SIMULACIÓN EN PYTHIA 6 DE LA PRODUCCIÓN DE $\widetilde{q}$ Y $\widetilde{g}$ VÍA $W^{\pm}$, CON ESTADO FINAL ν_{τ} Y τ A ENERGÍA DE CENTRO DE MASA DE 14 TeV

LUIS HERNAN PALACIOS ORTEGA

UNIVERSIDAD DE NARIÑO
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
PROGRAMA DE FÍSICA
SAN JUAN DE PASTO
2014

SIMULACIÓN EN PYTHIA 6 DE LA PRODUCCIÓN DE $\widetilde{q}$ Y $\widetilde{g}$ VÍA $W^{\pm}$, CON ESTADO FINAL ν_{τ} Y τ A ENERGÍA DE CENTRO DE MASA DE 14 TeV

LUIS HERNAN PALACIOS ORTEGA

TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE FÍSICO

DIRECTOR:
JAIME BETANCOURT MINGANQUER
MÁSTER EN CIENCIAS FISICAS

UNIVERSIDAD DE NARIÑO
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
PROGRAMA DE FÍSICA
SAN JUAN DE PASTO
2014

"Las ideas y conclusiones aportadas en la tesis de grado son responsabilidad exclusiva de los autores"
Artículo 1. del acuerdo No. 324 del 11 de Octubre de 1966, emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

Nota de aceptación

_____

_____

_____

_____

_____

_____

Msc. Jaime Betancourt Minganquer.
Asesor.

(C) Ph.D. David Martinez Caicedo
Jurado.

Javier Contreras Grijalva
Jurado.

San Juan de Pasto, junio 2014

AGRADECIMIENTOS

A todos aquellos que de una u otra manera han contribuido en el desarrollo en este largo tiempo que me ha tomado finalizar este maravilloso camino conocido como FÍSICA

LUIS HERNAN PALACIOS ORTEGA

RESUMEN

Supersimetria (SUSY), es un modelo que estudia física mas allá del modelo estándar (ME), la cual es una simetría propuesta entre fermiones y bosones, de esta manera a cada boson le corresponde un supercompañero fermión y a cada fermión un supercompañero boson.

En este trabajo se realizará una simulación computacional de colisión protón-protón a una energía de centro de masa de 14 TeV, bajo condiciones experimentales del COMPACT MUON SOLENOID (CMS) ,para el estudio de un canal en particular con decaimiento en estados finales de neutrino y tau.

ABSTRACT

Supersymmetry (SUSY) is a model studying physics beyond the standard model(ME), which is a proposed symmetry between fermions and bosons, this is each boson carries a fermion superpartner and each boson superpartner fermion one.

In this work will be develop a computational simulation of collision proton-proton with energy center of mass of 14 TeV, under frame work of the Compact Muon Solenoid (CMS), for study of a particular channel with final state of decay tau and tau neutrino.

CONTENIDO

INTRODUCCIÓN 15

1. SUPERSIMETRIA 17

1.1. MODELO ESTÁNDAR (ME) 17

1.2. Supersimetria 18

1.3. Rompimiento de supersimetria 20

1.4. Parámetros m-SUGRA 20

1.5. CMS 22

2. Simulación del canal 24

2.1. Phytia 6 24

2.2. Root 24

2.3. Canal de estudio 25

2.3.1. Procesos 26

3. Análisis y resultados 28

CONCLUSIONES Y RECOMENDACIONES 35

ANEXOS 37

3.1. Código canal de simulación 38

ANEXOS 48

3.2. Código energía transversal faltante 49

LISTA DE CUADROS

2.1. ID de las partículas 25

2.2. Parámetros mSUGRA 26

2.3. Identificación de decaimientos 27

LISTA DE FIGURAS

1. Canal de estudio 15

1.1. Partículas modelo estándar 18

1.2. Partículas ME y SUSY 19

1.3. Parametros mSUGRA 21

1.4. CMS 23

3.1. Canal por grupos 28

3.2. Energía transversal faltante 29

3.3. Separación angular entre τ y ν_τ y W^- y χ_1^0 30

3.4. Separación angular entre χ_1^- y q 30

3.5. Separación angular entre τ y ν_τ y W^+ y χ_1^0 31

3.6. Momento transversal τ y ν_τ 31

3.7. Momento transversal W^- y χ_1^0 32

3.8. Momento transversal χ_1^- y q 32

3.9. Momento transversal τ y ν_τ 32

3.10. Momento transversal W^+ y χ_1^0 33

3.11. Energía τ y ν_τ 33

3.12. Energía W^- y χ_1^0 33

3.13. Energía χ_1^- y q 34

3.14. Energía τ y ν_τ 34

3.15. Energía W^+ y χ_1^0 34

LISTA DE ANEXOS

ANEXO A. 37

ANEXO B. 48

GLOSARIO

LSP: Son las siglas en ingles de Lightest Supersymmetric Particle, partícula súper simétrica mas liviana.

LHC: Large Hadron Collider, Gran colisionador de hadrones

GUT: Grand Unification Theory, teoría de la gran unificación

SUSY: Supersymmetry, supersimetría

CMS: Compact muon solenoid, Solenoide compacto de muones

MSUGRA: Minimal supergravity, Minima supergravedad

MSSM: Minimal Supersymmetric Standard Model, modelo estándar mínimo súper simétrico

PARIDAD-R: Numero cuántico en modelos supersimetricos, que permite identificar las sparticulas y las partículas

INTRODUCCIÓN

El modelo estándar (ME) de partículas físicas ha sido notablemente útil en la descripción de los fenómenos a la escala de bajas energías, sin embargo presenta algunos inconvenientes entre los mas notables tenemos la jerarquía de masas y la no inclusión de la interacción gravitacional. [1]

Lo cual hace necesario la introducción de modelos que no presenten estos inconvenientes o extensiones del mismo, entre estos modelos se encuentra la supersimetría (SUSY), de sus siglas en ingles supersymmetry, la cual postula que a cada partícula del ME le corresponde una súper compañera simétrica de SUSY, la diferencia entre las partículas del ME y las partículas SUSY radica en su espín por una cantidad de 1/2.

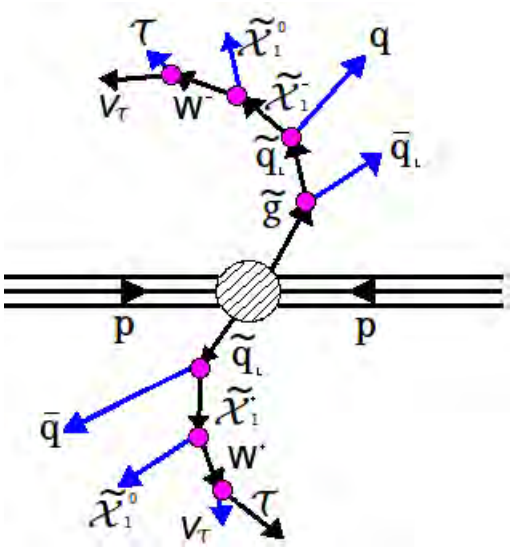

Figura 1: Canal de estudio, fuente de esta investigación.

Lo que se pretende hacer en este trabajo es simular un canal de decaimiento en particular el cual se muestra en la figura 1

Teniendo en cuenta que los canales de decaimiento con estados finales leptonicos tienen

[1] Gokcenur Yesilyurt, Supersymmetry physics studies for the CMS detector upgrade program, Turquia, 2013, p 3, Universidad tecnica de medio oriente

una gran taza de probabilidad y grandes razones de ramificación en este proceso se busca trabajar con un canal de decaimiento de este tipo en especifico el canal de decaimiento mostrado en la figura 1. En el marco de referencia del solenoide compacto de muones, (CMS) de sus siglas en ingles Compact Muon Solenoid, pues a pesar de ser un detector multiproposito del Gran Colisionador de Hadrones (LHC) de sus siglas en ingles Large Hadron Collider, su diseño y detectores ofrecen una gran capacidad en la detección de leptones. [2]

Asi para la simulación de él canal de estudio se contara con el programa PYTHIA 6, en la simulación de la colisión protón-protón y posteriores decaimientos con tanto detalle como le es posible, con base en los modelos teóricos actuales, el otro programa que se usara es ROOT el cual nos permitirá la generación y posterior análisis de los histogramas.

[2]Colaboradores CMS Search for gluino mediated bottom-and top-squark production in multijet final state in pp collisions at 8 TeV, Physics Letter, doi:10.1016/j.physletb.2013.06.058, p 3

1 SUPERSIMETRIA

1.1. MODELO ESTÁNDAR (ME)

La física de partículas es la encargada de explicar como esta constituida la materia y las interacciones que sufre, para ello busca entender el comportamiento de las partículas elementales y así poder explicar todo desde sus cimientos, en la actualidad solo a un pequeño numero de partículas se las llama partículas elementales estas interactúan bajo la acción de campos, por esta razón se dice que es una teoría cuántica de campos, a la descripción de este conjunto de partículas y sus interacciones es a lo que llamamos ME.

Así podemos clasificar las partículas fundamentales y sus interacciones, tomemos primero a las partículas su historia se remonta al año 1942 donde se creía que el universo material estaba constituido por cuatro partículas elementales electrón (e^-), protón (p^+), neutrón (n) y neutrino (ν), pero con los avances de la ciencia y la tecnología se fueron descubriendo nuevas partículas y cada vez la cantidad aumentaba, a tal grado que ya no se podía decir que eran elementales por lo tanto fue necesario clasificarlas en un conjunto mas pequeño, [1] en la actualidad este conjunto es denominado como fermiones a los cuales se les considera como partículas elementales, en los fermiones nos encontramos con dos grupos uno de ellos compuesto por seis quarks y el otro por seis leptones pero estas partículas interactúan con los bosones a los cuales se les conoce como las partículas mediadoras de interacción, entre ellas se destaca la diferencia en el valor del espín pues mientras los bosones tienen espín entero los fermiones tienen espín semientero. Así el conjunto compacto de partículas e interacción del ME se resume como se muestra en la figura 1.1

[1] Gokcenur Yesilyurt, Supersymmetry physics studies for the CMS detector upgrade program, Turquia, 2013, p 3, Universidad tecnica de medio oriente

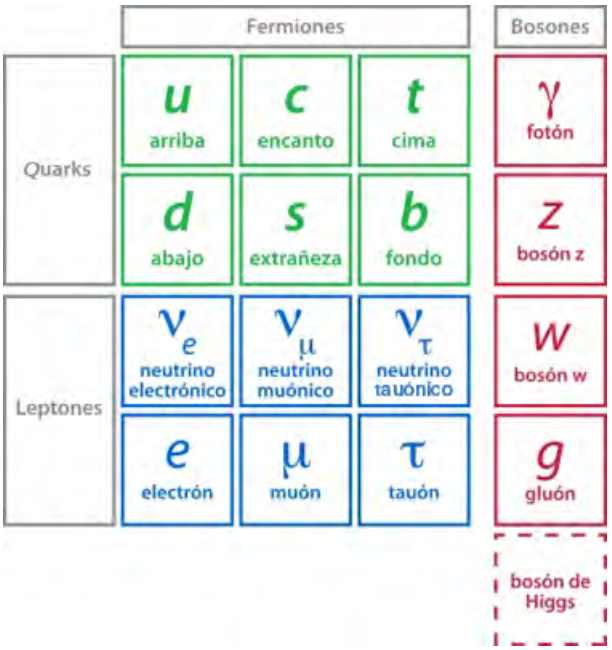

Figura 1.1: Partículas modelo estándar Fuente: URL: http://www.revistadelauniversidad.unam.mx/articulo.php?publicacion=9&art=100&sec=Art %C3 %ADculos

Tratándose de una teoría de unificación la primera restricción es que este modelo no incluye la interacción gravitacional, no puede explicar la materia oscura ni la energía oscura, y estas representan un 95.1 % del universo, tampoco puede explicar la asimetría entre materia y antimateria, no puede explicar por que hay tres generaciones de fermiones, tampoco la jerarquía de masas,[2] y trabaja con al menos 19 parámetros arbitrarios entre ellos acoplamientos, masas y mezclas las cuales son medidas experimentalmente no hay forma de predecirlas teóricamente.

