ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА»

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра физики элементарных частиц

ДИПЛОМНАЯ РАБОТА

Сухорукова Р. В.

На тему

"Оценка возможности поиска суперсимметрии на установке АТЛАС при энергии $\sqrt{s}$ =8 ТэВ в системе центра масс."

Научные руководители:
доктор физ.-мат. наук
Бедняков В. А.,
кандидат физ.-мат. наук
Храмов Е. В.

Заведующий кафедрой
академик РАН
Матвеев В.А.

Москва, 2015

Оглавление

ВВЕДЕНИЕ

Цель и задачи

Поиск Суперсимметрии – одна из приоритетнейших задач ученых, работающих на детекторе АТЛАС Большого адронного коллайдера. Целью моей работы является оценка вероятности нахождения суперсимметричного сигнала при энегрии √s = 8 ТэВ. Ранее эта оценка была получена для энергии в √s = 7 ТэВ. В том что для энергии √s=8ТэВ еще не была получена оценка вероятности нахождения суперсимметричного сигнала состоит новизна моей задачи. Актуальность проблемы состоит в том, что полученный метод оценки вероятности может быть применен в дальнейшем и при энергии в √s = 14 ТэВ.

Мною был проведен обзор статей посвященных поиску суперсимметрии при энергии в 7 √s = ТэВ[1], методике восстановления адронных струй[2] и потерянной поперечной энергии[3] на детекторе АТЛАС. Были изучены методы работы с Монте Карло генератором $Pythia$ [4].

За пределами Стандартной модели

Несмотря на впечатляющий успех в описании экспериментов, Стандартная модель не может считаться окончательной теорией элементарных частиц[11]. У нее есть свои трудности: проблема иерархии масс, неизвестная частица темной материи, преобладание вещества над антивеществом.

Физики уверены, что она должна быть частью некоторой более глубокой теории строения микромира, той частью, которая видна в экспериментах на коллайдерах при энергиях ниже примерно √s =8 ТэВ в системе центра масс. Главная задача Большого адронного коллайдера — получить хотя бы первые намеки на то, что это за более глубокая теория.

Теоретики разработали большое число кандидатов на такую теорию. Все они, естественно, включают какие-то параметры, которые отсутствуют в Стандартной модели. Часто такие теории коллективно называют **«Новая физика»** или **«За пределами Стандартной модели»**. Я хотел бы остановиться на одном активно изучаемом варианте **«Новой физики».**

Введение в Суперсимметрию

В Стандартной модели есть четкое разделение между частицами материи и частицами-переносчиками взаимодействий[6]. Кварки и лептоны — являются фермионами, частицами со спином 1/2 (полуцелым спином), в то время как все частицы — переносчики сил (фотон, глюоны, W- и Z-частицы) являются бозонами, частицами со спином 1 (целым спином).

В рамках математических теорий, опирающихся на идею **Суперсимметрии** — симметрии между фермионами и бозонами можно рассматривать эти частицы, как части некоторого единого семейства, называемого супермультиплетом частиц. Этот супермультиплет описывает частицы, которые двигаются в **суперпространстве** — расширении обычного четырехмерного пространства-времени, к которому добавляются совершенно необычные измерения с некоммутирующими координатами. Оказывается,

если супермультиплет повернуть в этом суперпространстве, то бозоны могут превратиться в фермионы и наоборот. Иными словами, бозоны и фермионы — это лишь разные проекции на наш обычный мир единого объекта, живущего в суперпространстве.

У Суперсимметричных моделей обнаруживается редкая мощь, способность решать проблемы, которые трудно решить как-то иначе:

- близкая к нулю энергия вакуума.
- естественное возникновение хиггсовского механизма.
- устранение проблемы иерархии масс.
- более сильные свидетельства в пользу теории объединения взаимодействий.

По этой причине Суперсимметрия уже давно воспринимается не как экзотическая гипотеза, а как один из самых главных вариантов физики за пределами Стандартной модели. Частицы в Минимальной Суперсимметричной стандартной модели (MSSM): частицы Стандартной модели, пять хиггсовских бозонов, плюс полный набор суперпартнеров этих частиц. По теории Суперсимметрии частица вещества (например, электрон) входит в один супермультиплет не с известными бозонами, а с некоторой новой частицей, которая называется «скалярный суперпартнер электрона», или, коротко, «сэлектрон». Аналогичный суперпартнер есть у каждого фермиона; называется он так же, как исходная частица, только с приставкой «с-» (*смюон*, *скварк* и т. д.), а обозначается той же буквой, только с тильдой. Все суперпартнеры фермионов — бозоны. Частицы-переносчики взаимодействий (а также хиггсовские бозоны) тоже входят в свои супермультиплеты, и их суперпартнеры являются фермионами. Название частиц получается в этом случае путем добавления суффикса «-ино»: *фотино*, *хиггсино*, *глюино* и т. д.

Если бы Суперсимметрия строго выполнялась в нашем мире, массы частиц и их суперпартнеров были бы равны. Но среди экспериментально открытых элементарных частиц мы не видим ни одного примера такого суперпартнерства. Значит, Суперсимметрия — если она вообще реализуется в нашем мире — должна быть нарушена. Наиболее привлекателен для теоретиков механизм спонтанного нарушения Суперсимметрии: то есть теория формулируется симметрично, но решения, описывающие наш мир, симметрию теряют.

Существует много вариантов Суперсимметрии, и какой именно из них может существовать в нашем мире - мы не знаем. Наиболее предпочтительным является вариант Минимальной Суперсимметричной стандартной модели (МССМ). Состав МССМ приведен в табл. 1 (тильда над символом обозначает суперпартнера обычной частицы). В МССМ имеются два дублета скалярных полей с квантовыми числами (1, 2, −1) и (1, 2, 1). В табл. 1 отсутствуют переносчики гравитационных взаимодействий. В простейшей теории супергравитации к полям МССМ следует добавить также пару гравитон и гравитино (частицу со спином 3/2).

Другим важным свойством модели является нарушение Суперсимметрии. В противном случае суперпартнеры были бы вырождены по массе с обычными частицами, чего не наблюдается. При нарушении Суперсимметрии вырождение исчезает и суперпартнерыприобретают большие массы, чем и объясняется их ненаблюдение в настоящий момент. Однако след существования Суперсимметрии остается, что проявляется в связи между амплитудами различных процессов (с

участием обычных частиц и суперпартнеров) и вкладе суперпартнеров в радиационные поправки в подпороговой области. При этом конкретные предсказания зависят от деталей нарушения Суперсимметрии, но механизм нарушения в настоящий момент неизвестен.

Суперполя	Бозоны	Фермионы	$SU(3) \times SU(2) \times U_Y(1)$
Векторные:			
G^a	Глюон g^a	Глюино $\tilde{g}^a$	8 1 0
V^k	Слабые W^k $(W^{\pm}, Z)$	Вино, зино $\tilde{w}^k$ $(\tilde{w}^{\pm}, \tilde{z})$	1 3 0
V'	Гиперзаряд B (γ)	Бино $\tilde{b}(\tilde{\gamma})$	1 1 0
Материи:			
L_i	Слептоны $\begin{cases} \tilde{L}_i = (\tilde{\nu}, \tilde{e})_L \\ \tilde{E}_i = \tilde{e}_R \end{cases}$	Лептоны $\begin{cases} L_i = (\nu, e)_L \\ E_i = e_R \end{cases}$	1 2 −1
E_i			1 1 2
Q_i	Скварки $\begin{cases} \tilde{Q}_i = (\tilde{u}, \tilde{d})_L \\ \tilde{U}_i = \tilde{u}_R \\ \tilde{D}_i = \tilde{d}_R \end{cases}$	Кварки $\begin{cases} Q_i = (u, d)_L \\ U_i = u_R^c \\ D_i = d_R^c \end{cases}$	3 2 1/3
U_i			3^* 1 −4/3
D_i			3^* 1 2/3
Хиггса:			
H_1	Бозоны Хиггса $\begin{cases} H_1 \\ H_2 \end{cases}$	Хиггсино $\begin{cases} \tilde{H}_1 \\ \tilde{H}_2 \end{cases}$	1 2 −1
H_2			1 2 1

Таблица 1. Состав полей МССМ[5].

Поиск Суперсимметрии на LHC

Для характеристики суперсимметричных частиц вводится новое квантовое число, называемое R-четность. Для обычных частиц R = +1, для суперсимметричных частиц R=-1. $R = (-1)^{3B+L+2J}$, где B − барионное число, L − лептонное число и J − спиновое квантовое число. Если R-четность сохраняется, то самая легкая суперсимметричная частица должна быть стабильной, либо очень долгоживущая[9], и, кроме того, она очень слабо взаимодействует с веществом, поэтому детекторы элементарных частиц не смогут зарегистрировать ее (рис. 1).

Наличие такой частицы, однако, можно будет легко заметить косвенно — по **дисбалансу поперечного импульса** зарегистрированных частиц. Высокоэнергетические протоны сталкиваются вдоль оси, их суммарный поперечный импульс практически нулевой, а значит, суммарный импульс всех родившихся частиц — как тех, которые детекторы видят, так и тех, для которых детекторы прозрачны, — тоже будет близок к нулю. Поэтому если поперечный импульс всех измеренных частиц заметно отличается от нуля, то значит, в столкновении образовалась одна или несколько частиц, которые унесли с собой недостающий поперечный импульс. Анализ **событий с «незарегестрированным» поперечным импульсом** является одним из ключевых пунктов стратегии поиска Суперсимметрии. Вначале ведется подсчет таких событий, затем проверяется смогут ли такие события быть описаны Стандартной моделью (ведь нейтрино не регистрируется детектором и тоже уносит поперечный импульс). И наконец, если окажется, что такие события не описываются Стандартной

моделью, будет проверено, предсказания каких вариантов Суперсимметричных теорий лучше всего опишут данные.

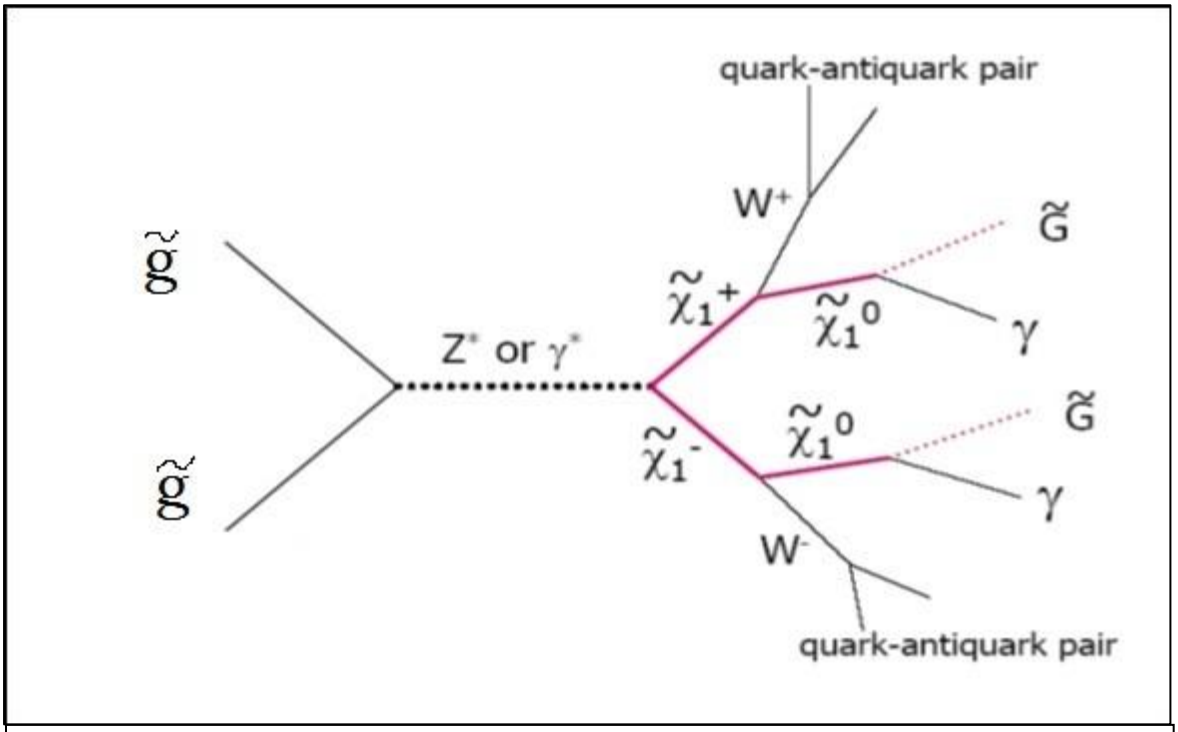

Рис 1. Пример процесса рождения пары Суперсимметричных частиц, которые распадаются на четыре адронные струи, два фотона, а также два гравитино (суперпартнер гравитона), которые уносят поперечный импульс. Изображение с сайта www-d0.fnal.go

ДЕТЕКТОР АТЛАС

Для поиска новых массивных частиц из всех событий pp соударений необходимо отбирать такие взаимодействия, в которых имело бы место жесткое столкновение составляющих протон кварков и глюонов[7]. Жесткое столкновение сопровождается большой передачей импульса и приводит к образованию частиц с большими поперечными импульсами. Такими частицами, образованными в жестких соударениях как непостредственно, так и в результате распада других частиц, являются: **кварки, глюоны, лептоны и фотоны.** Эти частицы регистрируются детектором.

