit=0;
    for (Int_t i=0;i<CHANTIEvent->GetNHits();i++) {
     TRecoCHANTIHit* Hit = (TRecoCHANTIHit*) CHANTIEvent->GetHit(i);
     FillHisto("TimeDiffCHANTITrack",Hit->GetTime()-TrackTime);
     if (fabs(Hit->GetTime()-TrackTime)<3.) ExtraCHANTIHit++;
    }
    if (ExtraCHANTIHit>1) continue;

    Double_t Phi_Plane = (GTKCand->GetMomentum().Cross(DTracks[it].GetMomentumBeforeMagnet())).Phi()-TMath::Pi()/2.;
    if (Phi_Plane < -TMath::Pi()) Phi_Plane = 2.*TMath::Pi() + Phi_Plane;


    Double_t Q2B, Q2D, Q2E, Q2F, ElectronMomA, ElectronMomB;

    Q2B = 2.*MEL*(BeamMom.E()-HadronMom.E());
    Q2D = 2.*MEL*BeamMom.P()*(BeamMom.P()-
HadronMom.P()*cos((HadronMom.Vect()).Angle(BeamMom.Vect())))/(MEL+BeamMom.E());
    //  cout << "Beam " << BeamisKaon << " " << BeamisPion << " " << BeamMom.E() << " " << BeamMom.P() << " " <<
BeamMom.M() << " Hadron " << HadronIsKaon << " " << HadronIsPion << " " << HadronIsProton << " " << HadronMom.E() << " "
<< HadronMom.P() << " " << HadronMom.M() << " Q2 " << Q2B << " " << Q2D << endl;
    Double_t Theta_elec_data_expected =
acos((MEL+BeamMom.E())*Q2B/2./MEL/BeamMom.P()/sqrt(Q2B*Q2B/4./MEL/MEL+Q2B));
    Double_t E_elec_data_expected = Q2B/(2.*MEL)+MEL;

    Double_t Theta_hadron_data = (BeamMom.Vect()).Angle(HadronMom.Vect());


    Int_t ExtraTrack=0;
    Int_t GoodElectronTrack = -1;
    for (UInt_t it2=0;it2<DTracks.size();it2++) {
     if (it==it2) continue;
     if (fabs(DTracks[it2].GetTrackTime()-TrackTime)<5.) {
      ExtraTrack++;
      FillHisto("ZPosExtraTracks",DTracks[it2].GetBeamAxisVertex().Z());
      if (fabs(DTracks[it2].GetBeamAxisVertex().Z()-GigaTrackerStationPositionZ[3]) < 2000.) {
       FillHisto("PosDiffExtraTrackGTK3",GTKCand->GetPosition(3).X()-
DTracks[it2].xAtBeforeMagnet(GigaTrackerStationPositionZ[3]),GTKCand->GetPosition(3).Y()-
DTracks[it2].yAtBeforeMagnet(GigaTrackerStationPositionZ[3]));
       FillHisto("MomExtraTrack",DTracks[it2].GetMomentum());
       FillHisto("DiffMomExtraTrack",DTracks[it2].GetMomentum()-E_elec_data_expected);

FillHisto("DataElectronTrackAnglevsHadronAngle",Theta_hadron_data,(BeamMom.Vect()).Angle(DTracks[it2].GetMomentumBefore
Magnet()));
       if (fabs(DTracks[it2].GetMomentum()-E_elec_data_expected)<1500.) {
        GoodElectronTrack = it2;
        ExtraTrack--;
       }
      }
     }

    }
    if (ExtraTrack>0) continue;

    // LAV check:  Is there a hit from the electron in the LAV?
    Int_t GoodLAV = 0;
    //    std::cout << "LAV NCands " << LAVEvent->GetNCandidates() << std::endl;
    for (UInt_t il=0;il<LAVEvent->GetNCandidates();il++) {
     TRecoLAVCandidate* LAVCand = (TRecoLAVCandidate*)LAVEvent->GetCandidate(il);
```