1.2. Supersimetria

El modelo estándar mínimo supersimétrico (MSSM), es un esquema para introducir SUSY en el ME ya que la supersimetría permite una relación fermión-bosón para cada grado de libertad dado por un operador que transforma un bosón en un fermión y viceversa, a estas partículas resultantes se les conoce como compañeras supersimetricas de las partículas del ME, ya que tiene los mismos números cuánticos excepto uno, el

[2]Lina Huertas, producción de s-quarks y gluinos en el experimento CMS, San Juan de Pasto, 2009, p 19, fisica, Universidad de Nariño, Ciencias exactas y naturales, Física

espín, el cual difiere en 1/2, para diferenciar las partículas del ME de las partículas SUSY a los nombres se les dio un prefijo y un sufijo, a los nombres de las partículas se les antepone una "s" por ejemplo un quark se convierte en un s-quark y a las partículas mediadoras es decir a los bosones la terminación "ino" es decir un gluón se convierte en un gluino, adicional a esto a las letras que representan partículas SUSY se coloca sobre ellas el símbolo " ˜ " como se muestra en la figura 1.2.

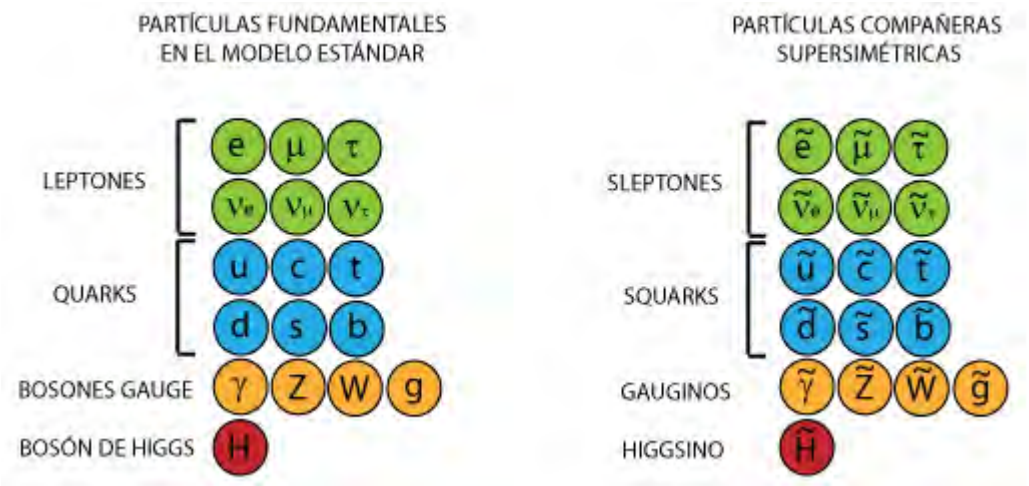

Figura 1.2: Partículas ME y SUSY, Fuente: URL:http://www.ramanujan25449.blogspot.com/2013/01/susy_6764.html

Las características mas importantes de las partículas supersimetricas derivadas de la conservación de R-paridad son :

- Las s-partículas pueden ser producidas solo en pares
- La partícula supersimetrica mas liviana llamada LSP (*Lightest Supersymmetric Particle*) es estable con R-paridad =-1 por lo tanto no puede decaer en una partícula del modelo estandar
- S-partículas decaen finalmente en partículas del ME y LSP

Donde la R-paridad es un nuevo numero cuántico con el cual podemos distinguir las partículas del ME con las de SUSY y se define como

$$R = (-1)^{3B+L+2S} \tag{1.1}$$

Donde B y L hacen referencia a los números barionico y leptonico de la partícula respectivamente mientras que S representa al espín, teniendo en cuenta que la única diferencia entre las partículas supersimetricas y las del modelo estándar es en 1/2 el valor del espín tenemos R=1 para las partículas del ME y para las partículas supersimetricas R=-1.

1.3. Rompimiento de supersimetria

Susy es una simetría rota pues por su relación bosón-fermión y fermión-bosón hubieran permitido la detección de partículas supersimetricas a bajas energías pero esto no es así, no se ha encontrado degenerancia en la masa de las partículas del ME, así la única explicación es que hay un mecanismo que rompe la simetría de SUSY, en el momento se tiene varios candidatos para esto, uno de ellos es el modelo de mínima supergravedad mSUGRA de sus siglas minimal supergravity. [3]

1.4. Parámetros m-SUGRA

Así para que SUSY sea una teoría que cumple con todos los requisitos para ser una teoría de unificación justificando clases de masas y acoplamientos es necesario contar con ciertos parámetros que me permiten esto y que sean la mínima cantidad posible, para ello usamos la gravedad como mensajero la cual nos da cinco parámetros los cuales son:

- m_0 Masa escalar comun
- $m_{1/2}$ Masa del gaugino comun
- A_0 Cupla trilineal
- $tan\beta = \frac{v_2}{v_1}$ donde v_1 y v_2 son los valores esperados en el vacío del campo de Higgs H1 H2
- μ Signo de la masa del Higgisino

[3] Daniel Sprenger, Search for supersymmetry in opposite-sign dilepton final states with the CMS experiment, Leverkusen, 2012, p 8, doctor en física RWTH Aachen University, Matematicas, informatica y ciencias naturales

Los parámetros mSUGRA están dados por los puntos LM1 a LM9 y por los puntos HM1 a HM4 donde LM esta relacionado con valores de masa bajos y HM con valores de masa altos, estos se ubican en el plano m_0 vs $m_{1/2}$ como se muestra en la gráfica 1.3

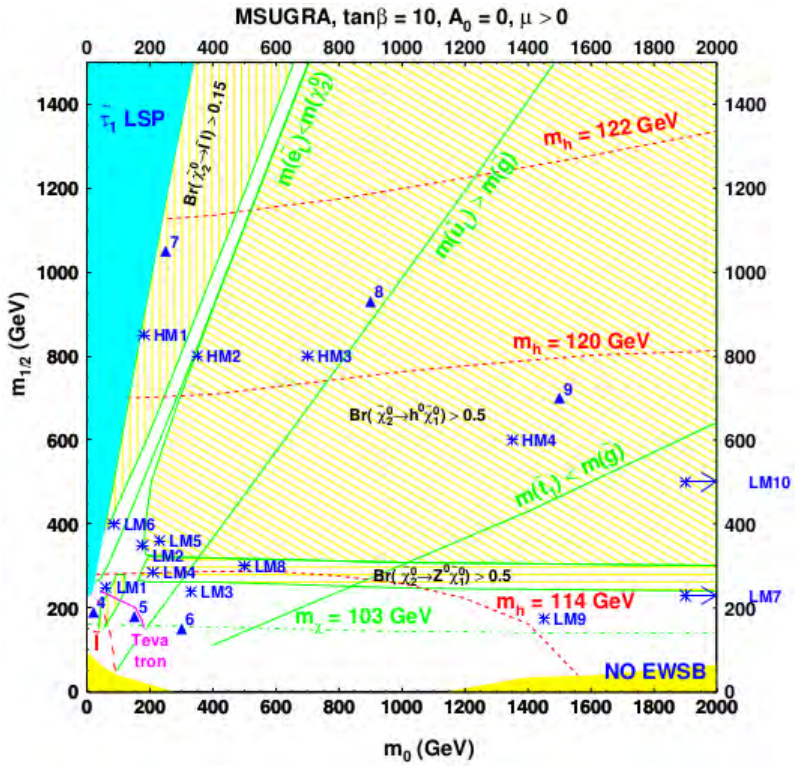

Figura 1.3: Parametros mSUGRA. Fuente: CMS Physics, Volume II: Detector Performance and Software, CERN/LHCC 2006-001, CMS TDR 8.1., pag. 408.

Esta gráfica esta dividida por dos diagonales principales de color verde la primera inicia cerca al origen en la región amarilla, etiquetada por $m(\widetilde{u}_L) > m(\widetilde{g})$, la otra es la segunda diagonal de color verde que inicia aproximadamente en 400 (GeV) del eje m_0 y en 100 (GeV) del eje $m_{1/2}$ etiquetada con $m(\widetilde{t}_1) < m(\widetilde{g})$ [4] estas regiones nos informan los modos posibles de decaimiento después de la colisión protón-protón de esta manera tenemos tres regiones: [5]

- Región 1: En esta región, los gluinos son mas pesados que cualquier s-quark. Los canales de decaimiento esperados son $\widetilde{g} \to \widetilde{q}\bar{q}$, $\bar{q} \to q\chi$

[4]Niklas Mohr, Neutralino reconstuction in dilepton final states with the CMS experiment Aachen, 2008, p 14,físico, RWTH university,

[5]Colaboradores CMS, CMS physics technical desing report, 2006, p 395, CERN/LHCC 2006-021

- Región 2: En esta región algunos quarks son mas pesados, otros son mas ligeros que el gluino. Resultando cadenas de decaimiento mas complicadas, algunas de ellas son $\widetilde{q}_L \to \widetilde{g}q$, $\widetilde{g} \to \widetilde{b}\bar{b}$, $\widetilde{b} \to b\chi$, como los $\widetilde{q}_L$ de las dos primeras generaciones se esperan sean los mas pesados mientras que $\widetilde{b}_1$ y $\widetilde{t}_1$ los mas livianos.

- Región 3: En esta región los gluinos son mas livianos que los s-quarks y el modo de decaimiento es $\widetilde{q} \to \widetilde{g}q$, $\widetilde{g} \to q\bar{q}\chi$.