Структура элементов детектора ATLAS (рис. 2):

- Магнитная система
- Внутренний детектор ATLAS
- Калориметры ATLAS
- Мюонный спектрометр детектора ATLAS
- Передние детекторы ATLAS
- Триггер детектора ATLAS

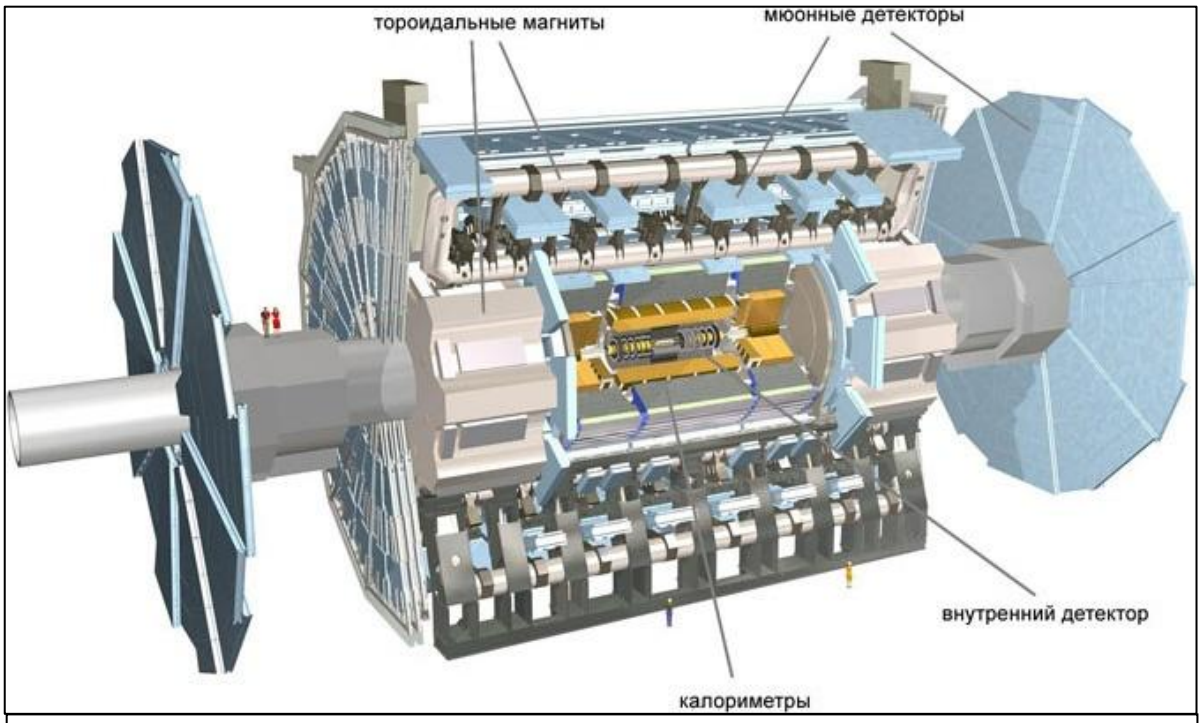

Рис 2. Детектор ATLAS, CERN. Рис. с сайта atlas.ch

Внутренний детектор

Внутренний детектор расположен ближе всего к области соударений протонов высоких энергий внутри соленоида с магнитным полем 2Т. Он представляет собой герметичную жесткую систему, задачей которой является прецизионное измерение координат и импульсов заряженных частиц и вершин первичных и вторичных взаимодействий (рис.3). Импульсы заряженных частиц измеряются выше порога 0,5

ГэВ/с в области псевдобыстрот, $\eta = -\ln\left[tg\left(\frac{\theta}{2}\right)\right]$: |η| < 2,5. Наиболее точные измерения координат необходимо проводить вблизи соударений пучков. Эту задачу выполняют пиксельные кремниевые детекторы. Они сегментированы по радиусу и азимутальному углу (R – φ) и в продольном направлении z. Каждая частица пересекает три слоя пиксельных детекторов. Элемент сенсорной ячейки пиксельных детекторов имеет размеры 50 мкм × 400 мкм. Собственное координатное разрешение пикселей в барреле составляет 10 мкм по (R – φ) и 115 мкм по продольной оси z. Для дисков это, соответственно, 10 мкм по (R – φ) и 115 мкм по R. Общее число каналов считывающей электроники пиксельных детекторов составляет 80,4 млн.

Преодолев три слоя пикселей, каждая частица пересекает восемь слоев микростриповых кремниевых детекторов (SCT), обеспечивая измерение координат четырех пространственных точек. Микростриповые кремниевые детекторы размещены, подобно пиксельным детекторам, на цилиндрах и дисках. Сенсоры детекторов имеют длину 6,4 см и период нанесения стрипов для считывания сигнала 80 мкм. Для координатных измерений стрипы ближайших слоев расположены под углом 40 мрад. При этом детекторы одного из слоев имеют стрипы с направлением вдоль оси z в барреле и по радиусу R на дисках. Число каналов электроники детекторов SCT составляет примерно 6,3 млн.

Пиксели и микростриповые кремниевые детекторы составляют дискретную трековую систему Внутреннего детектора. Её дополняет «непрерывная» трековая система, состоящая из тонких дрейфовых трубок (straw) диаметром 4 мм, расположенных близко друг к другу, и позволяющая зарегистрировать до 36 координат пересечения частицей трубок. В пространстве между трубками размещены мелко структурированные пластиковые материалы, которые обеспечивают переходное излучение заряженных частиц при пересечении ими этого множества слоев. Фотоны переходного излучения регистрируются дрейфовыми трубками наряду с сигналами от ионизационных потерь. Поэтому эта трековая система именуется детектором переходного излучения (TRT). Эффективная регистрация переходного излучения позволяет разделять треки адронов и электронов. Фотоны переходного излучения обеспечивают значительно большее энерговыделение при регистрации, чем ионизационные потери релятивистских частиц. Поэтому они легко выделяются при использовании высокого порога регистрации сигнала.

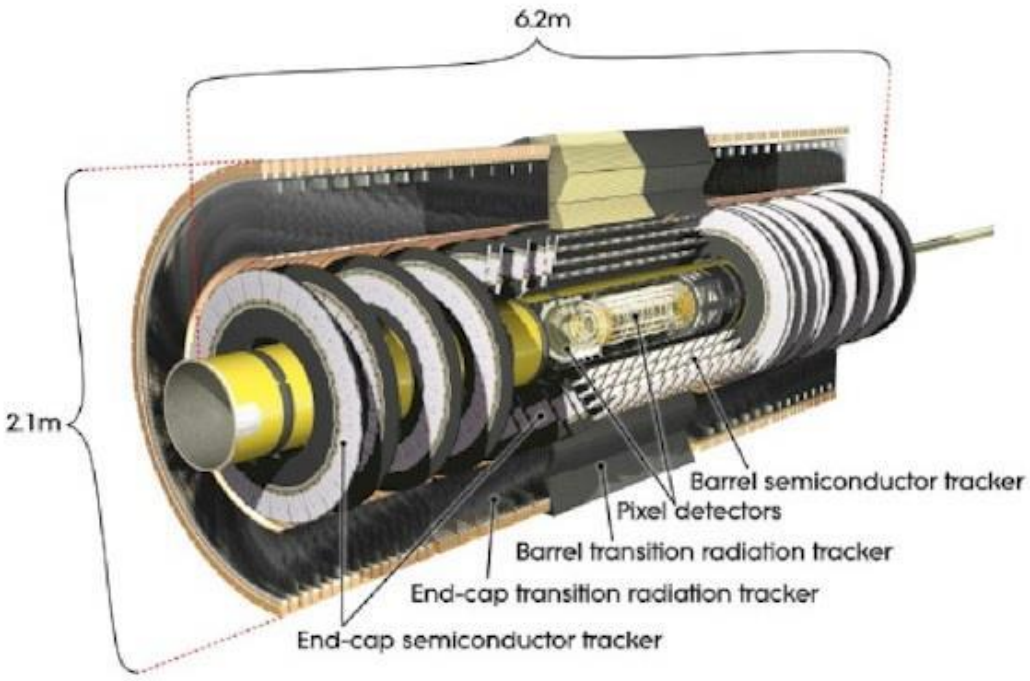

Рис 3. Внутренний детектор. Рис. с сайта atlas.ch.

Жидкоаргонный (LAr) электромагнитный калориметр (EM)

Все **калориметры детектора ATLAS (рис. 4)** являются составными. Они перекрывают область псевдобыстрот $|\eta| < 4.9$. Полная толщина электромагнитного калориметра (EM) калориметра детектора ATLAS составляет $> 22\ X_0$ в центральной части (барреле) и $> 24\ X_0$ в торцевых частях. X_0 - радиационная длина; длина за которую энергия частицы уменьшается в e раз. Толщина адронного калориметра составляет 9,7 $\lambda_{вз}$ в барреле и 10 $\lambda_{вз}$ на торцевых участках; $\lambda_{вз}$ – длина поглощения. Вместе с толщиной механических креплений 1,3 $\lambda_{вз}$ это составляет необходимые 11 $\lambda_{вз}$ для эффективного поглощения фона мюонов и надежного измерения недостающей энергии.

Электромагнитный LAr калориметр ATLAS состоит из центрального блока (барреля), соответствующего области псевдобыстрот $|\eta| < 1.475$, и двух торцевых блоков, перекрывающих области $1{,}375 < |\eta| < 3.2$. Каждый их этих трех элементов размещен в собственном криостате. Как уже отмечалось, соленоид ATLAS находится внутри барреля EM калориметра и в целях уменьшения количества вещества перед калориметром он помещен в единый вакуумный корпус с калориметром. Центральный блок электромагнитного калориметра состоит из двух идентичных частей, разделенных в центре при $z = 0$ промежутком в 4 мм. Каждый торцевой калориметр состоит из двух коаксиальных колес: внешнего, перекрывающего область $1{,}375 < |\eta| < 2{,}5$, и внутреннего, соответствующего области $2{,}5 < |\eta| < 3{,}2$.