```
    //      cout << il << " Time " << TrackTime << " " << LAVCand->GetTime() << " Phi Plane " << Phi_Plane << " " << " Pos " <<
LAVCand->GetPosition().X() << " " << LAVCand->GetPosition().Y() << " " << LAVCand->GetPosition().Z() << " " << LAVCand-
>GetPosition().Phi() << " Energy " << LAVCand->GetEnergy() << " Type " << LAVCand->GetClusterType() << endl;
    TVector3 ElectronCand = LAVCand->GetPosition()-GTKCand->GetPosition(3);
    FillHisto("TimeDiffTrackLAV",LAVCand->GetTime()-TrackTime);
    if (fabs(LAVCand->GetTime()-TrackTime+0.2)>4.) continue;
    FillHisto("DataElectronLAVAnglevsHadronAngle",Theta_hadron_data,(BeamMom.Vect()).Angle(ElectronCand));
    FillHisto("AngleDiffPlaneLAV",Phi_Plane-LAVCand->GetPosition().Phi());
    FillHisto("TimeAngleDiffTrackLAV",LAVCand->GetTime()-TrackTime,Phi_Plane-LAVCand->GetPosition().Phi());
    if (
       fabs(Phi_Plane-LAVCand->GetPosition().Phi()-0.087)<0.08 ||
       fabs(Phi_Plane-LAVCand->GetPosition().Phi()+TMath::Pi()-0.087)<0.08 ||
       fabs(Phi_Plane-LAVCand->GetPosition().Phi()-TMath::Pi()-0.087)<0.08
       ) GoodLAV++;; // for now

    ElectronMomA = (BeamMom.P()-
HadronMom.P()*cos((HadronMom.Vect()).Angle(BeamMom.Vect())))/(cos((BeamMom.Vect()).Angle(ElectronCand)));
    ElectronMomB = (BeamMom.E()-
HadronMom.E())*(BeamMom.E()+MEL)/(BeamMom.P()*cos(BeamMom.Vect().Angle(ElectronCand)));

    Q2E = 2.*MEL*(sqrt(ElectronMomA*ElectronMomA+MEL*MEL)-MEL);
    Q2F = 2.*MEL*(sqrt(ElectronMomB*ElectronMomB+MEL*MEL)-MEL);


  }
  FillHisto("GoodLAV",GoodLAV);
  if (GoodLAV>2) continue;

  // multi-track segments evaluation
  TVector3 position,direction;
  Bool_t GoodSegment = false;
  /*
  std::vector<Int_t> trackIDs = {(int)it};
  if (GoodElectronTrack > -1) trackIDs.push_back(GoodElectronTrack);
  fSegmentsAlgo->Process(GetEvent<TRecoSpectrometerEvent>(), trackIDs, GTKCand->GetPosition(3),TrackTime);
  fIsSegments = fSegmentsAlgo->GetIsSegment();
  //  cout << "Segments " << fIsSegments << endl;
  if (fIsSegments) {
   const std::vector<StrawSegmentAlgorithm::TrackSegment>& segments_before_magnet =
     fSegmentsAlgo->GetTrackSegmentsBeforeMagnet();
   for (const auto & segment : segments_before_magnet) {
    position = segment.GetPosition();
    direction = segment.GetDirection();
    //      cout << " Pos " << position.X() << " " << position.Y() << " " << position.Z() << " Dir " << direction.X() << " " <<
direction.Y() <<  " " << direction.Z() << endl;
    }
    if (segments_before_magnet.size() == 1) {
     FillHisto("ElectronAngleSegmant",(BeamMom.Vect()).Angle(direction));
     Double_t Theta_elec_data = (BeamMom.Vect()).Angle(direction);
     if (GetWithMC()) {
      FillHisto("ElectronAngleMC",Theta_elec_MC);
      FillHisto("ElectronAngleDiffMC",Theta_elec_MC-Theta_elec_data);
     }
     FillHisto("ElectronAngleData",Theta_elec_data);
          FillHisto("DataElectronSegmentAnglevsHadronAngle",Theta_hadron_data,Theta_elec_data);
     FillHisto("DataHadronMomDiffvsHadronAngle",Theta_hadron_data,BeamMom.P()-HadronMom.P());
     FillHisto("ElectronAngleDiffData",Theta_elec_data_expected-Theta_elec_data);
     if (fabs(Theta_elec_data_expected-Theta_elec_data)<0.002) GoodSegment = true;
    }
    if (!GoodSegment) continue;
  }
  */