1.5. CMS

El experimento CMS es uno de los detectores del LHC el cual esta construido para un amplio rango de investigación en teorías físicas tales como dimensiones extra, partículas candidatas a materia oscura, bosón de Higgs y nueva física. CMS tiene un diseño de multicapas provisto de una gran solenoide superconductor que le provee un campo magnético de 4 teslas, cuando las partículas abandonan su trayectoria debido a la colisión, el primer detector que encuentran en el CMS es llamado tracking system (sistema de rastreo) formado por detectores de pixeles de silicio, donde son medidas las partículas cargadas permitiendo reconstruir su trayectoria, seguido a este detector tenemos los calorímetros los cuales se encargan de medir la energía de electrones, fotones y jets el primero de los calorímetros es el calorímetro electromagnético (ECAL), que mide con gran precisión la energía de los electrones y fotones pues estos interactúan electromagneticamente luego esta el calorímetro hadronico (HCAL), el cual se encarga de medir la energía de los hadrones los cuales interactúan bajo la acción de la fuerza fuerte, finalmente las ultimas partículas en ser detectadas son los muones, los cuales interactúan débilmente por esta razón pueden llegar hasta los puntos mas externos del detector sin perder energía, donde están ubicados los detectores de muones. [6]

La estructura del detector se muestra en la figura 1.4

[6]Kolja Kaschube, Search for stable stau production at the LHC, Hamburgo, 2011, p 31-41, Doctor en física, Universidad de Hamburgo, física

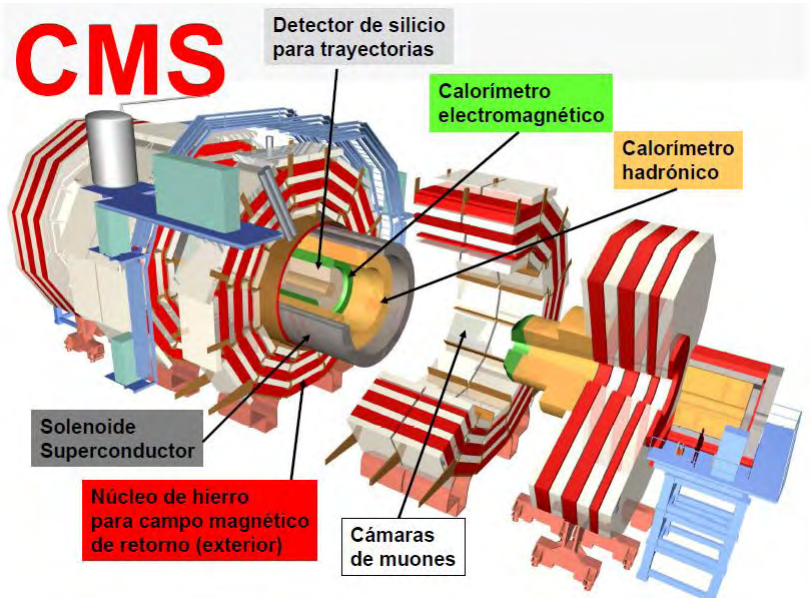

Figura 1.4: CMS, Fuente: URL: http://cms.web.cern.ch/news/cms-detector-design

2 Simulación del canal

Para llevar acabo la simulación del canal escogido, indicado en la figura 1 es necesario escoger los procesos pertinentes y los códigos de las partículas que se desea en el orden correcto, para ello contamos con el programa Phytia 6 que tiene una interfase con Root para hacer el posterior análisis de los datos obtenidos.

2.1. Phytia 6

Pythia 6 es un sofware libre que genera eventos de colisión de partículas bajo condiciones de un marco de referencia especifico basado en un sistema montecarlo para generar números aleatorios, trabaja principalmente con librerías de fortran 77 y C++. Este sofware permite la simulación de canales supersimetricos a diferentes valores de energía, lo puede hacer con tanta precisión como le es posible basado en las teorías existentes al momento de su creación es así como puede generar un gran cantidad de procesos y subprocesos que se producirían después de la colisión principal. En este trabajo se usa colisión protón-protón, en el marco de referencia del CMS por lo tanto cada paquete o bunch que la simulación hace colisionar contiene $1.1\text{x}10^{11}$ protones, en esta simulación se usara un total de 500000 paquetes o bunch dando así un total de $5.5\text{x}10^{16}$ protones esto con el fin de obtener una buena estadística y acercarse a lo que ocurre en la realidad. Es importante tener en cuenta que los procesos y subprocesos usan códigos tanto numéricos como literales específicos de Pyhtia 6 [1] al igual que para identificar las partículas y sus decaimientos que en este caso están dados por el Particle Data Group (PDG).

2.2. Root

Root es un sofware libre que trabaja principalmente bajo librerías de C++ y su interfaz con el usuario se fundamenta en el uso de objetos y clases, su diseño esta enfocado en el análisis de datos mediante histogramas, fue desarrollado en el Centro Europeo de investigaciones Nucleares (CERN) posee una amplia gama de opciones para la generación de

[1] Torbjörn Sjöstrand, Steben Mrenna, Peter Skand, Pythia 6.4 Physics and Manual 2006

histogramas con base en datos adquiridos ya sea en una simulación o con datos reales de un experimento determinado.

Para nuestro caso trabajaremos con una interfaz ente Pythia y root permitiendo la creación directa de los histogramas en la simulación. En las lineas 4, 5 y 7 segunda pagina del apéndice A se puede ver los comandos para la interfase entre pythia y root. [2]

2.3. Canal de estudio

El canal de estudio figura 1 esta conformado tanto por partículas del ME como por partículas SUSY estas son quarks (q_L), antiquarks ($\bar{q_L}$), neutrino taunico (ν_τ), tau (τ) y $W^\pm$ representando al ME y gluinos ($\widetilde{g}$), squaks ($\widetilde{q_L}$), charginos ($\chi_1^\pm$) y neutralinos (χ_1^0) a SUSY.

En el cuadro 2.1 se encuentran los códigos usados por el programa para identificar estas partículas. Donde el subíndice L viene del ingles left con esto el canal de decaimiento escogido solo tomara en cuenta quarks y squaks de este tipo.

En el PDG las partículas y antipartículas comparten ID, no es necesario especificar cada uno de los quarks existentes pues se conoce que son 6 por lo tanto su ID va del 1 al 6 de forma similar para los squaks los cuales van del 1000001 al 1000006.

Partículas	ID
q_L y $\bar{q_L}$	1-6
τ	15
ν_τ	16
$W^\pm$	24
$\widetilde{q_L}$	1000001-1000006
$\widetilde{g}$	1000021
χ_1^0	1000022
$\chi_1^\pm$	1000024

Cuadro 2.1: ID de las partículas

[2]The root Team, Root an object-oriented data analysis framework, 2009

2.3.1. Procesos El primer proceso en ocurrir es la colision proton-proton a una energía de centro de masa de 14 TeV en el marco de referencia del CMS, en este punto se genera una colisión de 5.5x10^{16} protones que viajan en direcciones opuestas, que bajo las condiciones de los parámetros mSUGRA generaran $\widetilde{q}$ y $\widetilde{g}$ los cuales a su vez decaerán en un tipo limitado de partículas y estas decaerán en otro tipo de partículas y así sucesivamente, por lo tanto es necesario escribir una linea de código en el programa para identificar las partículas que son de nuestro interés.

El primer paso entonces es activar el proceso físico deseado el cual se ejecuta con el comando MSUB(ISUB)=1, donde ISUB es el numero del proceso que se desea, una lista completa de estos procesos puede ser revisada en el manual. [3] Para nuestro caso tenemos

Procesos	numero ISUB
$f_i g \rightarrow \widetilde{q}_{iL}\widetilde{g}$	258

Con esta información el programa después de la colisión protón-protón solo tomara en cuenta la producción de $\widetilde{q}_{iL}$ y $\widetilde{g}$.

Seguido a esto es necesario proporcionar al programa los parámetros mSUGRA con los cuales se desea trabajar la simulación, en este caso se trabajara en la primera región con parámetros de masa altos en especifico con el punto HM1 cuyos valores se muestran en el cuadro 2.2

Punto	m_0	$m_{1/2}$	$tan\beta$	signo μ	A_0
HM1	180	850	10	+	0

Cuadro 2.2: Parámetros mSUGRA

Ahora con los parámetros de procesos físicos y mSUGRA tenemos una primera parte fundamental para el desarrollo del decaimiento de nuestro canal. Recordando que estas partículas decaerán en una amplia gama de partículas es necesario restringir el proceso a las partículas de nuestro canal, para ello es indispensable desactivar todos estos posibles decaimientos y activar solo los de nuestro interés pues hay un total de 4352 canales de decaimiento los cuales se pueden consultar en el PDG, para nuestro caso los

[3] Torbjörn Sjöstrand, Steben Mrenna, Peter Skand, Pythia 6.4 Physics and Manual 2006

decaimientos requeridos se muestran en el cuadro 2.3

Partícula	ID	decaimiento
$\widetilde{g}$	1000021	1975-2162
$\widetilde{d}_L$	1000001	1592-1637
$\widetilde{u}_L$	1000002	1637-1661
$\widetilde{s}_L$	1000003	1661-1706
$\widetilde{c}_L$	1000004	1706-1730
$\widetilde{b}_L$	1000005	1730-1775
$\widetilde{t}_L$	1000006	1775-1802
$\widetilde{\chi}_1^{\pm}$	1000024	2595-2826
$W^{\pm}$	24	190-209

Cuadro 2.3: Identificación de decaimientos

3 Análisis y resultados

Una vez se ha escogido el proceso físico indicado, los respectivos decaimientos y los números que identifican a las partículas se procede a estructurar el código como se muestra en el apéndice A en la sección de generador de eventos, bajo las condiciones necesarias para obtener el canal de decaimiento el cual separamos en 2 grupos. Dicha separación es llamada *Algoritmo Hemisferio* [1] muy utilizado para analizar las partículas que decaen de un gluino y por separado las que decaen de un squark asi el grupo 1 y grupo 2 quedan como se muestra en la figura 3.1

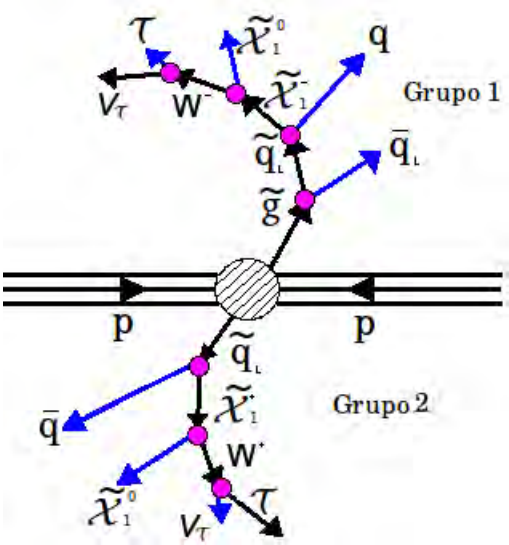

Figura 3.1: Grupos canal de decaimiento, fuente de esta investigación.

Después se extraerá la información de energía transversal faltante, momento transversal, separación angular para generar sus respectivos histogramas. Por lo tanto todas las gráficas a continuación mostradas en este trabajo, son fuente de esta investigación.

Para catalogar a un evento como súper simétrico su característica principal es un gran valor de energía transversal faltante esto es $E_T > 230$ GeV, para encontrar el valor de esta energía utilizamos la siguiente ecuación[2]

[1] CMS Physics Technical Desing Report, Volume II: Physics Performance, CERN/LHC 2006-021
[2] HEYNINCK, Jan. The CMS Top Quark Physics, p94, 2008.

$$\vec{E}_t^{miss} = -\sum_n (E_n \, sen\,\theta_n \, cos\,\phi_n \, \hat{\imath} + E_n \, sen\,\theta_n \, sen\,\phi_n \, \hat{\jmath}) \tag{3.1}$$

Esta esta incluida en el código desarrollado para encontrar energía transversal faltante.

En este trabajo se usa una energía de centro de masa de 14 TeV ya que será la máxima energía alcanzada en el LHC. a una luminosidad de $\pounds = 10^{34} cm^{-2} s^{-1}$ la cual es la máxima luminosidad que alcanzara el LHC, un motivo adicional para trabajar con estos valores es el uso de parámetros mSUGRA de masa en la región 1 con el punto (HM1) esto es para valores de masa altos, pues estos proporcionan las mejores condiciones para generar el canal de decaimiento con estado final de único leptón (τ).