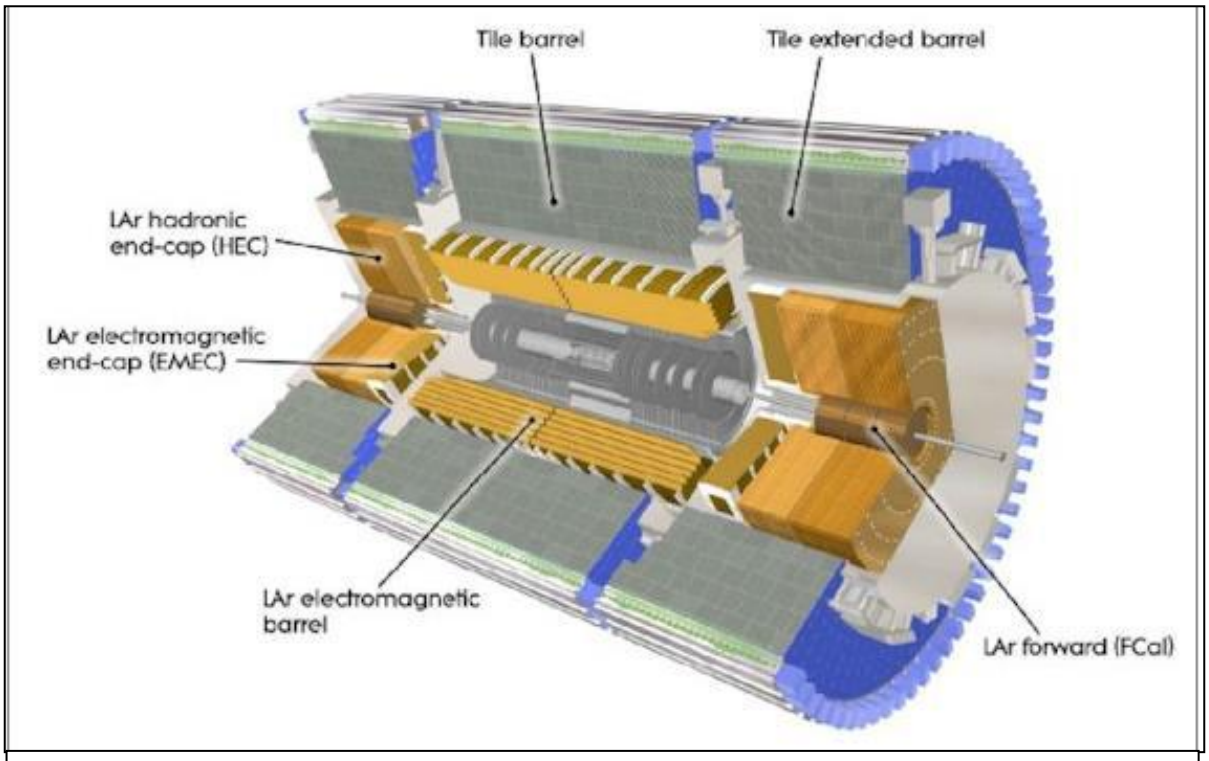

Рис. 4. Схема расположения калориметров в детекторе ATLAS. *Коричневым* показаны калориметры на жидком аргоне (LAr), *серым* — калориметры на органических сцинтилляторах. Рис. с сайта cdsweb.cern.ch

Активным веществом EM калориметра служит жидкий аргон, находящийся при температуре около 160 К, веществом поглотителя служит свинец. Слои поглотителя и электродов, выполненных из каптона, имеют ребристую форму аккордеона. Это обеспечивает полную симметрию калориметра по азимутальному углу и быстрый сбор сигнала на электроды (рис. 5). Толщина слоя абсорбера оптимизирована в зависимости от значения псевдобыстроты |η| с целью обеспечения наилучшего энергетического разрешения калориметра. Слой LAr в торцевых блоках увеличивается с радиусом калориметра. В области, отвечающей прецизионной физике и использованием Внутреннего детектора |η| < 2,5, EM калориметр разделен на три секции по глубине. Внутреннее колесо торцевого калориметра состоит из двух секций по глубине и имеет большую коаксиальную гранулярность, чем остальной калориметр.

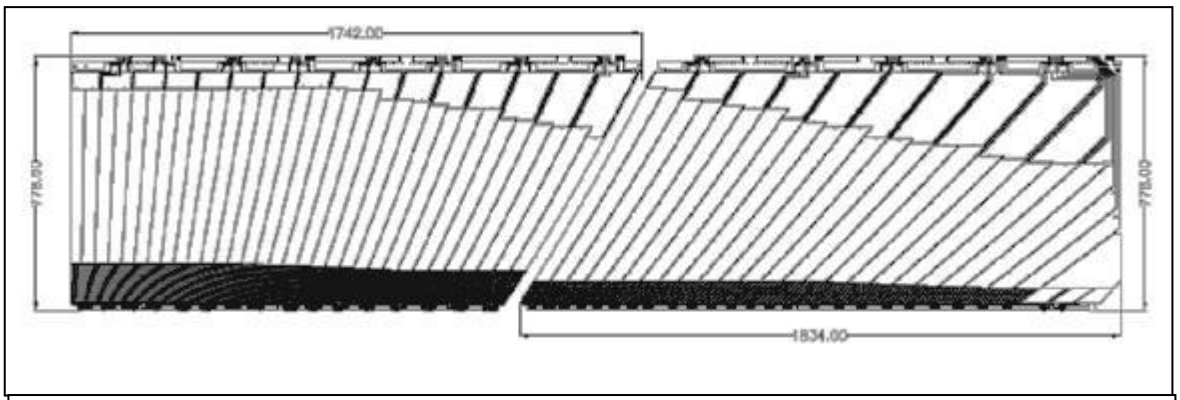

Рис.5. Рисунок расположения электродов EM калориметра $ATLAS$[7] .

Электроды считывания представляют собой три медные пластины, разделенные изолирующими полимидными слоями. Каждая половина барреля содержит 1024 абсорбера в форме аккордеона. Электроды размещены посередине расстояния между абсорберами. Дрейфовый промежуток с каждой стороны электрода равен 2,1 мм, что соответствует полному времени дрейфа 450 нс при рабочем напряжении 2000 В.

Адронный Тайл-калориметр

Адронный Тайл-калориметр размещается снаружи корпуса ЕМ калориметра. Это составной калориметр с абсорберами из стали и сцинтиллятором в качестве активного вещества (рис. 6). Он состоит из трёх блоков барреля. Центральный блок длиной 5,8 м соответствует параметрам псевдобыстроты $|\eta| < 1{,}0$. Боковые блоки расширенного барреля имеют длину 2,6 м и перекрывают области $0{,}8 < |\eta| < 1{,}7$. По азимутальному углу калориметр состоит из 64 модулей. Его внутренний радиус составляет 2,28 м, внешний 4,25 м. По радиусу калориметр сегментирован на области с толщиной 1,5 $\lambda_{вз}$, 4,1 и 1,8 $\lambda_{вз}$ в центральном блоке. В боковых блоках он разделен радиально при толщинах 1,5 $\lambda_{вз}$, 2,6 $\lambda_{вз}$ и 3,3 $\lambda_{вз}$. Пластины сцинтиллятора сделаны из полистирола, имеют толщину 3 мм, размер по азимуту варьируется от 200 до 400 мм, по радиусу от 97 до 187 мм. При пересечении заряженной частицей сцинтиллятора в его веществе продуцируется ультрафиолетовое излучение, конвертируемое в видимый свет специальными добавками в составе сцинтиллятора. Свет собирается и от каждой пластины сцинтиллятора транспортируется оптическими волокнами к фотоумножителям. Сцинтилляторы тщательно проверялись по стабильности световыхода и длины поглощения света. Все размеры конструкции выдерживались с допуском ± 0,1 мм. Испытания показали, что после 10 лет работы потери световыхода в самой радиационно загруженной области калориметра составят менее 10%.

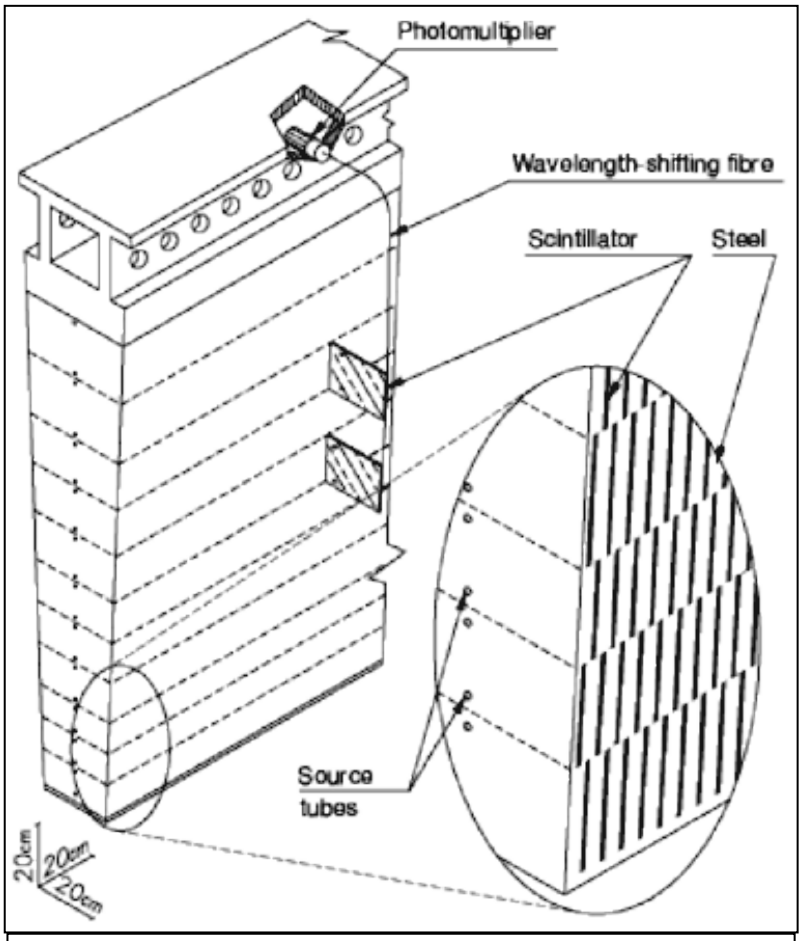

Рис. 6 Схема сектора центрального блока тайл-калориметра детектора $ATLAS$[7].

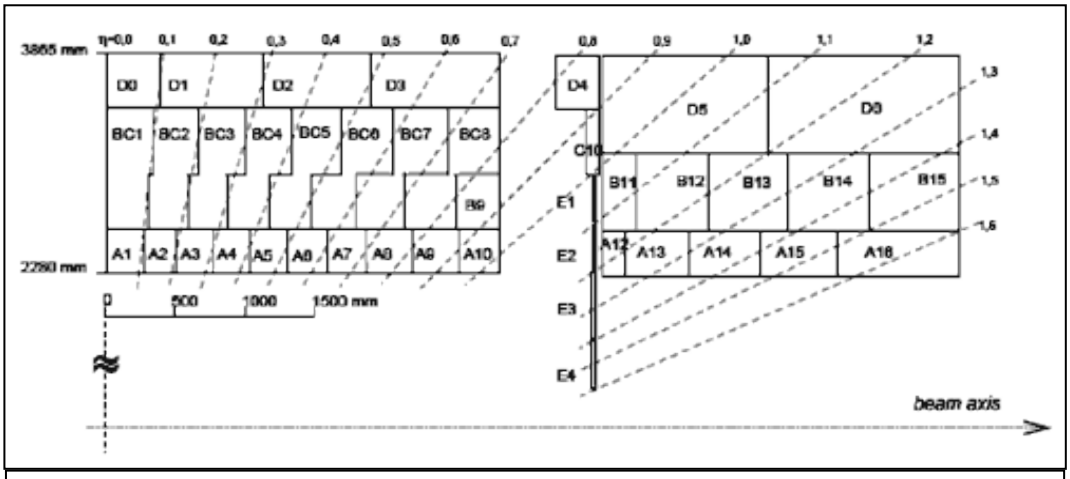

Рис. 7 Продольная сегментация тайл-калориметра центрального блока (слева) и боковых блоков[7].