  if ((GoodLAV>0) && GoodSegment && (GoodElectronTrack>-1)) FillHisto("ElectronSegment",7.);
  else if ((GoodLAV>0) && GoodSegment) FillHisto("ElectronSegment",3.);
```

```
   else if ((GoodLAV>0) && (GoodElectronTrack>-1)) FillHisto("ElectronSegment",5.);
   else if (GoodSegment && (GoodElectronTrack>-1)) FillHisto("ElectronSegment",6.);
   else if (GoodLAV>0) FillHisto("ElectronSegment",1.);
   else if (GoodSegment) FillHisto("ElectronSegment",2.);
   else if (GoodElectronTrack>-1) FillHisto("ElectronSegment",4.);
   else FillHisto("ElectronSegment",0.);
   if ((GoodLAV==0) && !GoodSegment && (GoodElectronTrack==-1)) continue;  // It has to be one...
   if ( ((GoodLAV>0) && GoodSegment) || ((GoodLAV>0) && (GoodElectronTrack>-1)) || (GoodSegment && (GoodElectronTrack>-
1)) || ((GoodLAV>0) && GoodSegment && (GoodElectronTrack>-1))) continue;    // but only one...

   if (BeamisKaon && HadronIsKaon) {
    FillHisto("Q2BKaon",Q2B);
    FillHisto("Q2DKaon",Q2D);
    FillHisto("Q2EKaon",Q2E);
    FillHisto("Q2FKaon",Q2F);
    if (GoodElectronTrack>0) FillHisto("Q2BKaonT",Q2B);
    if (GoodSegment) FillHisto("Q2BKaonS",Q2B);
    if (GoodLAV>0) FillHisto("Q2BKaonL",Q2B);
   }
   if (BeamisPion) {
    if (HadronIsPion) {
     FillHisto("Q2BPion",Q2B);
     FillHisto("Q2DPion",Q2D);
    }
    if (HadronIsProton) {
     FillHisto("Q2BProton",Q2B);
     FillHisto("Q2DProton",Q2D);
    }
   }
   //  cout << "Q2 Data " << Q2_MC << " " << Q2B << " " << Q2D << " Electron Angle Exp " << Theta_elec_MC << " " <<
Theta_elec_data_expected << " Meas " << Theta_elec_data << endl;

  }  //   loop over tracks
}


void Difraction2::PostProcess(){

  //  if (GigaTrackerEvent) delete GigaTrackerEvent;

}

void Difraction2::EndOfBurstUser(){
        /// \MemberDescr
        /// This method is called when a new file is opened in the ROOT TChain (corresponding to a start/end of burst in the normal
NA62 data taking) + at the end of the last file\n
        /// Do here your start/end of burst processing if any.
        /// Be careful: this is called after the event/file has changed.
        /// \EndMemberDescr
}

void Difraction2::EndOfRunUser(){
        /// \MemberDescr
        /// This method is called at the end of the processing (corresponding to a end of run in the normal NA62 data taking)\n
        /// Do here your end of run processing if any\n
        /// \EndMemberDescr