Utilizando la ecuación 4.1 se obtuvo la siguiente gráfica de energía transversal faltante.

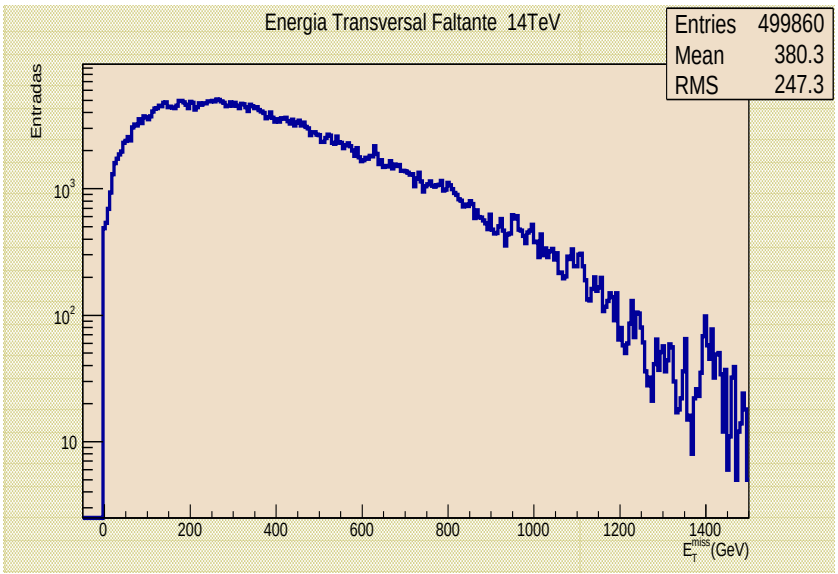

Figura 3.2: Energía transversal faltante

De la figura 3.2 podemos observar un valor de E_T = 380,3 GeV, lo cual nos da indicios de que el canal simulado es un canal SUSY.

En las siguientes gráficas se muestra la separación angular entre pares de partículas de la parte superior e inferior del canal, las cuales se pueden comparar con gráficas de separación angular obtenidas con los datos del CMS para hacer una primera selección de partículas candidatas según sea el caso, pues cabe recordar que solo estamos tomando en cuenta uno de los posibles procesos físicos y partículas tales como las que se trabajan en este canal pueden provenir de otros decaimientos o interacciones.

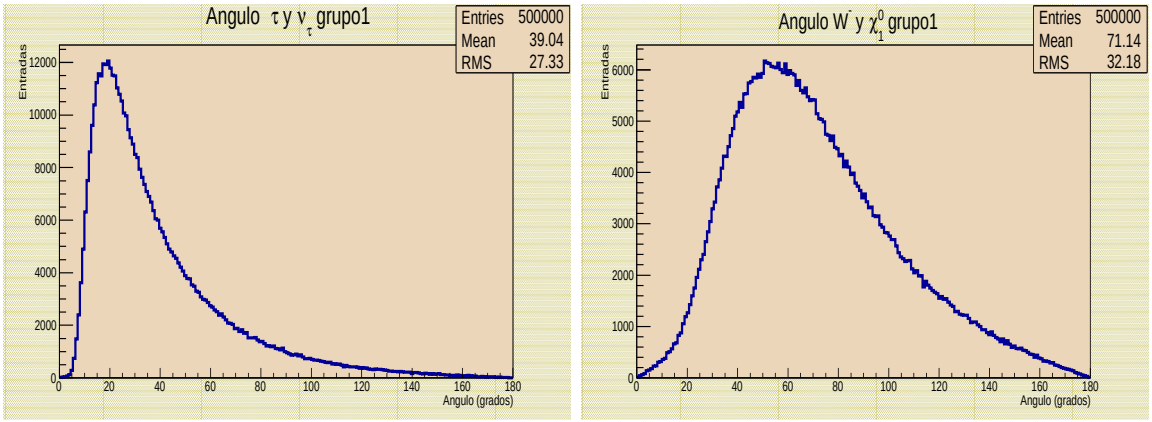

Figura 3.3: Separación angular entre τ y ν_τ izquierda y separación W^- y χ_1^0 derecha

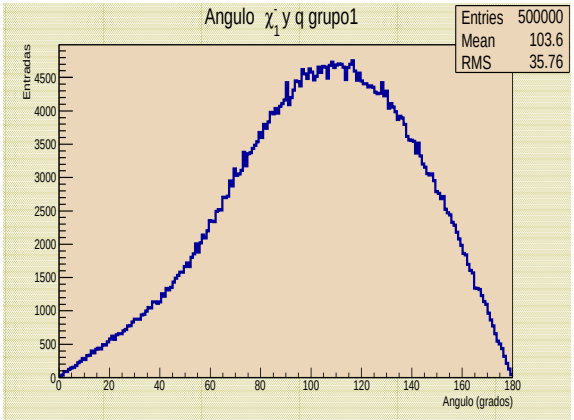

Figura 3.4: Separación angular entre χ_1^- y q

En el siguiente grupo de gráficas encontramos el momento transversal faltante para cada una de las partículas del canal. Tanto para el grupo 1 como para el grupo 2

Comparando los valores de momento para las partículas que conforman este canal tenemos que las del grupo 2 tienen un valor mayor comparadas con las del grupo 1, si observamos el canal de decaimiento vemos que del gluino el cual tiene mayor masa que cualquier squark hay un decaimiento mas que del squark lo cual explicar por que el momento es mayor en el grupo 2 y no en el grupo 1 a pesar de tener mayor masa.

Finalmente tenemos el grupo de gráficas que representan la energía de cada una de las partículas en las cuales se puede apreciar como la suma de las energías de un par de partículas es cercana a la energía de la partícula madre.

Ejemplo si sumamos la distribución de energía de los tau $E_\tau = 218{,}1$ y de los neu-

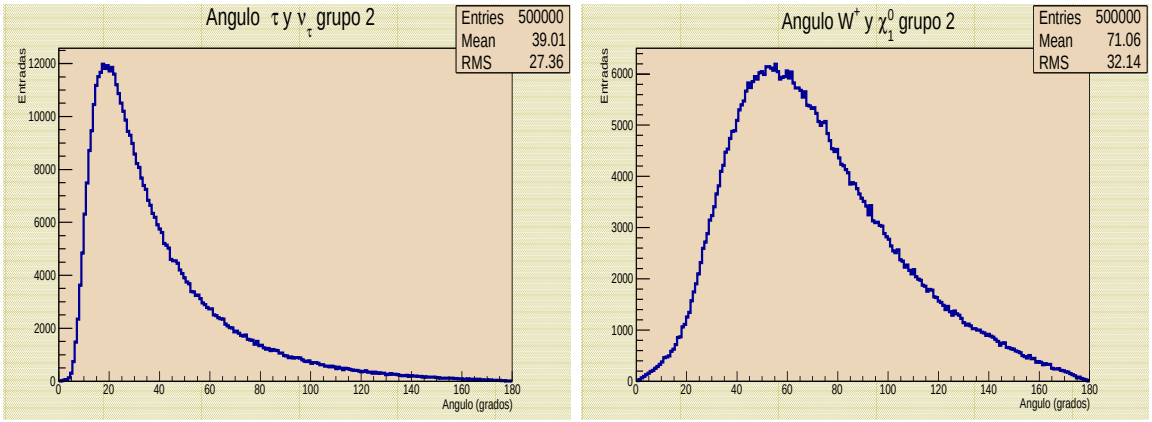

Figura 3.5: Separación angular entre τ y ν_τ izquierda y separación W^+ y χ_1^0 derecha

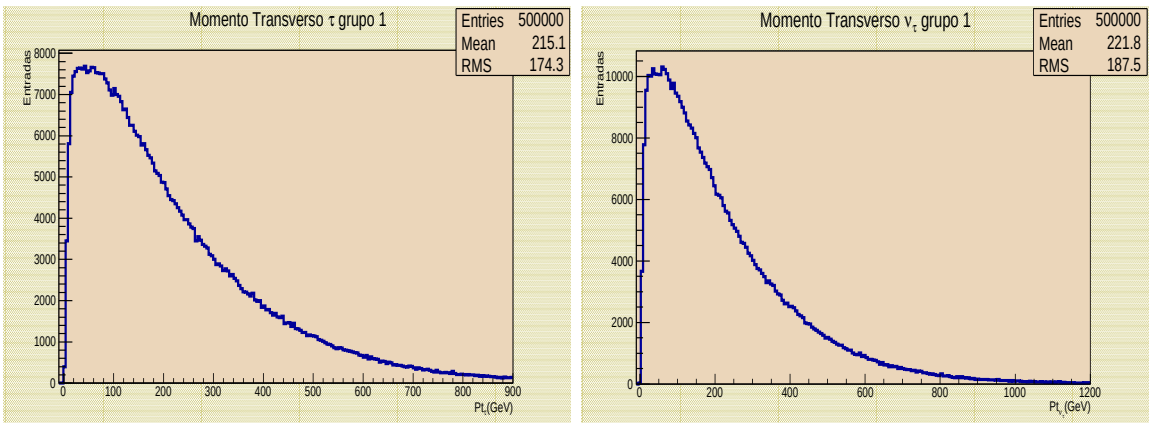

Figura 3.6: Momento transversal τ izquierda y Momento transversal ν_τ

trinos tauonicos $E_{\nu_\tau} = 218{,}5$ tenemos $E = 436{,}6$ este valor se aproxima al valor de la distribución de energía de las partículas madres los W^- la cual tiene un valor de $E_{W^-} = 445{,}1$

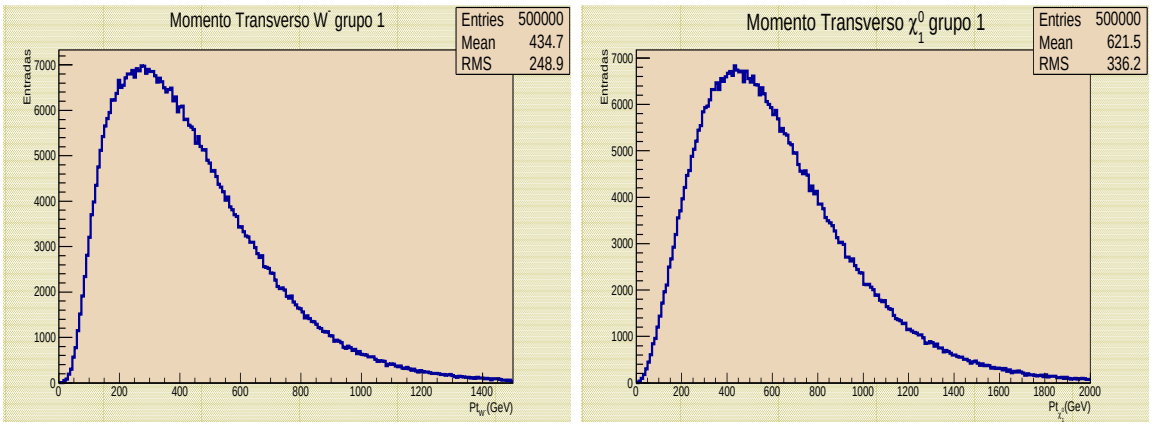

Figura 3.7: Momento transversal W^- izquierda y Momento transversal χ_1^0 derecha