Мюонный спектрометр детектора ATLAS

Детектирующие треки мюонов камеры размещены во внешнем магнитном поле, создаваемом тороидальными магнитами ATLAS[7]. Спектрометр (рис. 8) предназначен для измерения импульсов мюонов с псевдобыстротой $|\eta| < 2,7$, что перекрывает область Внутреннего детектора. При $|\eta| < 1,4$ мюоны измеряются в центральном тороиде с воздушными зазорами (барреле), в которых размещены камеры спектрометра. Магнитное поле в основном перпендикулярно направлению мюонов, что обеспечивает минимально возможное снижение разрешения за счет многократного рассеяния. Интегральная величина поля составляет 1,5 – 5,5 Тм в барреле и от 1 до 7,5 Тм для $1,6 < |\eta| < 2,7$. В области $1,4 < |\eta| < 1,6$ интеграл поля ниже из-за наложения полей магнитов. В центральной области камеры размещены в виде трёх цилиндрических слоёв, ось которых параллельна направлению протонных пучков. На торцах и в переходной области они размещаются на трёх плоскостях, перпендикулярных направлению пучка протонов. Конструкция спектрометра предназначена для работы в условиях больших потоков частиц и обеспечивает необходимые быстродействие, гранулярность и радиационную стойкость при минимальных эффектах «старения» детекторов.

Измерения координат практически во всей области псевдобыстрот проводятся с помощью прецизионных мониторируемых дрейфовых трубок (MDT). Анодные проволоки внутри трубок надежно изолированы и хорошо позиционированы, что гарантирует высокую точность их измерений. В области $1,4 < |\eta| < 1,6$, где существенно больше плотность потока частиц и высокий радиационный фон, измерения осуществляются с помощью многопроволочных пропорциональных камер с сегментированным на стрипы считывания катодом (CSC). Положение камер постоянно строго контролируется методами механической и оптической привязки.

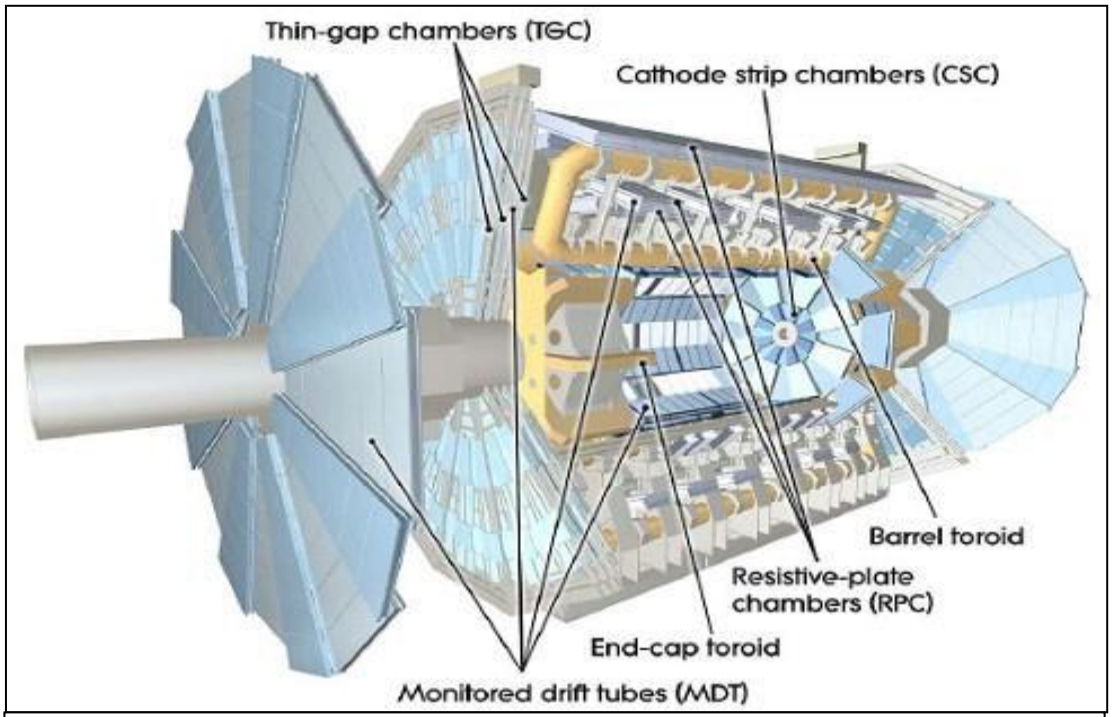

Рис. 8 Схема мюонного спектрометра ATLAS в разрезе. Обозначены положения и типы камер спектрометра и магниты, создающие внешнее поле[7].

Два типа камер (RPC и TGC) обслуживают систему триггера и перекрывают область $|\eta| < 2{,}4$. Их задачей является определение времени пересечения сгустков протонов в области соударений, точные измерения поперечных импульсов в пороговых областях для запуска триггера и определение координат треков мюонов в направлении, перпендикулярном измерениям прецизионных камер. Камеры RPC перекрывают центральную область $|\eta| < 1{,}05$. На торцах при $1{,}05 < |\eta| < 2{,}4$ установлены камеры TGC. Оба типа камер обеспечивают поступление сигнала в пределах 15-25 нс, что соответствует периоду столкновения банчей. Оба типа триггерных камер измеряют две координаты: одну в плоскости прогиба траектории (η), другую в азимутальной плоскости (φ) без прогиба траектории.

Задача спектрометра – обеспечить измерение прогиба траектории мюона с поперечным импульсом 1 ТэВ по оси z, который равен 500 мкм, с точностью ≤ 50 мкм. Нижняя граница спектра поперечных импульсов мюонов определяется величиной потерь энергии в калориметрах (порядка 3 ГэВ) и составляет несколько ГэВ. Выше этой границы мюоны могут измеряться мюонным спектрометром автономно.

Методы восстановления адронных струй

События SUSY характеризуются несколькими струями с большими поперечными импульсами и недостающей поперечной энергией[2]. Из-за большого количества струй в событиях используется алгоритм их реконструкции. Будут рассматриваться только кластерные алгоритмы, потому что один из них применяется сейчас на детекторе ATLAS. В общем случае, алгоритмы определяют расстояние между объектами и те условия при которых формирование кластера должно быть окончено. Параметр, по которому ведется образование струи, высчитывается по следующей формуле:

$$d_{ij} = \min(k_{Ti}^{2p}, k_{Tj}^{2p}) \frac{(\Delta R_{ij}^2)}{R^2}$$

$$d_{iB} = k_{Ti}^{2p}$$

где,

$$(\Delta R)_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$$

y_i- скорость объекта i ; $(\Delta R)_{ij}$- расстояние между объектами i, j; R – параметр настроек разрешения алгоритма в пределах которого пучки разделены между собой; p – изменяющийся параметр. Составляется лист с параметрами d_{ij}всех объектов. Если параметр d_{ij} минимален, то объекты i и j объединяются в один, и лист обновляется. Так происходит до тех пор пока наименьший параметр не станет равным d_{iB} - это означает что объединение завершено, и сформирована цельная струя. Теперь вернемся к параметру p:

- Если p =1, то по формуле – объекты с меньшим поперечным импульсом объединяются первыми. Это также означает, что последним сольется объект с наибольшим поперечным импульсом – эту информацию можно использовать для выявления подструктуры струи. Этот алгоритм называется $k_\perp$ (рис. 9)
- Если p=0, то d_{ij} по формуле - становится параметром зависящим только от относительного расстояния объектов i и j, поэтому объекты стоящие ближе всего друг к другу объединяются первыми, до тех пор пока их расстояние не сравнится с параметром R. Этот алгоритм называется Cambridge.
- Если p=1, то все объекты в пределах $(\Delta R)_{ij} < R$ будут объединены с наиболее энергоемким (с наибольшим поперечным импульсом) объектом. Этот алгоритм называется $anti\ k_\perp$(рис.10)

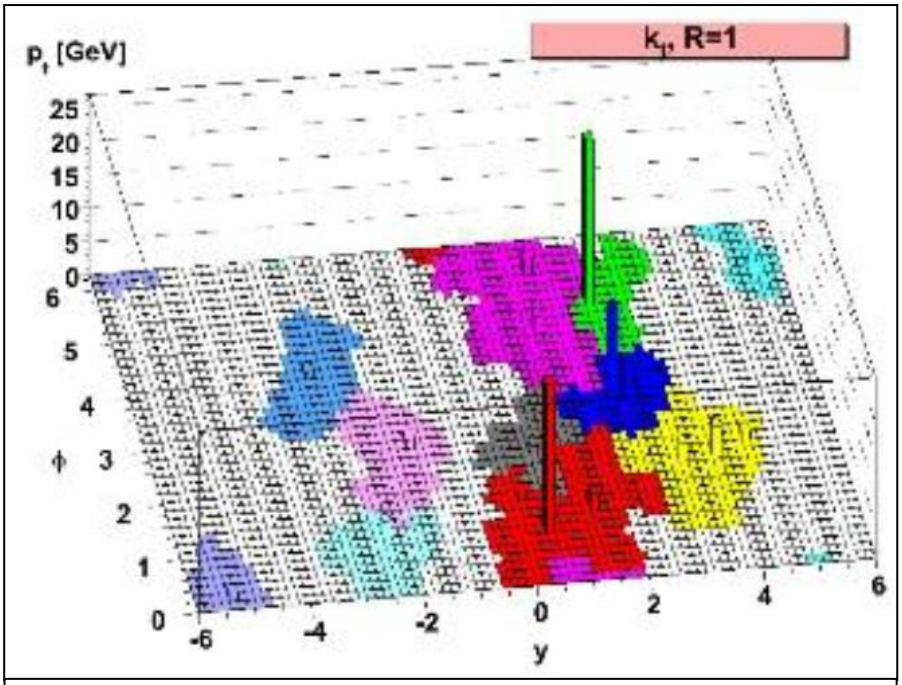

Рис 9. $k_\perp$ алгоритм восстановления адронных струй.

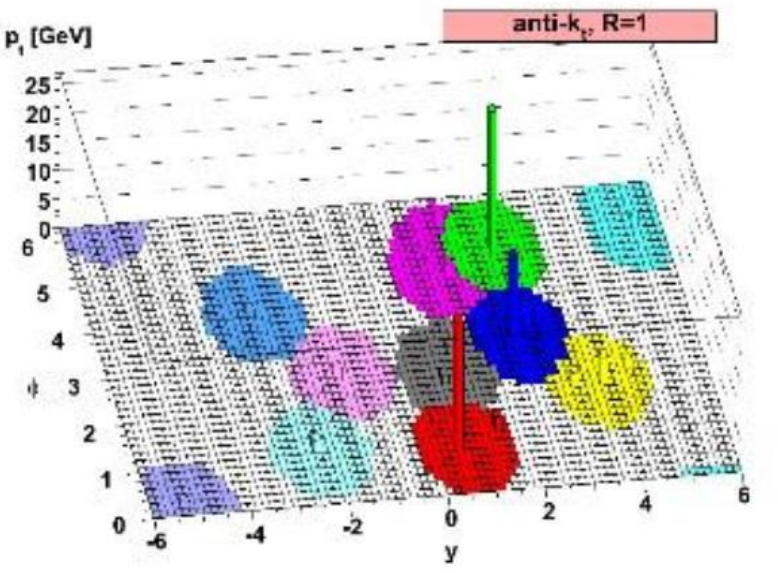

Рис 10. $anti\ k_{\perp}$ алгоритм восстановления адронных струй.