}

void Difraction2::EndOfJobUser(){
  //  fSegmentsAlgo->SaveAllPlots();
  SaveAllPlots();
}

void Difraction2::DrawPlot(){
```

```
	/// \MemberDescr
	/// This method is called at the end of processing to draw plots when the -g option is used.\n
	/// If you want to draw all the plots, just call\n
	/// \code
	///		DrawAllPlots();
	/// \endcode
	/// Or get the pointer to the histogram with\n
	/// \code
	///		fHisto.GetTH1("histoName");// for TH1
	///		fHisto.GetTH2("histoName");// for TH2
	///		fHisto.GetGraph("graphName");// for TGraph and TGraphAsymmErrors
	///	fHisto.GetHisto("histoName");// for TH1 or TH2 (returns a TH1 pointer)
	/// \endcode
	/// and manipulate it as usual (TCanvas, Draw, ...)\n
	/// \EndMemberDescr
}

Difraction2::~Difraction2(){
	/// \MemberDescr
	/// Destructor of the Analyzer. If you allocated any memory for class
	/// members, delete them here.
	/// \EndMemberDescr
}
```

Bibliografía:

[1]. ROOT-CERN Tbinomialefficiencyfitter class reference, 2021.
[2]. J. B. Bellicard et al.: Phys. Rev. Lett. 19 (1967) 527
[3]. R. Hofstadter: Ann. Rev. Nucl. Sci. 7 (1957) 231
[4]. Eschrich, I. 1998. Measurement of the Σ^- Charge Radius at the Fermilab Hyperon Beam. Inaugural Dissertation, Ruprecht – Karls University
[5]. S. R. Amendolia et al.: Phys. Lett. B178 (1986) 435
[6]. S. R. Amendolia et al.: Phys. Lett. B146 (1984) 116.
[7]. Vorwalter, K. 1998. Determination of the Pion Charge Radius with a Silicon Microstrip Detector System. Inaugural Dissertation, Ruprecht – Karls University.
[8]. E. B. Hughes et al.: Phys. Rev. B139 (1965) 458.
[9]. R. E. Taylor: Proc. Int. Symp. on Electron and Photon Interactions at High Energies (Stanford 1967).
[10]. F. Borkowski et al.: Nucl. Phys. B93 (1975) 461.
[11]. E. Cortina Gil, E. Martín Albarrán, E. Minucci, et al, NA62 collaboration. The beam and detector of the NA62 experiment at CERN. Journal of Instrumentation, 12(05):P05025–P05025, May 2017.
[12]. M. N. Rosenbluth: Phys. Rev. 79 (1950) 615.
[13]. Yorikiyo Nagashima. Elementary Particle Physics. Wiley-Vch, 2010.
[14]. Alan D. Martin Francis Halzen. QUARKS AND LEPTONS: An Introductory Course in Modern Particle Physics. JOHN and SONS, 1991.
[15]. T. Cai, M. L. Moore, A. Oliver, etal. (2022) Measurement of the axial vector form factor from antineutrino-proton scattering. Nature 614-48
[16]. CERN. The standard model, https://home.cern/science/physics/standard-model, 2021.
[17]. CERN. The Higgs boson, https://home.cern/science/physics/higgs-boson, 2021.
[18]. Douglas Ross Alexander S Belyaev. The Basics of Nuclear and Particle Physics. Springer Nature, 2021.
[19]. David Griffiths. Introduction to Elementary particles. Wiley-VCH, 2004.
[20]. Christoph Scholz Frank Zetsche Werner Rodejohann Bodgan Povh, Klaus Rith. Particles and Nuclei. Springer, 2014.
[21]. Ulrich E. Schröder. Special Relativity. World Scientific, 1990.
[22]. Graham Shaw Brian R. Martin. Particle Physics. John Wiley and Sons, 2017.
[23]. Francis, Naukas, La confirmación definitiva de la solución del problema del radio de carga del protón https://francis.naukas.com/2020/12/01/la-confirmacion-definitiva-de-la-solucion-del-problema-del-radio-de-carga-del-proton/
[24]. Alexey Grinin, Arthur Matveev, …, Thomas Udem, Two-photon frequency comb spectroscopy of atomic hydrogen, Science 370: 1061–1066 (27 Nov 2020)