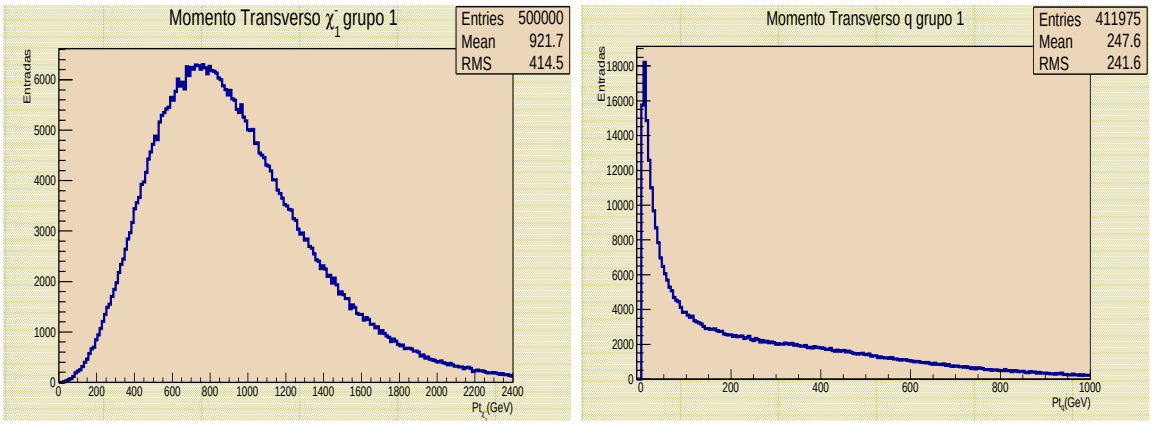

Figura 3.8: Momento transversal χ_1^- izquierda y Momento transversal q derecha

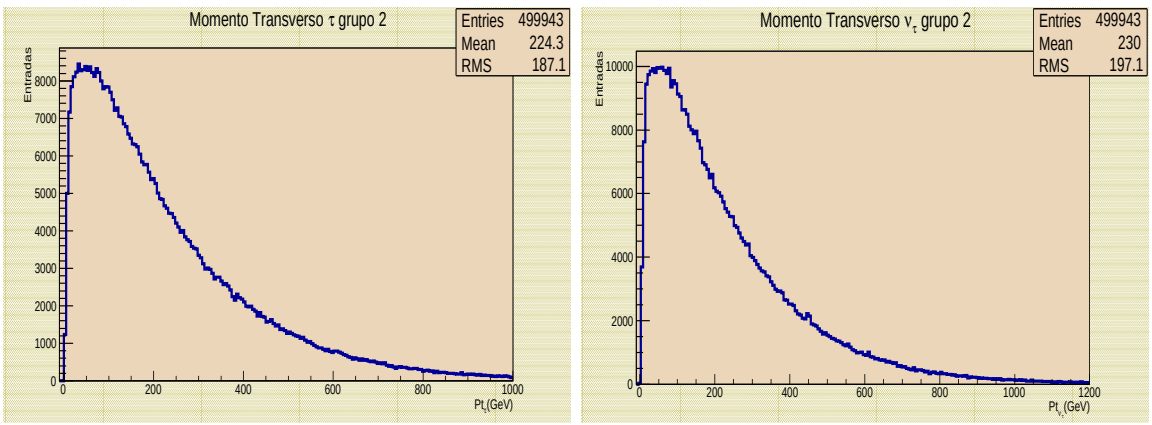

Figura 3.9: Momento transversal τ izquierda y Momento transversal ν_τ

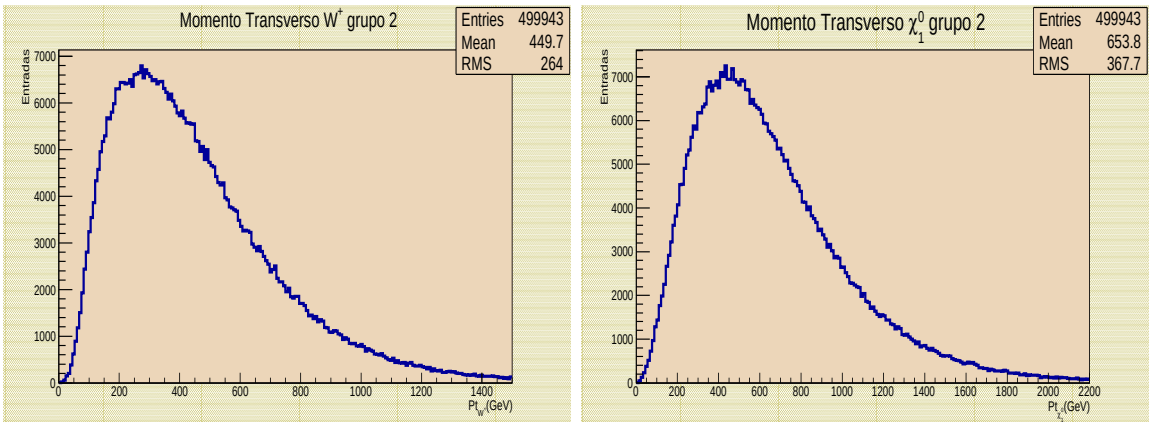

Figura 3.10: Momento transversal W^+ izquierda y Momento transversal χ_1^0 derecha