Если тяжелый объект не имеет тяжелых соседей в радиусе 2R, тогда она будет «собирать» все мягкие частицы в радиусе R. Если в пределах $R < \Delta R < 2R$, то они не смогут образовать идеальные конусы. Получаются полуконусы, которые будут разделены по границе, задаваемой формулой: $\frac{\Delta R_{1b}}{k_1} = \frac{\Delta R_{2b}}{k_2}$, где ΔRa_{1b}, ΔRa_{1b} – расстояния от границы до соответственно 1-го, 2-го тяжелого объект; k_1, k_2 – их импульсы. Именно этот алгоритм с параметром R=0,4 используется при восстановлении адронных струй на детекторе АТЛАС.

ОЦЕНКА ВЕРОЯТНОСТИ НАХОЖДЕНИЯ СУПЕРСИММЕТРИЧНОГО СИГНАЛА

Основным критерием оценки вероятности нахождения суперсимметричного сигнала будет являться статистическая значимость. Статистическая значимость – это отношение сигнала к ошибке, которую вносит фон событий Стандартной модели и систематическая погрешность. Если статистическая значимость больше трех, то можно сделать вывод, что суперсимметричный сигнал будет виден на фоне остальных событий.

Изучаемая суперсимметричная модель

Я изучал упрощенную суперсимметричную модель GG2step, которая характеризуется единственным двухшаговым каналом распада глюино:$\tilde{g} \to q\bar{q}\tilde{\chi}_2^{\mp}$ с $\tilde{\chi}_2^{\mp} \to W\tilde{\chi}_2^0$ и $\tilde{\chi}_2^0 \to Z\tilde{\chi}_1^0$.(рис11.) Конечное состояние содержит две или четыре легкие адронные струи, два W и два Z бозона и не содержит b-струй. Массы промежуточных суперсимметричных частиц взяты так, что масса $\tilde{\chi}_2^{\mp}$ лежит точно между массами глюино и массой легчайшей суперсимметричной частицы (LSP), и масса $\tilde{\chi}_2^0$ лежит точно между массами глюино и массой легчайшей суперсимметричной частицы (LSP).

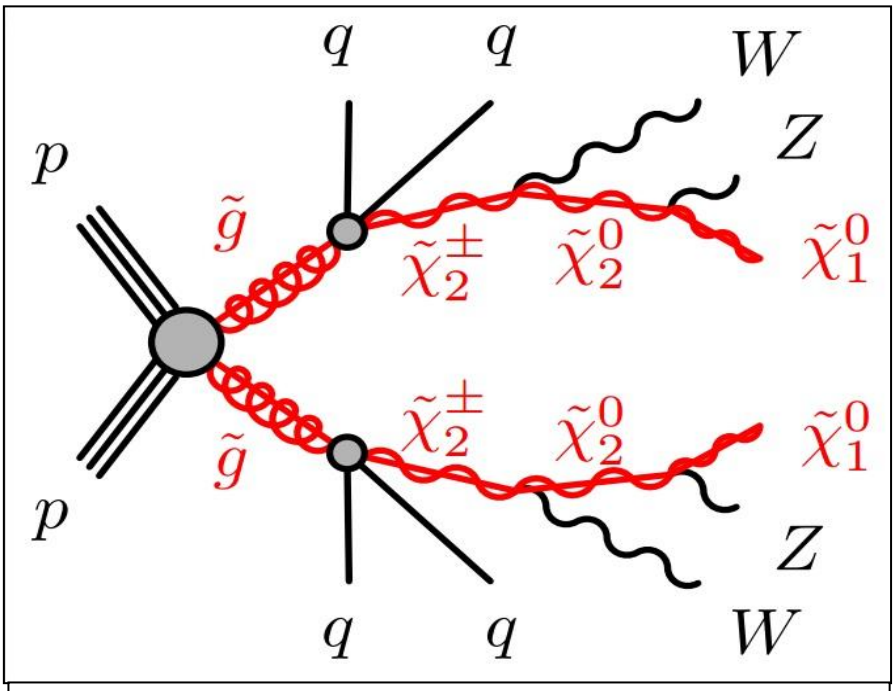

Рис. 11 Единственный канал распада глюино в модели GG2step.

Критерии отбора суперсимметричного сигнала

События SUSY характеризуются несколькими струями с большими поперечными импульсами и недостающей поперечной энергией. За основу критериев отбора были выбраны критерии, используемые для энергии √s = 7 ТэВ:

- Ровно один лептон с pt>25ГэВ.

- Как минимут 7 адронных струй с поперечным импульсом ведущей струи: pt>80 ГэВ и поперечным импульсом остальных струй: pt>25ГэВ.
- Ограничение на поперечную массу: m_T>120 ГэВ.

  $m_T = \sqrt{2p_T^l E_T^l (1 - \cos(\Delta\varphi(\vec{l}, \vec{p}_T^{miss})))}$
- Ограничение на эффективную массу: M_{eff}>750 ГэВ.

  $M_{eff} = \sum_{i=1}^{N\ lep} p_{T,i}^l + \sum_{j=1}^{N\ jet} p_{T,j} + E_T^{miss}$
- Ограничение на потерянную поперечную энергию (MET): E_T^{miss}>180 ГэВ.

Оценка фона суперсимметричного сигнала

Основными процессами в рамках Стандартной модели служащими фоном суперсимметричного сигнала, то есть так же дающих в конечном состоянии 1 лептон и не менее 7 адронных струй[1], являются:

- Прямое рождение **ttbar** пар (80.2%)
- **W(→ ℓν)** + адронные струи. (15.6%)
- **Z/γ∗(→ ℓℓ)** + адронные струи. (1.1%)
- Одиночное рождение топ кварка. (3.0%)
- Остальные процессы пренебрежимо малы.

Была написана программа, считающая число событий данного рождения и относящая его к общему числу событий. Результат, полученного отношения, приведен в скобках после каждого процесса.

Суммарный фон от этих процессов выглядит следующим образом (рис. 12):

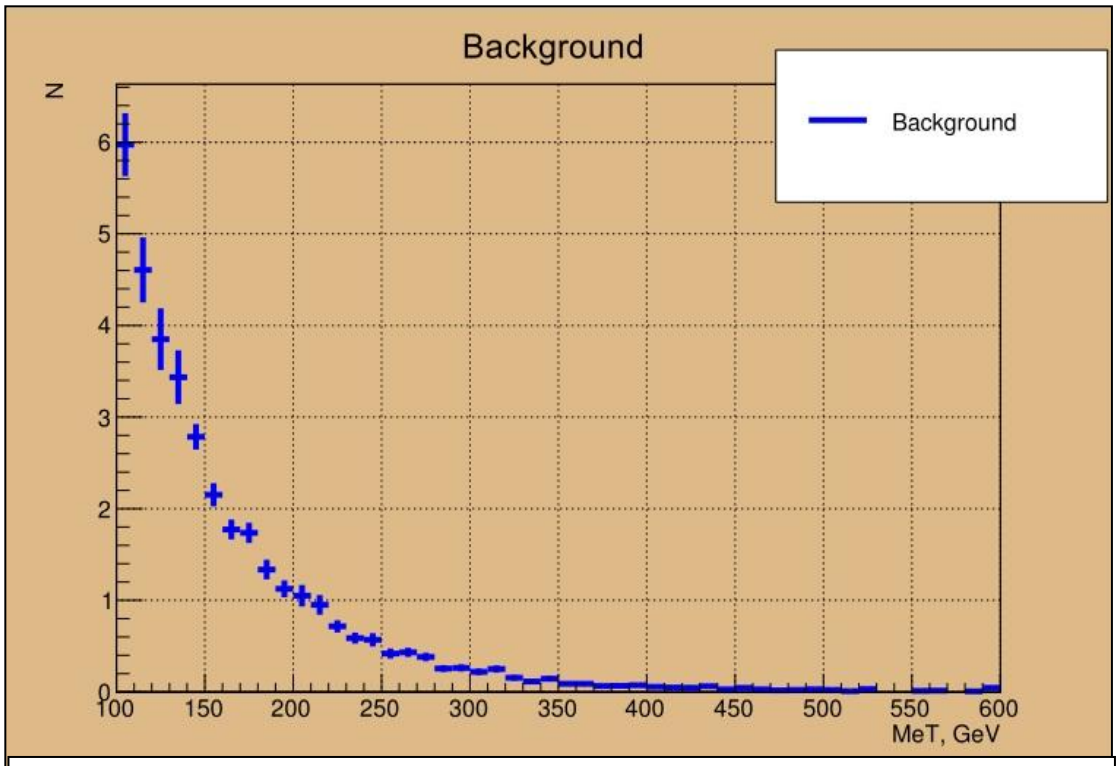

Рис. 12 График суммарного фона процессов Стандартной модели

Оценка систематической погрешности

Мною была проделана оценка величины систематической погрешности связанной с различными функциями распределения партонов. Монте-Карло генератором событий Pythia была написана программа, рассчитывающая сечение прямого рождения топ – антитоп кварковой пары с различными функциями распределения партонов (приложение 1) , так как прямое рождение топ-антитоп пары – это основное фоновое событие. Для каждой PDF генерировалось миллион событий при энергии pp взаимодействия √s = 8 ТэВ.

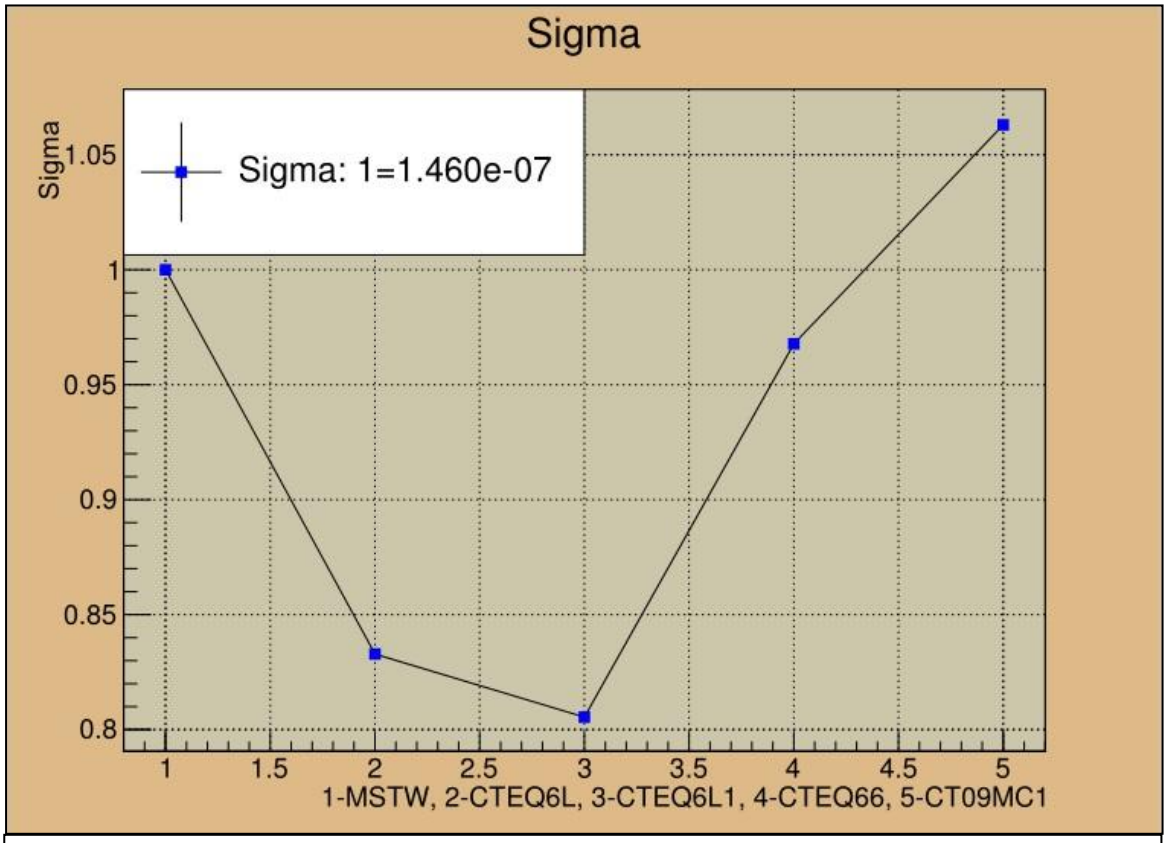

Рис 13. График зависимости сечения прямого рождения топ – антитоп пары от функции распределения партонов (приложение 1).