[25]. SELEX Collaboration, Ivo M. Gough Eschrich, 1998. Measurement of the Σ^- charge radius at SELEX.
[26]. Reichert, S. El tamaño del protón. *Nat. Phys.* **15**, 306 (2019).
[27]. Ahmadi, M., Alves, B.X.R., Baker, C.J. et al. Characterization of the 1S–2S transition in antihydrogen. Nature 557, 71–75 (2018).
[28]. Aldo Antognini et al., «Proton Structure from the Measurement of 2S-2P Transition Frequencies of Muonic Hydrogen,» Science 339: 417-420, 25 Jan 2013
[29]. W. Xiong, A. Gasparian, …, Z. W. Zhao, «A small proton charge radius from an electron–proton scattering experiment,» Nature 575: 147-150 (06 Nov 2019)

[30]. Agradecimientos:

Agradezco principalmente a mi asesor y director de tesis **Dr. Jürgen Engelfried**, por aceptarme y recibirme en su grupo de trabajo, formando parte así del experimento NA62. Agradezco profundamente su paciencia y dedicación en este trabajo, alentándome a seguir adelante con proyectos futuros. El conocimiento que he recibido de él a lo largo de mi carrera universitaria y en el proceso de esta investigación me ayudo a decidir mi futuro profesional como físico.

A **M. C. Nora Estrada** por su tiempo y consejos en el tiempo que duro este proceso, por su apoyo y aceptación como un miembro más del equipo potosino del NA62.

A **I. E. Luz del Carmen Nuche Garza** técnico del laboratorio de física de altas energías del instituto de física de la UASLP, por su recibimiento y confianza en el área de trabajo del laboratorio.

A **L. E. S. D. José Limón Castillo** del centro de computación del instituto de física, por apoyarme con la creación de mi cuenta personal de instituto y por compartir experiencias relacionadas a esta rama de la física.

A mis compañeros de grupo Tom, Alex, Kevin, Akbar, Didier, Fernando, Ericka e Israel por apoyarme y alentarme ofreciéndome consejos para llevar a cabo mi investigación. Por todo el tiempo de convivencia a lo largo de este proyecto, espero seguir colaborando cona ellos en trabajos futuros.

A mi padre Carlos y a mi madre Betty que siempre estuvieron para mí ofreciéndome todo su apoyo en lo que yo necesite a lo largo de toda mi carrera escolar, dándome las herramientas necesarias para convertirme en un profesional. Gracias por alentarme a mejorar, a ser un mejor hijo y un buen estudiante. He llegado hasta aquí gracias a ustedes. Agradezco también a mi madre Martha, aunque ya no este con nosotros, sé que estaría orgullosa de mi por llegar hasta donde estoy.

A mi hermano Uriel por estar ahí en las buenas y en las malas, por apoyarnos mutuamente en los proyectos personales, por compartir experiencias y ser unidos.

A toda mi familia, por alentarme, por confiar en mí y acompañarme en este proceso.

A todos los profesores del instituto de fisca que me moldearon para convertirme en el estudiante que soy ahora. A mis compañeros de clase, por compartir experiencias y hacer de mi estancia en la universidad una de las mejores etapas de mi vida estudiantil. A todo ellos espero seguir viéndolos en un futuro y les deseo todo el éxito en sus proyectos profesionales.

Este trabajo fue apoyado por el Fondo Sectorial de Investigación para la Educación SEP/CONACyT, proyecto 242139, y para el proyecto "Participación de México en la Frontera de Física de Altas Energías en el CERN", CONACyT Proyectos de la Frontera number 2042.