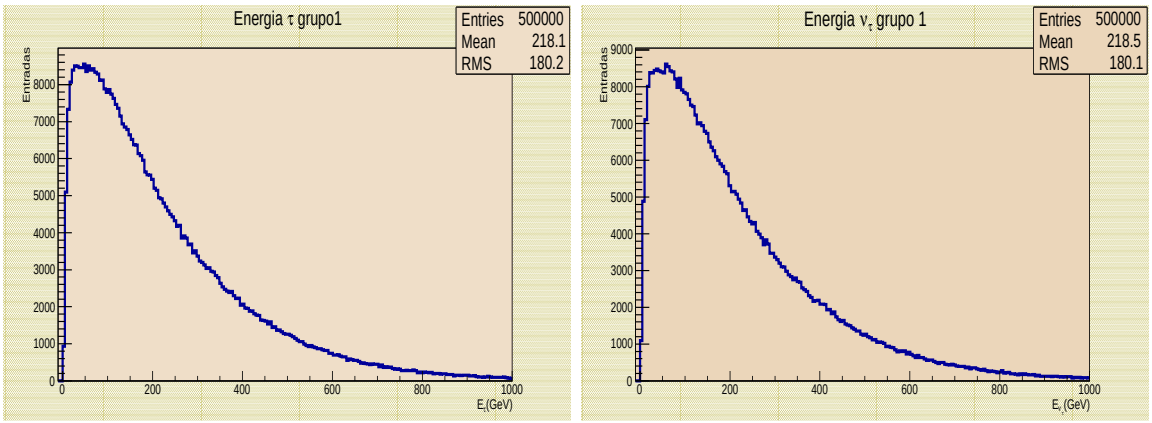

Figura 3.11: A la izquierda energía τ y a la derecha energía ν_τ

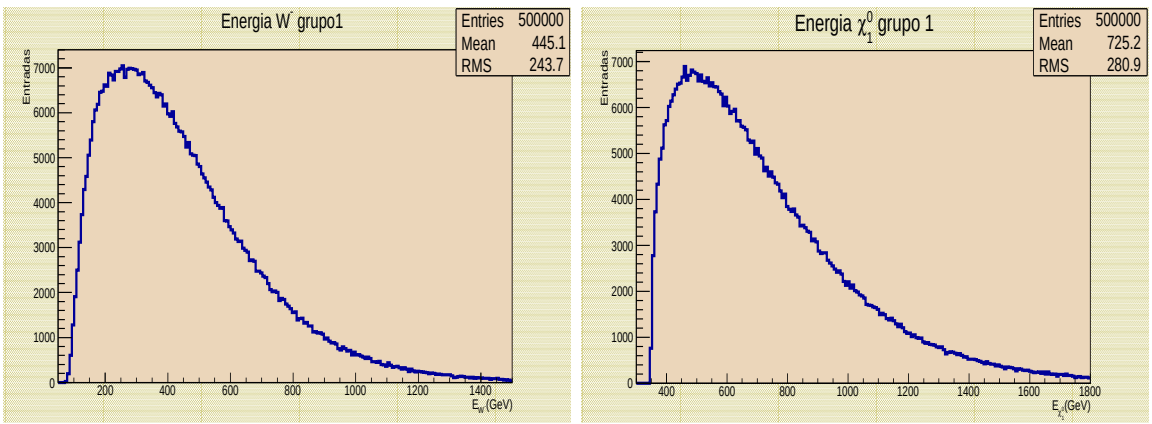

Figura 3.12: A la izquierda energía W^- y a la derecha energía χ_1^0

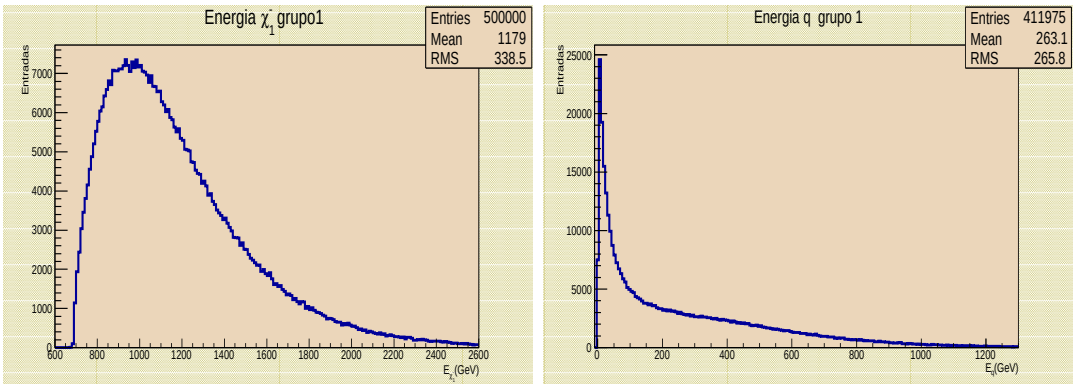

Figura 3.13: A la izquierda energía χ_1^- y a la derecha energía q

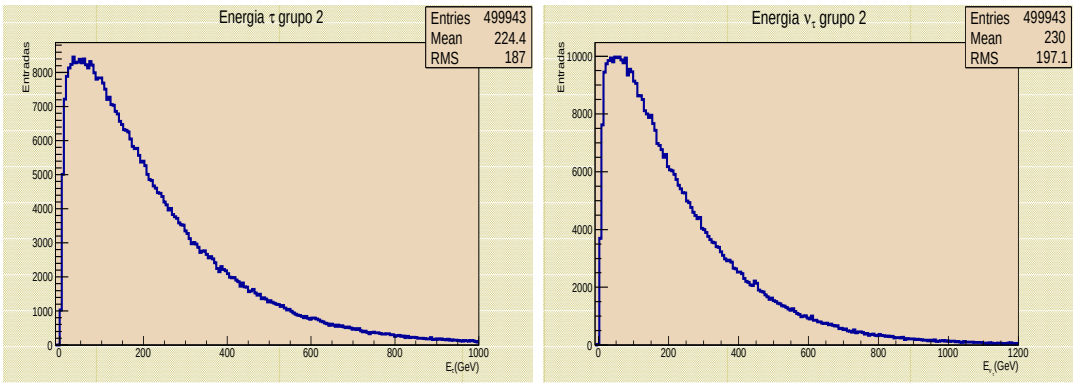

Figura 3.14: A la izquierda energía τ y a la derecha energía ν_τ

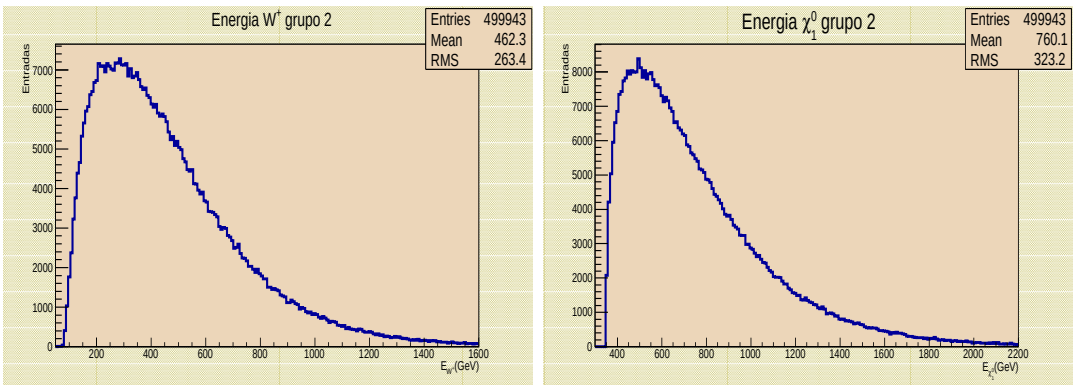

Figura 3.15: A la izquierda energía W^+ y a la derecha energía χ_1^0

CONCLUSIONES Y RECOMENDACIONES

- Para cumplir con los objetivos es necesario obtener el canal de decaimiento de la figura 1 esto fue posible usando los condicionales necesarios como se muestra en el apéndice A
- En este estudio se obtuvo un valor de energía transversal faltante $E_T = 380$ GeV el cual es mayor a 230 GeV lo cual nos puede hacer inferir que nuestra simulación es un proceso SUSY, pues es la condición para que un proceso se considere como supersimetrico.
- la separación angular nos confirma que los pares de partículas producidos provienen del canal de decaimiento y estos valores nos serviran como parámetros de medidas para futuros experimentos. El maximo valor medio que se obtuvo de separación angular fue entre χ_1^0 y los q que es de 103 grados.

Se recomendaría hacer un estudio con datos tomados en el CMS de los valores de separación angular y comprobar que corresponden al canal de decaimiento simulado recordando que esta simulación esta realizada con el máximo valor tanto en energía como en luminosidad que alcanzara el LHC lo cual sera aproximadamente en el año 2017. Y si los resultados coinciden también se puede comprobar los valores de energía y momento.

BIBLIOGRAFÍA

Baden, Andrew R. Jets and Kinematics in Hadronic Collisions, International Journal of Modern Physics A, World Scientific Publishing Company, Revised December 24, 1997.

Carlos Arturo Ávila El Experimento CMS del Acelerador LHC, Proyectos conjuntos entre la Universidad de los Andes y las Universidades de Nariño y Tolima. Video conferencias 2010.

CMS Physics, Volume I: Detector Performance and Software, CERN/LHCC 2006-001, CMS TDR 8.1. 673. p.

Heinz R. Pagels, Perfect Symmetry- The Search for the Beginning of Time, New York, Batam Books, Junio de 1985, 209. p.

Juan Vicente Guerrero, Más Allá del Modelo Estándar a la Escala TeV, Bariloche Argentina 2012, Maestría en ciencias físicas, Instituto Balseiro, Universidad Nacional de Cuyo, Comisión Nacional de Energía Atómica Argentina.

Lina Huertas Guativa, producción de s-quarks y gluinos en el experimento CMS.

Michel Della Negra, Alain Herve, Lorenzo Foa, Paraskevas sphicas, Darin Acosta, Albert De Roeck, Lucia Silvestri, Bari Avi Yagil, CMS Physics Technical Desing Report.

Mohr Niklas Neutralino Reconstruction in Dilepton Final State with the CMS Experiment, Rheinisch-Westfälischen Technischen Hochschule Aachen 2008,I. Physikalischen Institut B.

O. Buchmueller, R Cavanaugh, A, De Roeck,J.R. Elis, H. Flächer, S. Heinemeyer, G. Isidori, K.A. Olive, F.J. Ronga, G. Weiglen, Likelihood Function for Supersymmetric Observables in Frequentist Analyses of the CMSSM an NUHM1, 2009.

Perkins, Donald H. Introduction to High Energy Physics Four Edition, Cambridge University Press, 2000. 19. p.

P. Binétruy, Supersymmetry Theory, Experiment, and Cosmology, Unisersidad de Paris, 2006.

Torben Schum, QCD Processes and search for supersymmetry at the LHC, 2012, Doctorado, Universität Hamburg, Departamento de Física.

Torbjörn Sjöstrand, Steben Mrenna, Peter Skand, Pythia 6.4 Physics and Manual. 489. p.

ANEXO A.