По графику (рис. 13) видно, что значение сечения колеблется в пределах 20% в зависимости от функции распределения партонов. Поскольку невозможно знать внутреннюю структуру взаимодействующего партона, то от 20% ошибки в дальнейших расчетах никак не избавиться. Это ошибку в дальнейшем я буду считать систематической погрешностью.

Переоптимизация критериев отбора суперсимметричного сигнала

Мною была проделана работа переоптимизации одного из этих критериев, а именно критерия на потерянную поперечную энергию (E_T^{miss}) для √s = 8 ТэВ. Для этого я использовал данные Монте Карло моделирования суперсимметричных моделей с возможными значениями масс глюино, чарджино, нейтралино и легчайшей суперсимметричной частицы. Было рассмотрено, сколько событий удовлетворяют критериям отбора без ограничения на потерянную поперечную энергию (рис. 14). Среди всех точки из всего диапазона масс глюино и LSP были выбраны те, в которых

число событие превышало 100, чтобы обеспечить ошибку: $\sqrt{N}$ менее 10%. Я рассмотрел 8 точек со следующими значениями масс глюино и LSP и получил следующее число событий:

- $M_{\tilde{g}}$ = 985, M_{LSP} = 265: N=1684.
- $M_{\tilde{g}}$ = 865, M_{LSP} = 225: N=1403.
- $M_{\tilde{g}}$ = 865, M_{LSP} = 385: N=990.
- $M_{\tilde{g}}$ = 1025, M_{LSP} = 225: N=1741.
- $M_{\tilde{g}}$ = 1065, M_{LSP} = 665: N=632.
- $M_{\tilde{g}}$ = 1225, M_{LSP} = 185: N=2191.
- $M_{\tilde{g}}$ = 1425, M_{LSP} = 465: N=2368.
- $M_{\tilde{g}}$ = 905, M_{LSP}= 505: N=565.

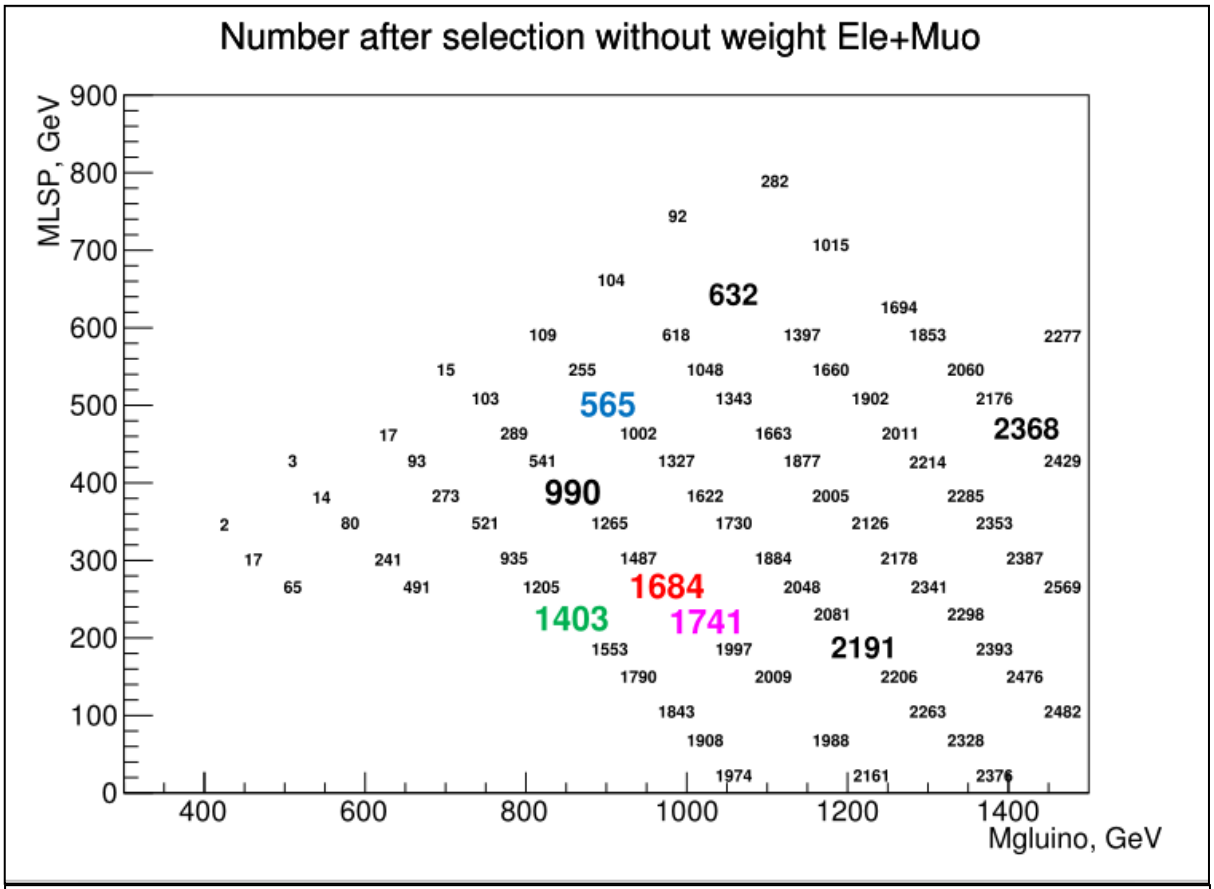

Рис. 14 Двумерная гистограмма числа событий, прошедших критерии отбора, без ограничения на потерянную поперечную энергию (приложение 3).

Следующим шагом была построена двумерная гистограмма числа событий со статистическим весом с учетом интегральной светимости большого адронного коллайдера 20,3 fb^{-1} (рис. 15). Так же построен график зависимости числа этих событий от потерянной поперечной энергии, чтобы увидеть, при каких значениях потерянной поперечной энергии эти события будут наиболее заметны на фоне событий стандартной модели (рис. 16).

Следующим шагом была построена зависимость статистической значимости от E_T^{miss} (рис. 17) по формуле:

$$Z = \frac{N_{Signal}}{\sqrt{N_{background} + 0.2N_{background}}}$$

где, N_{Signal} - число сигнальных событий, $N_{background}$ - число фоновых событий. Дополнительный член $0.2\ N_{background}$ - это учет систематической погрешности в 20%.

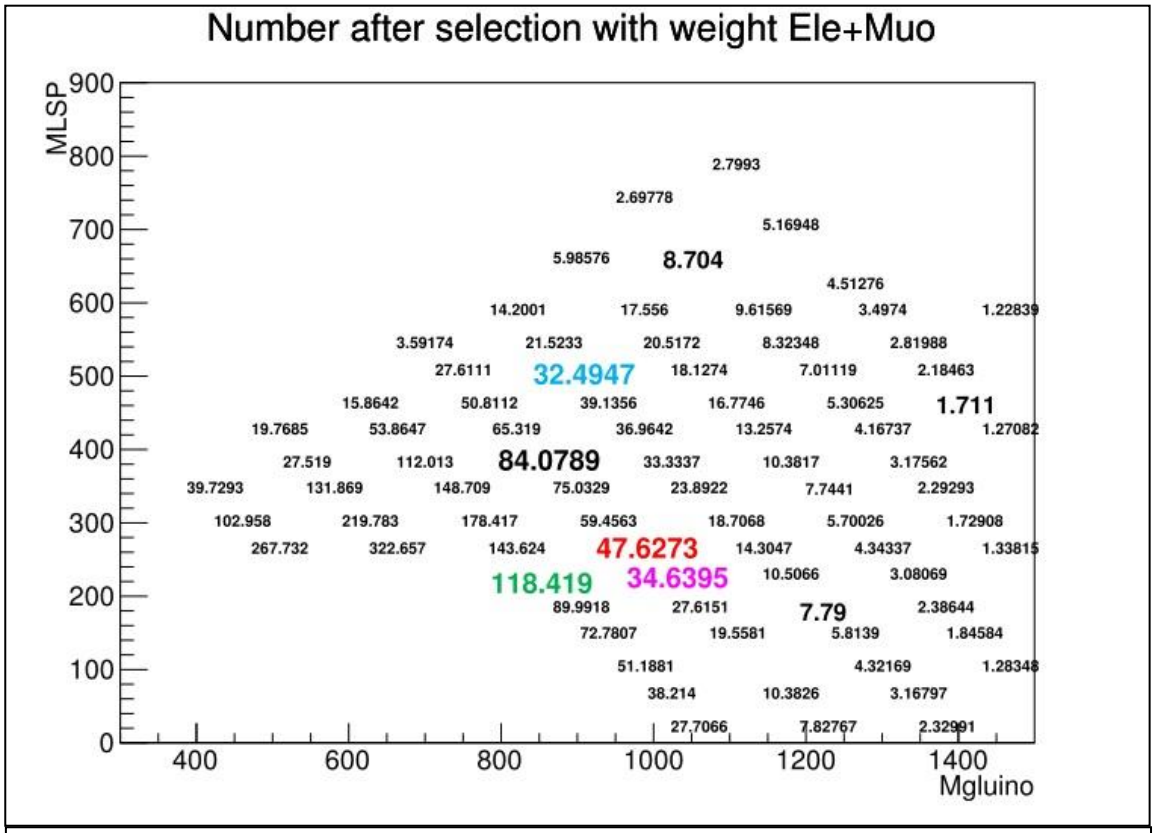

Рис.15 Двумерная гистограмма числа событий с учетом статистического веса и интегральной светимости большого адронного коллайдера 20,3 fb^{-1}, прошедших критерии отбора, без ограничения на потерянную поперечную энергию (приложение 3).

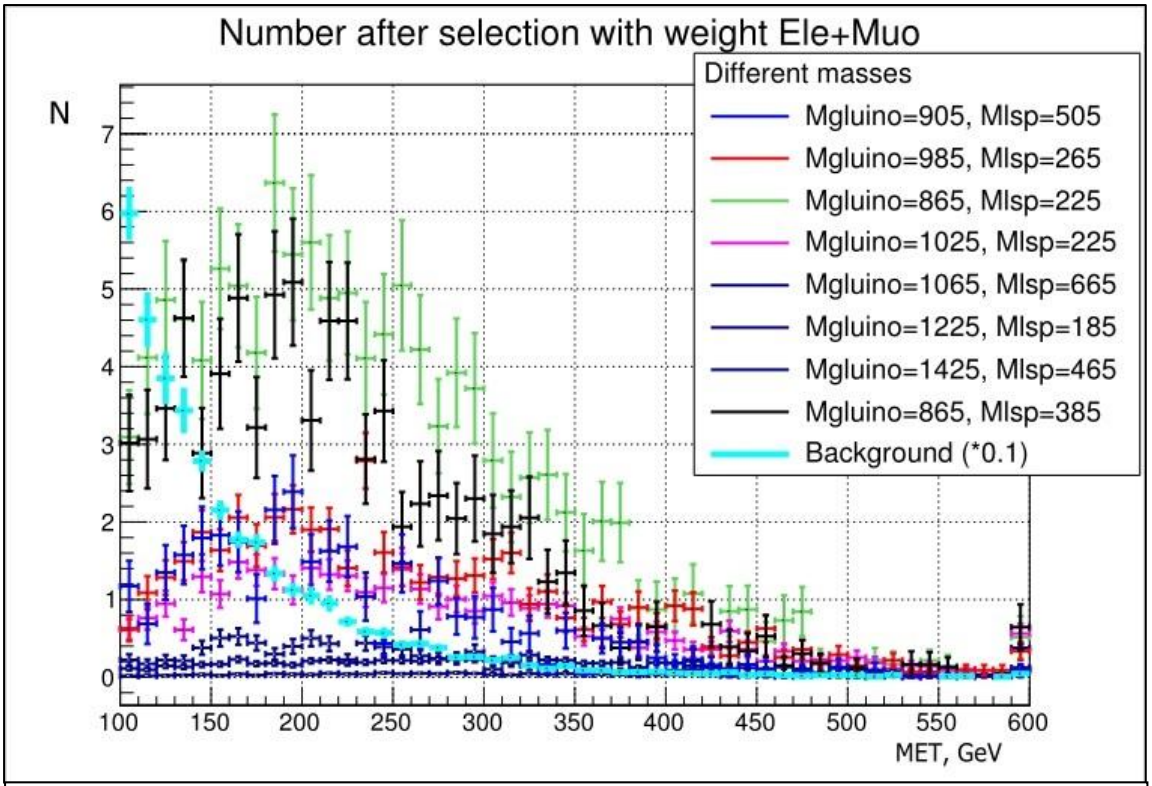

Рис. 16 Распределение числа событий с учетом статистического веса и интегральной светимости большого адронного коллайдера 20,3 fb^{-1} по потерянной поперечной энергии (приложение 3).

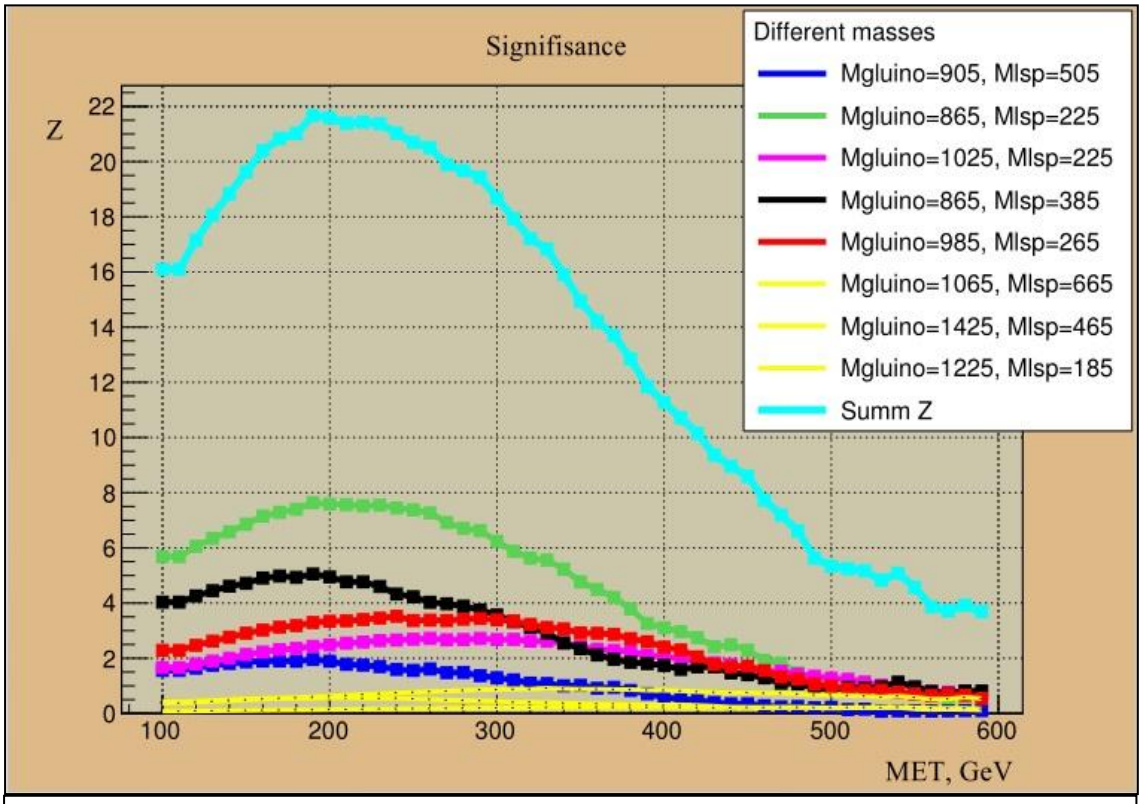

Рис. 17 Зависимость статистической значимости от потерянной поперечной энергии (приложение 3).

Максимум суммарной значимости наблюдается при значении E_T^{miss}>190 ГэВ, то есть данный критерий по E_T^{miss} является оптимальным для энергии в √s = 8 ТэВ. Полученное значение критерия для E_T^{miss} близко к критерию, взятому при рассмотрении событий с энергией в √s = 7 ТэВ: E_T^{miss}>180 ГэВ.

ЗАКЛЮЧЕНИЕ

Анализ данных с переоптимизированными критериями отбора

Взяв критерий E_T^{miss}>190 ГэВ, были получены следующие значения статистической значимости для всех точек (рис. 18). Область значений масс глюино и LSP, где Z>3 чувствительна к суперсимметричному сигналу (приложение 4).

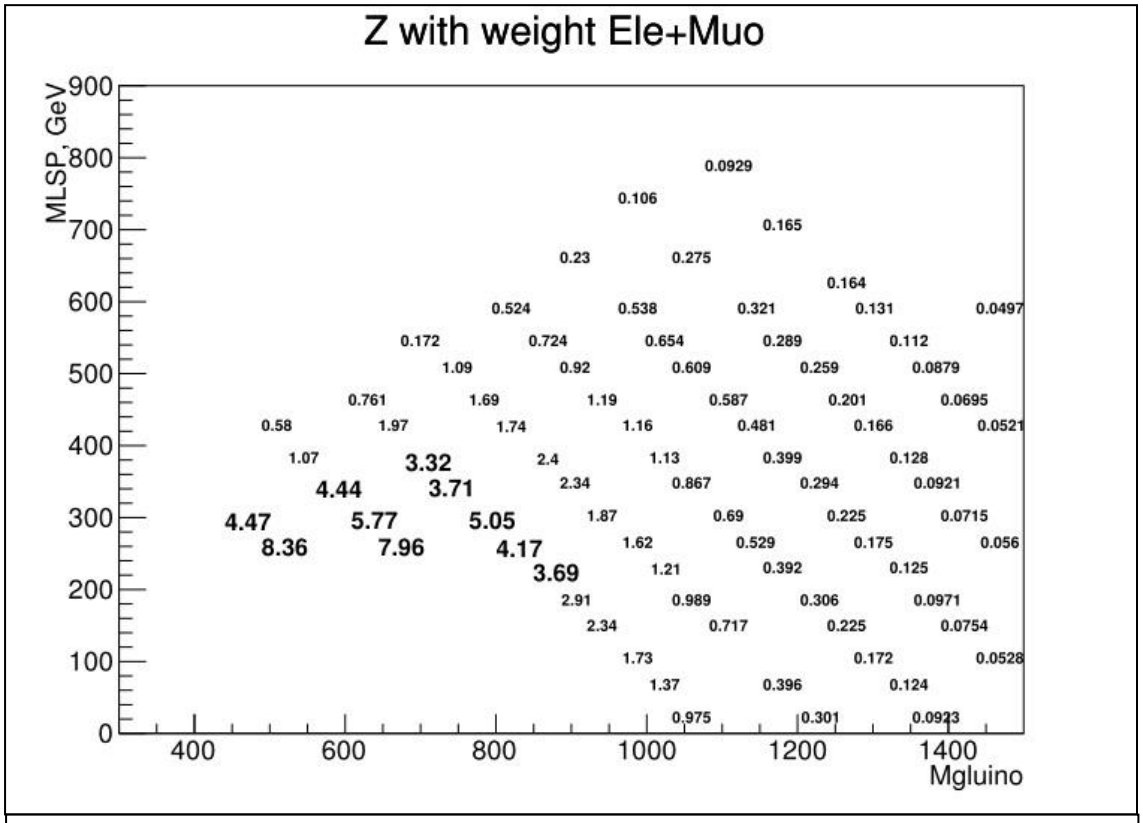

Рис. 18 Двумерная гистограмма статистической значимости, полученная при использовании критерия по MET>190 ГэВ (приложение 4).

Вывод

В ходе данной работы мною была изучена техника работы с Монте Карло генератором случайных событий Pythia. Для оценки систематической погрешности дальнейших исследований были сгенерированы события прямого рождения топ-антитоп пар кварков (приложение 1).

Была проведена работа по переоптимизации критерия отбора суперсимметричного сигнала для энергии в √s = 8 ТэВ и была написана программа для определения области значений масс глюино и LSP, чувствительной к суперсимметричному сигналу GG2step модели (приложение 2, 3).

Полученный критерий отбора E_T^{miss}>190 ГэВ позволяет определить область: 500 ГэВ < $M_{\tilde{g}}$ < 900 ГэВ, 200 ГэВ < M_{LSP} < 400 ГэВ, в которой статистическая значимость больше трех, то есть область, в которой теоретически возможно обнаружение суперсимметричного сигнала (приложение 4).

В ходе работы были освоены методы работы с объектно ориентированным пакетом анализа данных $root$[10]. Приведенный анализ и метод переоптимизации критериев отбора в дальнейшем может быть использован для обработки данных будущих экспериментов на Большом адронном коллайдере при энергии $\sqrt{s}$ =14 ТэВ.

Список использованной литературы

[1] The ATLAS Collaboration, Search for supersymmetry at √s = 7 TeV in final states with large jet multiplicity, missing transverse momentum and one isolated lepton with the ATLAS detector. October 5, 2012 (http://cds.cern.ch/record/1483511)

[2] L. Asquith, B. Brelier, Performance of Jet Algorithms in the ATLAS Detector, November 29, 2010.

[3] The ATLAS Collaboration, Performance of Missing Transverse Momentum Reconstruction in Proton-Proton Collisions at √s = 7 TeV with ATLAS, December 6, 2011 (http://arxiv.org/pdf/1108.5602v2.pdf).

[4] http://home.thep.lu.se/~torbjorn/pythia81html/Welcome.html

[5] А. В. Гладышев*, Д. И. Казаков, СУПЕРСИММЕТРИЯ И LHC, 2007 г.

[6] http://elementy.ru/LHC/HEP/SM/SUSY.

[7] Л.Н.Смирнова, ДЕТЕКТОР ATLAS БОЛЬШОГО АДРОННОГО КОЛЛАЙДЕРА, 2010 (http://window.edu.ru/resource/659/74659/files/Smirnova.pdf).