3.1. Código canal de simulación

Simulación para el grupo 1

```
 //=======================================================
// Histogramas Angulo, Energia y Momento
// Author Luis H. Palacios
//Simulación Pythia 6 Estado Final neutrinos tauonicos
//=======================================================
#ifndef __CINT__
#include <stdlib.h>
#include <TROOT.h>
#include <TRint.h>
#include <TApplication.h>
#include <TFile.h>
#include <TMCParticle.h>
#include <TObjArray.h>
#include <TPythia6.h>
#include <TTree.h>
#include <TClonesArray.h>
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>
#include "Riostream.h"
#include "TNetFile.h"
#include "TRandom.h"
#include "TBranch.h"
#include "TStopwatch.h"
#endif

 void loadLibraries()
{
#ifdef __CINT__

gSystem->Load("libEG");
gSystem->Load("usr/local/lib/root/libPythia6");
//change to your setup
gSystem->Load("usr/local/lib/root/libEGPythia6");
#endif
}
struct pythia_particle {
Int_t status;
// status of particle ( LUJETS K[1] )
Int_t pdg_id;
// flavour code  ( LUJETS K[2] )
Int_t parent;
// parrent's id  ( LUJETS K[3] )
Int_t firstChild;
// id of first child  ( LUJETS K[4] )
Int_t lastChild;
// id of last  child  ( LUJETS K[5] )
Float_t momentum[4];
//X.Y,Z,energy momenta [GeV/c] (LUJETS P[1]=P[4])
Float_t mass;
// Mass [Gev/c^2]  ( LUJETS P[5] )
Float_t vertex[4];   // X,Y,Z vertex  [mm]
// time of procuction [mm/c]( LUJETS V[1]-V[4] )
Float_t lifetime;
// proper lifetime [mm/c]     ( LUJETS V[5] )
```

```
};
int P2ATest(int nEvent=5, int compress=1);
int P2ATest(int nEvent, int compress)
{
TStopwatch fulltime;
Int_t i;
Int_t NEvents;
Int_t startdecay;
Int_t enddecay;
NEvents = nEvent;

//====================== Histogramas  ==================
TFile* file=new TFile("tesis.root","RECREATE","tesis.root"
,compress);
TPythia6 Pythia;
TTree *tree = new TTree("T","tree");
TH1F *ener1 = new TH1F("Energia1","Energia #tau grupo 1"
, 200, -10, 1000);
TH1F *ener2 = new TH1F("Energia2","Energia #nu_{#tau} grupo
1", 200, -10, 1000);
TH1F *ener3 = new TH1F("Energia3","Energia W^{+} grupo1",
200, 50, 1500);
TH1F *ener4 = new TH1F("Energia4","Energia #chi_{1}^{0}
grupo 1", 200, 300, 1800);
TH1F *ener5 = new TH1F("Energia5","Energia #chi_{1}^{+}
grupo1", 200, 600, 2600);
TH1F *ener6 = new TH1F("Energia6","Energia q grupo 1", 200,
-10,1300);
TH1F *ener7 = new TH1F("Energia7","Energia ~q grupo1", 200,
0,600);
TH1F *ener8 = new TH1F("Energia8","Energia q grupo 1", 200,
350, 1000);
TH1F *momento1 = new TH1F("Pt-1","Momento Transverso #tau
grupo 1",200, -10,900);
TH1F *momento2 = new TH1F("Pt-2","Momento Transverso
#nu_{#tau} grupo 1", 200,-10,1200);
TH1F *momento3 = new TH1F("Pt-3","Momento Transverso W^{+}
grupo 1",200, 0,1500);
TH1F *momento4 = new TH1F("Pt-4","Momento Transverso
#chi_{1}^{0} grupo 1", 200,0,2000);
TH1F *momento5 = new TH1F("Pt-5","Momento Transverso
#chi_{1}^{+} grupo 1", 200,0,2400);
TH1F *momento6 = new TH1F("Pt-6","Momento Transverso q
grupo 1",200,-10, 1000);
TH1F *momento7 = new TH1F("Pt-7","Momento Transverso ~q
grupo 1",200,-50, 600);
TH1F *momento8 = new TH1F("Pt-8","Momento Transverso q
grupo 1",200,-10, 500);
TH1F *angu = new TH1F("Angle1","Angulo  #tau y #nu_{#tau}
grupo 1", 200, 0, 180);
TH1F *angu2 = new TH1F("Angle2","Angulo W^{+} y #chi_{1}^{0}
grupo1", 200, 0, 180);
TH1F *angu3 = new TH1F("Angle3","Angulo  #chi_{1}^{+} y q
grupo1", 200, 0, 180);
TH1F *angu4 = new TH1F("Angle4","Angulo ~q y q_bar grupo1",
200, 0, 180);

//=========================================================
// PROCESOS   =
//=========================================================
Pythia.SetMSEL(0);
// subprocesos deseados deben ser activados.
// permite la produccion de squark-gluino
Pythia.SetMSUB(258,1);  // f_i g -> ~q_L ~g
Pythia.SetMSTP(61, 1);          //ISR "on"
```

```
Pythia.SetMSTP(71, 1); // FSR "on"
Pythia.SetMSTP(81,0);
//seleccionando multiiples interacciones
Pythia.SetMSTP(111,1); //fragmentacion on
Pythia.SetMSTP(82, 3);
//multiple interaction "vary impact param"
Pythia.SetPARP(82, 2.41);
//cut-off pt para multiples interaciones
Pythia.SetMRPY(1, 88158204); //semilla numero aleatorio
Pythia.SetIMSS(1,2); //aproximacion SUGRA
Pythia.SetMSTJ(11, 3);
//selecion funcion fragmentacion "Bowler"
Pythia.SetMSTJ(22, 2);
//escoge decaimiento particulas inestables
Pythia.SetMSTJ(1, 1);
//hadronizacion partonic y produccion parti en estado final
Pythia.SetPARJ(50+4, -0.07);    //parametro peterson charm
Pythia.SetPARJ(50+5, -0.006);   //parametro peterson bottom
Pythia.SetPARJ(50+5, -0.000001);//parametro peterson top

//=========================================================
// PARAMETROS MSUGRA  (usando HM1)           =
//=========================================================
Pythia.SetRMSS(8,180);  //masa escalar m0 (squark)
Pythia.SetRMSS(1,850);  //m1/2 masa gaugino
Pythia.SetRMSS(5,10);  //tanbeta (chargino y neutralino)
Pythia.SetRMSS(4,100); //mu
Pythia.SetRMSS(16,0); //A0 coupling
//=========================================================
//  CANALES DE DECAIMIENTO    =
//=========================================================
for (int i=1; i<4352; i++) {
//desabilita todos los canales existentes
Pythia.SetMDME(i,1,0);
}
// primero se deshabilita
for (int i=1975; i<2162; i++) {
Pythia.SetMDME(i,1,0);
}
// Forzar ~g-> ~q_L + q_bar
Pythia.SetMDME(1976,1,1); //~g-> ~d_L + dbar
        Pythia.SetMDME(1977,1,1); //~g-> ~d_Lbar + dbar
        Pythia.SetMDME(1980,1,1); //~g-> ~u_L + ubar
        Pythia.SetMDME(1981,1,1); //~g-> ~u_Lbar + u
        Pythia.SetMDME(1984,1,1); //~g-> ~s_L + sbar
        Pythia.SetMDME(1985,1,1); //~g-> ~s_Lbar + s
        Pythia.SetMDME(1988,1,1); //~g-> ~c_L + cbar
        Pythia.SetMDME(1989,1,1); //~g-> ~c_Lbar + c

//  primero se deshabilita
for (int i=1592; i<1802; i++) {
Pythia.SetMDME(i,1,0);
}
//   Forzar ~q_L-> q + ~chi_1+-
Pythia.SetMDME(1593,1,1); //~d_L-> u + ~chi_1-
Pythia.SetMDME(1638,1,1); //~u_L-> d + ~chi_1+
Pythia.SetMDME(1662,1,1); //~s_L-> c + ~chi_1-
Pythia.SetMDME(1707,1,1); //~c_L-> s + ~chi_1+

//  primero se deshabilita
for (int i=2595; i<2826; i++) {
Pythia.SetMDME(i,1,0);
}
//   Forzar ~chi_1+- -> ~chi_10 + W+-
Pythia.SetMDME(2597,1,1); //u + ~chi_1+-
```

```
//  primero se deshabilita
for (int i=190; i<210; i++) {
Pythia.SetMDME(i,1,0);
}
//   Forzar W+- -> d_bar + c
Pythia.SetMDME(208,1,1); //u + ~chi_1+-
//=========================================================
// Particulas finales estables  =
//=========================================================
Pythia.SetMDCY(15,1,0); // tau estables
Pythia.SetMDCY(310,1,0); // chi_10 estable
//=========================================================
// INICIALIZA LA  COLISION  =
//=========================================================
Pythia.Initialize("CMS", "p", "p", 14000.0);
TStopwatch ioTime;
ioTime.Stop();
//=========================================================
// GENERATOR EVENTS  =
//=========================================================
for ( i=0; i<NEvents; i++ ) {
int flag1 = 0;
int flag2 = 0;
int flag3 = 0;
int flag4 = 0;
int flag5 = 0;
int flag6 = 0;
int flag7 = 0;
int flag8 = 0;
Double_t ang,ang2,ang3,ang4,angulo,angulo2,angulo3,angulo4;
Double_t px1,px2,px3,px4,px5,px6,px7,px8,py1,py2,py3,py4,
py5,py6,py7,py8,pz1,pz2,pz3,pz4,pz5,pz6,pz7,pz8;
Double_t E1,E2,E3,E4,E5,E6,E7,E8,p1,p2,p3,p4,p5,p6,p7,p8;
Double_t c = 1.;
  Double_t pi = 4*atan(1.);
Double_t ET=0;
if ( i%1 == 0 ) cout << "Event No.: " << i << endl;
Pythia.GenerateEvent();
for (Int_t nparti=1; nparti<=Pythia.GetN(); nparti++)
{
//================= PADRES  ====================
if(Pythia.GetK(Pythia.GetK(nparti,3),2)!= 24)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!=
1000024)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3)
,3),3),2)!= 1000001 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3)
,3),2)!= 1000002 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3)
,3),2)!= 1000003 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3)
,3),2)!= 1000004 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3)
,3),2)!= 1000005 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3)
,3),2)!= 1000006)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(nparti,3),3),3),3),2)!= 1000021)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(nparti,3),3),3),3),3),2)!= 0)
```

```
{
//=== Quiero tau de 1Pa=W,2Pa=Chi_1+-,3Pa=~q,4Pa=~g, 5Pa =0
if(Pythia.GetK(nparti,2)== 15 || Pythia.GetK(nparti,2)
== -15)
{
px1 = Pythia.GetP(nparti,1);
py1 = Pythia.GetP(nparti,2);
pz1 = Pythia.GetP(nparti,3);
E1 = Pythia.GetP(nparti,4);
p1 = sqrt(px1*px1+py1*py1+pz1*pz1);
flag1 = 1;
cout<<Pythia.GetK(nparti,2)<<"\t 1Pa ="<<Pythia.GetK
(Pythia.GetK(nparti,3),2)<<"\n";
cout<<"\t 2Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK
(nparti,3),3),2)<<"\n";
cout<<"\t 3Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK
(Pythia.GetK(nparti,3),3),3),2);
cout<<"\t 4Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK
(Pythia.GetK(Pythia.GetK(nparti,3),3),3),3),2);
cout<<"\t 5Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK
(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),3)
,3),3),2)<<"\n";  //ver jets 1Pa= primer padre
momento1->Fill(p1,1.);
ener1->Fill(E1,1.);
}
// Quiero nu_tau de 1Pa=W,2Pa=Chi_1+-,3Pa=~q,4Pa=~g, 5Pa =0
if(Pythia.GetK(nparti,2)== 16||Pythia.GetK(nparti,2)== -16)
{
px2 = Pythia.GetP(nparti,1);
py2 = Pythia.GetP(nparti,2);
pz2 = Pythia.GetP(nparti,3);
E2 = Pythia.GetP(nparti,4);
p2 = sqrt(px2*px2+py2*py2+pz2*pz2);
flag2 = 1;
cout<<Pythia.GetK(nparti,2)<<"\t 1Pa ="<<Pythia.GetK(Pythia.
GetK(nparti,3),2)<<"\n";
cout<<"\t 2Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,
3),3),2)<<"\n";
cout<<"\t 3Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(nparti,3),3),3),2);
cout<<"\t 4Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(nparti,3),3),3),3),2);
cout<<"\t 5Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),3),3),3),2)<<"\n";
//ver jets 1Pa= primer padre
momento2->Fill(p2,1.);
ener2->Fill(E2,1.);
}
}
}
}
}
}


//=========================================================
// 2-Padres  =
//=========================================================
if(Pythia.GetK(Pythia.GetK(nparti,3),2)!= 1000024 )
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!=
1000001 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!=
1000002 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!=
```

```
1000003 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!=
1000004 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!=
1000005 &&
Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!=
1000006)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3)
,3),3),2)!= 1000021)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(nparti,3),3),3),3),2)!= 0)
{
//=========================================================
// Quiero W de  1Pa=Chi_1+-,2Pa=~q,3Pa=~g, 4Pa =0 =
//=========================================================
if(Pythia.GetK(nparti,2)== 24 || Pythia.GetK(nparti,2)==
-24)
{
px3 = Pythia.GetP(nparti,1);
py3 = Pythia.GetP(nparti,2);
pz3 = Pythia.GetP(nparti,3);
E3 = Pythia.GetP(nparti,4);
p3 = sqrt(px3*px3+py3*py3+pz3*pz3);
flag3 = 1;
cout<<Pythia.GetK(nparti,2)<<"\t 1Pa ="<<Pythia.GetK(Pythia
.GetK(nparti,3),2)<<"\n";
cout<<"\t 2Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti
,3),3),2)<<"\n";
cout<<"\t 3Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia
.GetK(nparti,3),3),3),2);
cout<<"\t 4Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia
.GetK(Pythia.GetK(nparti,3),3),3),3),2);
cout<<"\t 5Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia
.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),3),3),3),2)
<<"\n"; //ver jets 1Pa= primer padre
momento3->Fill(p3,1.);
ener3->Fill(E3,1.);
}
//=========================================================
// Quiero chi_10 de  1Pa=Chi_1+-,2Pa=~q,3Pa=~g, 4Pa =0
//=========================================================
if(Pythia.GetK(nparti,2)== 1000022)
{
px4 = Pythia.GetP(nparti,1);
py4 = Pythia.GetP(nparti,2);
pz4 = Pythia.GetP(nparti,3);
E4 = Pythia.GetP(nparti,4);
p4 = sqrt(px4*px4+py4*py4+pz4*pz4);
flag4 = 1;
cout<<Pythia.GetK(nparti,2)<<"\t 1Pa ="<<Pythia.GetK(Pythia.
GetK(nparti,3),2)<<"\n";
cout<<"\t 2Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti
,3),3),2)<<"\n";
cout<<"\t 3Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(nparti,3),3),3),2);
cout<<"\t 4Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(nparti,3),3),3),3),2);
cout<<"\t 5Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),3),3),3),2)<<"\n";
//ver jets 1Pa= primer padre
momento4->Fill(p4,1.);
ener4->Fill(E4,1.);
}
```

```
}
}
}
}


//=========================================================
// 3-Padres   =
//=========================================================
if(Pythia.GetK(Pythia.GetK(nparti,3),2)!= 1000001 &&
Pythia.GetK(Pythia.GetK(nparti,3),2)!= 1000002 &&
Pythia.GetK(Pythia.GetK(nparti,3),2)!= 1000003 &&
Pythia.GetK(Pythia.GetK(nparti,3),2)!= 1000004 &&
Pythia.GetK(Pythia.GetK(nparti,3),2)!= 1000005 &&
Pythia.GetK(Pythia.GetK(nparti,3),2)!= 1000006)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!=
1000021)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3)
,3),3),2)!= 0)
{
//=========================================================
// Quiero chi_1-+ de  1Pa=~q,2Pa=~g, 3Pa =0    =
//=========================================================
if(Pythia.GetK(nparti,2)== 1000024 || Pythia.GetK(nparti,
2)== -1000024)
{
px5 = Pythia.GetP(nparti,1);
py5 = Pythia.GetP(nparti,2);
pz5 = Pythia.GetP(nparti,3);
E5 = Pythia.GetP(nparti,4);
p5 = sqrt(px5*px5+py5*py5+pz5*pz5);
flag5 = 1;
cout<<Pythia.GetK(nparti,2)<<"\t 1Pa ="<<Pythia.GetK
(Pythia.GetK(nparti,3),2)<<"\n";
cout<<"\t 2Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK
(nparti,3),3),2)<<"\n";
cout<<"\t 3Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK
(Pythia.GetK(nparti,3),3),3),2);
cout<<"\t 4Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK
(Pythia.GetK(Pythia.GetK(nparti,3),3),3),3),2);
cout<<"\t 5Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK
(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),3),3),
3),2)<<"\n";//ver jets 1Pa= primer padre
momento5->Fill(p5,1.);
ener5->Fill(E5,1.);
}
//=========================================================
// Quiero q de  1Pa=~q,2Pa=~g,3Pa=0   =
//=========================================================
if(Pythia.GetK(nparti,2)== 1||Pythia.GetK(nparti,2)== 2||
Pythia.GetK(nparti,2)== 3 ||Pythia.GetK(nparti,2)== 4 ||
Pythia.GetK(nparti,2)== 5 ||Pythia.GetK(nparti,2)== 6 )
{
px6 = Pythia.GetP(nparti,1);
py6 = Pythia.GetP(nparti,2);
pz6 = Pythia.GetP(nparti,3);
E6 = Pythia.GetP(nparti,4);
p6 = sqrt(px6*px6+py6*py6+pz6*pz6);
flag6 = 1;
cout<<Pythia.GetK(nparti,2)<<"\t 1Pa ="<<Pythia.GetK(Pythia.
GetK(nparti,3),2)<<"\n";
cout<<"\t 2Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,
3),3),2)<<"\n";
```

```
cout<<"\t 3Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(nparti,3),3),3),2);
cout<<"\t 4Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(nparti,3),3),3),3),2); cout<<"\t 5Pa "
<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(nparti,3),3),3),3),3),2)<<"\n";
//ver jets 1Pa= primer padre
momento6->Fill(p6,1.);
ener6->Fill(E6,1.);
}
}
}
}
//=========================================================
// 4-Padres   =
//=========================================================
if(Pythia.GetK(Pythia.GetK(nparti,3),2)!= 1000021)
{
if(Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),2)!= 0)
{

//=========================================================
// Quiero ~q de  1Pa=~q,2Pa=0,    =
//=========================================================
if(Pythia.GetK(nparti,2)== 1000001 ||
Pythia.GetK(nparti,2)== 1000002 || Pythia.GetK(nparti,2)==
1000003 ||
Pythia.GetK(nparti,2)== 1000004 || Pythia.GetK(nparti,2)==
1000005 || Pythia.GetK(nparti,2)== 1000006 )
{
px7 = Pythia.GetP(nparti,1);
py7 = Pythia.GetP(nparti,2);
pz7 = Pythia.GetP(nparti,3);
E7 = Pythia.GetP(nparti,4);
p7 = sqrt(px7*px7+py7*py7+pz7*pz7);
flag7 = 1;
cout<<Pythia.GetK(nparti,2)<<"\t 1Pa ="<<Pythia.GetK(Pythia.
GetK(nparti,3),2)<<"\n";
cout<<"\t 2Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,
3),3),2)<<"\n";
cout<<"\t 3Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(nparti,3),3),3),2);
cout<<"\t 4Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(nparti,3),3),3),3),2);
cout<<"\t 5Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),3),3),3),2)
<<"\n";//ver jets 1Pa= primer padre
momento7->Fill(p7,1.);
ener7->Fill(E7,1.);
}
//=========================================================
// Quiero q_bar de  1Pa=~g,2Pa=0   =
//=========================================================
if(Pythia.GetK(nparti,2)== -1||Pythia.GetK(nparti,2)== -2||
Pythia.GetK(nparti,2)== -3 || Pythia.GetK(nparti,2)== -4 ||
Pythia.GetK(nparti,2)== -5 || Pythia.GetK(nparti,2)== -6)
{
px8 = Pythia.GetP(nparti,1);
py8 = Pythia.GetP(nparti,2);
pz8 = Pythia.GetP(nparti,3);
E8 = Pythia.GetP(nparti,4);
p8 = sqrt(px8*px8+py8*py8+pz8*pz8);
flag8 = 1;
cout<<Pythia.GetK(nparti,2)<<"\t 1Pa ="<<Pythia.GetK(Pythia.
GetK(nparti,3),2)<<"\n";
```

```
cout<<"\t 2Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(nparti,
3),3),2)<<"\n";
cout<<"\t 3Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(nparti,3),3),3),2);
cout<<"\t 4Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(nparti,3),3),3),3),2);
cout<<"\t 5Pa "<<Pythia.GetK(Pythia.GetK(Pythia.GetK(Pythia.
GetK(Pythia.GetK(Pythia.GetK(nparti,3),3),3),3),3),2)<<"\n";
//ver jets 1Pa= primer padre
momento8->Fill(p8,1.);
ener8->Fill(E8,1.);
}
}
}
//=========================================================
} //--> fin del loop de particulas
if(flag1==1 || flag2==1){
ang=acos((px1*px2+py1*py2+pz1*pz2)/(p1*p2));
Double_t angulo=ang*57.296;
// cout<<"P1= "<<p1<<"\n";
// cout<<"p2 "<<p2<<"\n";
// cout<<"angulo= "<<angulo<<"\n";
angu->Fill(angulo,1.);
}
if(flag3==1 || flag4==1){
ang2=acos((px3*px4+py3*py4+pz3*pz4)/(p3*p4));
Double_t angulo2=ang2*57.296;
// cout<<"P1= "<<p1<<"\n";
// cout<<"p2 "<<p2<<"\n";
// cout<<"angulo= "<<angulo<<"\n";
angu2->Fill(angulo2,1.);
}
if(flag5==1 || flag6==1){
ang3=acos((px5*px6+py5*py6+pz5*pz6)/(p5*p6));
Double_t angulo3=ang3*57.296;
// cout<<"P1= "<<p1<<"\n";
// cout<<"p2 "<<p2<<"\n";
// cout<<"angulo= "<<angulo<<"\n";
angu3->Fill(angulo3,1.);
}
if(flag7==1 || flag8==1){
ang4=acos((px7*px8+py7*py8+pz7*pz8)/(p7*p8));
Double_t angulo4=ang4*57.296;
// cout<<"P1= "<<p1<<"\n";
// cout<<"p2 "<<p2<<"\n";
// cout<<"angulo= "<<angulo<<"\n";
angu4->Fill(angulo4,1.);
}
//=========================================================
tree->Fill();
// Create "MCEvent" object
ioTime.Start(kFALSE);
//  Write the "whole" event into ROOT file
int runNumber = 777;
int eventNumber = i;
//  delete this "event"
ioTime.Stop();
} // Fin del loop de eventos
printf(" Full time: "); fulltime.Print();
printf(" I/O time:  "); ioTime.Print();
printf("nUsage: root LoadPythia.C 'P2ATest.C(nEvent) \n");
printf(" Donde nEvent numero total de eventos generados");
//=========================================================
// Nombres de Ejes y Lineas  =
//=========================================================
```

```
ener1->GetXaxis()->SetTitle("E_{#tau}(GeV)");
ener1->GetYaxis()->SetTitle("Entradas");
ener1->SetLineWidth(3);
ener2->GetXaxis()->SetTitle("E_{#nu_{#tau}}(GeV)");
ener2->GetYaxis()->SetTitle("Entradas");
ener2->SetLineWidth(3);
ener3->GetXaxis()->SetTitle("E_{W^{+}}(GeV)");
ener3->GetYaxis()->SetTitle("Entradas");
ener3->SetLineWidth(3);
ener4->GetXaxis()->SetTitle("E_{#chi_{1}^{0}}(GeV)");
ener4->GetYaxis()->SetTitle("Entradas");
ener4->SetLineWidth(3);
ener5->GetXaxis()->SetTitle("E_{#tau}(GeV)");
ener5->GetYaxis()->SetTitle("Entradas");
ener5->SetLineWidth(3);
ener6->GetXaxis()->SetTitle("E_{#nu_{#tau}}(GeV)");
ener6->GetYaxis()->SetTitle("Entradas");
ener6->SetLineWidth(3);
ener7->GetXaxis()->SetTitle("E_{W^{+}}(GeV)");
ener7->GetYaxis()->SetTitle("Entradas");
ener7->SetLineWidth(3);
ener8->GetXaxis()->SetTitle("E_{#chi_{1}^{0}}(GeV)");
ener8->GetYaxis()->SetTitle("Entradas");
ener8->SetLineWidth(3);
momento1->GetXaxis()->SetTitle("Pt_{#tau}(GeV)");
momento1->GetYaxis()->SetTitle("Entradas");
momento1->SetLineWidth(3);
momento2->GetXaxis()->SetTitle("Pt_{#nu_{#tau}}(GeV)");
momento2->GetYaxis()->SetTitle("Entradas");
momento2->SetLineWidth(3);
momento3->GetXaxis()->SetTitle("Pt_{W^{+}}(GeV)");
momento3->GetYaxis()->SetTitle("Entradas");
momento3->SetLineWidth(3);
momento4->GetXaxis()->SetTitle("Pt_{#chi_{1}^{0}}(GeV)");
momento4->GetYaxis()->SetTitle("Entradas");
momento4->SetLineWidth(3);
momento5->GetXaxis()->SetTitle("Pt_{#tau}(GeV)");
momento5->GetYaxis()->SetTitle("Entradas");
momento5->SetLineWidth(3);
momento6->GetXaxis()->SetTitle("Pt_{#nu_{#tau}}(GeV)");
momento6->GetYaxis()->SetTitle("Entradas");
momento6->SetLineWidth(3);
momento7->GetXaxis()->SetTitle("Pt_{W^{+}}(GeV)");
momento7->GetYaxis()->SetTitle("Entradas");
momento7->SetLineWidth(3);
momento8->GetXaxis()->SetTitle("Pt_{#chi_{1}^{0}}(GeV)");
momento8->GetYaxis()->SetTitle("Entradas");
momento8->SetLineWidth(3);
angu->GetXaxis()->SetTitle("Anglulo (grados)");
angu->GetYaxis()->SetTitle("Entradas");
angu->SetLineWidth(3);
angu2->GetXaxis()->SetTitle("Angulo (grados)");
angu2->GetYaxis()->SetTitle("Entradas");
angu2->SetLineWidth(3);
angu3->GetXaxis()->SetTitle("Anglulo (grados)");
angu3->GetYaxis()->SetTitle("Entradas");
angu3->SetLineWidth(3);
angu4->GetXaxis()->SetTitle("Angulo (grados)");
angu4->GetYaxis()->SetTitle("Entradas");
angu4->SetLineWidth(3);
//=========================================================
ener1->Write();
ener2->Write();
ener3->Write();
ener4->Write();
```

```
ener5->Write();
ener6->Write();
ener7->Write();
ener8->Write();
momento1->Write();
momento2->Write();
momento3->Write();
momento4->Write();
momento5->Write();
momento6->Write();
momento7->Write();
momento8->Write();
angu->Write();
angu2->Write();
angu3->Write();
angu4->Write();
}
```

ANEXO B.

3.2. Código energía transversal faltante

Para obtener esto solo es necesario modificar al codigo del apendice A la parte de generación de particulas por las siguientes lineas de código

```
Para obtener esto solo es necesario modificar al codigo del
apendice A la parte de generación de particulas por las
siguientes lineas de código

if (Pythia.GetK(nparti,1) != 1)continue;
        if (Pythia.GetK(nparti,2) == 1000021)continue;
        Double_t px = Pythia.GetP(nparti,1);
        Double_t py = Pythia.GetP(nparti,2);
        Double_t pz = Pythia.GetP(nparti,3);
        Double_t E = Pythia.GetP(nparti,3);
        Double_t theta = acos(pz/(sqrt(px*px+py*py+pz*pz)));
        Double_t phi = atan(py/px);
        etmx += E*sin(theta)*cos(phi);
        etmy += E*sin(theta)*sin(phi);
        Etmiss = sqrt(etmx*etmx+etmy*etmy);
        etmiss->Fill(Etmiss,1.);
        }
        tree->Fill();
        ioTime.Start(kFALSE);
```