[8] http://nuclphys.sinp.msu.ru/ATLAS/atlas16.htm

[9] http://nuclphys.sinp.msu.ru/elp/elp08.htm#8.2

[10] http://root.cern.ch

[11] http://elementy.ru/LHC/HEP/SM/beyondSM

Приложения к диплому

Приложение 1

Программа, рассчитывающая сечение прямого рождения топ – антитоп кварковой пары с различными функциями распределения партонов.

```
#include "Pythia.h"
#include <fstream>
#include "TH1.h"
#include "TFile.h"
#include "TCanvas.h"
#include "TRandom.h"
#include "TLorentzVector.h"
#include "TLegend.h"
#include "TMath.h"
#include "SigmaTotal.h"
#include "TPaveText.h"


using namespace Pythia8;
int main() {

  // Generator. Process selection. Tevatron initialization. Histogram.
  Pythia pythia;
            TLorentzVector v1, v2, v3;
            pythia.readString("PDF:pSet = 6");
            pythia.readString("Beams:eCM = 8000.");
            pythia.readString("Top:gg2ttbar = on");
            pythia.init();
  for (int iEvent = 0; iEvent < 100000; ++iEvent) {
    if (!pythia.next()) continue;
    for (int i = 0; i < pythia.event.size(); ++i)
{
     if (pythia.event[i].id() == 6 && pythia.event[pythia.event[i].daughter1()].id() == 24)
     {
                        v1.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());
                        h1_1->Fill( v1.Perp());
                        h6_1->Fill( v1.Perp());

            }
              if (pythia.event[i].id() == -6 && pythia.event[pythia.event[i].daughter1()].id() == -24)
     {
                        v2.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());
                        h1_2->Fill( v2.Perp());
                        h6_2->Fill( v2.Perp());
                        v3=v1+v2;
                        h1_3->Fill( v3.Perp());
                        h6_3->Fill( v3.Perp());
            }
             if (pythia.event[i].id() == 24 && pythia.event[pythia.event[i].mother1()].id() == 6)
     {
                        h1_4->Fill( pythia.event[i].pT() );
                        h6_4->Fill( pythia.event[i].pT() );
            }
             if (pythia.event[i].id() == 5 && pythia.event[pythia.event[i].mother1()].id() == 6)
     {
                        h1_5->Fill( pythia.event[i].pT() );
                        h6_5->Fill( pythia.event[i].pT() );
            }
}

  }
            pythia.stat();
            Double_t S6 = pythia.info.sigmaGen();
            Double_t S6ERR=pythia.info.sigmaErr();
            cout <<"Sigma"<< pythia.info.sigmaGen() << "  " << pythia.info.sigmaErr() << endl;
```

```
//////////////////////////////////////////////////////////////////
            pythia.readString("PDF:pSet = 7");
            pythia.readString("Beams:eCM = 8000.");
            pythia.readString("Top:gg2ttbar = on");
            pythia.init();
  for (int iEvent = 0; iEvent < 100000; ++iEvent) {
    if (!pythia.next()) continue;
    for (int i = 0; i < pythia.event.size(); ++i)
{
     if (pythia.event[i].id() == 6 && pythia.event[pythia.event[i].daughter1()].id() == 24)
     {
                        v1.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());

                        h2_1->Fill( v1.Perp());
                        h7_1->Fill( v1.Perp());
            }
              if (pythia.event[i].id() == -6 && pythia.event[pythia.event[i].daughter1()].id() == -24)
     {

                        v2.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());
                        h2_2->Fill( v2.Perp());
                        h7_2->Fill( v2.Perp());
                        v3=v1+v2;
                        h2_3->Fill( v3.Perp());
                        h7_3->Fill( v3.Perp());
            }
             if (pythia.event[i].id() == 24 && pythia.event[pythia.event[i].mother1()].id() == 6)
     {
                        h2_4->Fill( pythia.event[i].pT() );
                        h7_4->Fill( pythia.event[i].pT() );
            }
             if (pythia.event[i].id() == 5 && pythia.event[pythia.event[i].mother1()].id() == 6)
     {
                        h2_5->Fill( pythia.event[i].pT() );
                        h7_5->Fill( pythia.event[i].pT() );
            }
}

  }

            pythia.stat();
            Double_t S7 = pythia.info.sigmaGen();
            Double_t S7ERR=pythia.info.sigmaErr();
//////////////////////////////////////////////////////////////////

pythia.readString("PDF:pSet = 8");
            pythia.readString("Beams:eCM = 8000.");
            pythia.readString("Top:gg2ttbar = on");
            pythia.init();
  for (int iEvent = 0; iEvent < 100000; ++iEvent) {
    if (!pythia.next()) continue;

    for (int i = 0; i < pythia.event.size(); ++i)
{
     if (pythia.event[i].id() == 6 && pythia.event[pythia.event[i].daughter1()].id() == 24)
     {
                        v1.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());
                        h3_1->Fill( v1.Perp());
                        h8_1->Fill( v1.Perp());
            }
              if (pythia.event[i].id() == -6 && pythia.event[pythia.event[i].daughter1()].id() == -24)
     {

                        v2.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());
                        h3_2->Fill( v2.Perp());
                        h8_2->Fill( v2.Perp());
```

```
                    v3=v1+v2;
                    h3_3->Fill( v3.Perp());
                    h8_3->Fill( v3.Perp());
            }
             if (pythia.event[i].id() == 24 && pythia.event[pythia.event[i].mother1()].id() == 6)
    {

                    h3_4->Fill( pythia.event[i].pT() );
                    h8_4->Fill( pythia.event[i].pT() );
            }
             if (pythia.event[i].id() == 5 && pythia.event[pythia.event[i].mother1()].id() == 6)
    {
                    h3_5->Fill( pythia.event[i].pT() );
                    h8_5->Fill( pythia.event[i].pT() );
            }
}

 }
            pythia.stat();
            Double_t S8 = pythia.info.sigmaGen();
            Double_t S8ERR=pythia.info.sigmaErr();

//////////////////////////////////////////////////////////////////

pythia.readString("PDF:pSet = 9");
            pythia.readString("Beams:eCM = 8000.");
            pythia.readString("Top:gg2ttbar = on");
            pythia.init();


  for (int iEvent = 0; iEvent < 100000; ++iEvent) {
    if (!pythia.next()) continue;
    for (int i = 0; i < pythia.event.size(); ++i)
{
     if (pythia.event[i].id() == 6 && pythia.event[pythia.event[i].daughter1()].id() == 24)
     {
                    v1.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());

                    h4_1->Fill( v1.Perp());
                    h9_1->Fill( v1.Perp());
            }
              if (pythia.event[i].id() == -6 && pythia.event[pythia.event[i].daughter1()].id() == -24)
    {

                    v2.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());
                    h4_2->Fill( v2.Perp());
                    h9_2->Fill( v2.Perp());

                    v3=v1+v2;
                    h4_3->Fill( v3.Perp());
                    h9_3->Fill( v3.Perp());
            }
             if (pythia.event[i].id() == 24 && pythia.event[pythia.event[i].mother1()].id() == 6)
    {
                    h4_4->Fill( pythia.event[i].pT() );
                    h9_4->Fill( pythia.event[i].pT() );
            }
             if (pythia.event[i].id() == 5 && pythia.event[pythia.event[i].mother1()].id() == 6)
    {
                    h4_5->Fill( pythia.event[i].pT() );
                    h9_5->Fill( pythia.event[i].pT() );
            }
}

 }
            pythia.stat();
            Double_t S9 = pythia.info.sigmaGen();
            Double_t S9ERR=pythia.info.sigmaErr();
```

```
//////////////////////////////////////////////////////////////////////

            pythia.readString("PDF:pSet = 10");
            pythia.readString("Beams:eCM = 8000.");
            pythia.readString("Top:gg2ttbar = on");
            pythia.init();
  for (int iEvent = 0; iEvent < 100000; ++iEvent) {
    if (!pythia.next()) continue;


    for (int i = 0; i < pythia.event.size(); ++i)
{
     if (pythia.event[i].id() == 6 && pythia.event[pythia.event[i].daughter1()].id() == 24)
     {
                        v1.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());

                        h5_1->Fill( v1.Perp());
                        h10_1->Fill( v1.Perp());


            }
              if (pythia.event[i].id() == -6 && pythia.event[pythia.event[i].daughter1()].id() == -24)
     {
                        v2.SetPxPyPzE(pythia.event[i].px(),pythia.event[i].py(),pythia.event[i].pz(),pythia.event[i].e());
                        h5_2->Fill( v2.Perp());
                        h10_2->Fill( v2.Perp());
                        v3=v1+v2;
                        h5_3->Fill( v3.Perp());
                        h10_3->Fill( v3.Perp());

            }
             if (pythia.event[i].id() == 24 && pythia.event[pythia.event[i].mother1()].id() == 6)
     {
                        h5_4->Fill( pythia.event[i].pT() );
                        h10_4->Fill( pythia.event[i].pT() );

            }
             if (pythia.event[i].id() == 5 && pythia.event[pythia.event[i].mother1()].id() == 6)
     {
                          h5_5->Fill( pythia.event[i].pT() );
                          h10_5->Fill( pythia.event[i].pT() );
            }
}

  }
            pythia.stat();
            Double_t S10 = pythia.info.sigmaGen();
            Double_t S10ERR=pythia.info.sigmaErr();
            cout <<"Sigma"<< pythia.info.sigmaGen() << "  " << pythia.info.sigmaErr() << endl;
            Double_t Mean = h1_1->GetMean();
            Double_t MeanErr = h1_1->GetMeanError();
            Double_t RMS = h1_1->GetRMS();
            Double_t RMSErr = h1_1->GetRMSError();
            SigmaTotal sigmaTot;

cout <<"*****************__________Sigma__________*************************"<<endl;
 cout<<"pdf 6: MSTW 2008 NLO:  "<<S6 << " +- " << S6ERR<< endl;
cout<<"pdf 7: CTEQ6L, NLO:  "<<S7 << " +- " << S7ERR<< endl;
cout<<"pdf 8: CTEQ6L1, LO:  "<<S8 << " +- " << S8ERR<< endl;
cout<<"pdf 9: CTEQ66.00:  "<<S9 << " +- " << S9ERR<< endl;
cout<<"pdf 10: CT09MC1, LO:  "<<S10 << " +- " << S10ERR<< endl;

  return 0;
}
```

Приложение 2

Программа, строящая график фоновых событий.

```
#include "TROOT.h"
#include "TFile.h"
#include "TTree.h"
#include "TBrowser.h"
#include "TH2.h"
#include "TRandom.h"

// СОЗДАНИЕ ФОНА

Int_t number =2000000;
void tree1r()
{
  TFile *f = new TFile("bkgtree_HardEle.root","READ");
  TTree *t1 = (TTree*)f->Get("PowhegPythiaTTbar_NoSys");
  Float_t lep2Pt, lep1Pt,  met, mt, lep1Eta;
  Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
  Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;

  Double_t random;
  Int_t ev;
  t1->SetBranchAddress("lep1Eta",&lep1Eta);
  t1->SetBranchAddress("lep2Pt",&lep2Pt);
  t1->SetBranchAddress("lep1Pt",&lep1Pt);
  t1->SetBranchAddress("met",&met);
  t1->SetBranchAddress("mt",&mt);
  t1->SetBranchAddress("genWeight",&genWeight);
  t1->SetBranchAddress("eventWeight",&eventWeight);
  t1->SetBranchAddress("leptonWeight",&leptonWeight);
  t1->SetBranchAddress("triggerWeight",&triggerWeight);
  t1->SetBranchAddress("pileupWeight",&pileupWeight);
  t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
   t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);

// Create HISTOGRAM
            TH1F *Hmet = new TH1F("Hmet","met distribution TTBAR",50,100,600);

  Long64_t nentries = t1->GetEntries();
  cout<<"Number:  "<<nentries<<endl;
  for (Long64_t i=0;i<number;i++) {
    t1->GetEntry(i);

   // Fill_DATA_with_weight
   if
(mt>120&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))
                                    {

                                                Hmet-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                                if(met>600){Hmet-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                    }

  }

 TFile *fout = new TFile("output.root","RECREATE");
            Hmet->Write();
            fout->Close();
   f->Close();

}

void Single_Top()
{

  TFile *f = new TFile("bkgtree_HardEle.root","READ");
  TTree *t1 = (TTree*)f->Get("SingleTop_NoSys");
  Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
```

```
   Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
   Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;


   Double_t random;
   Int_t ev;
   t1->SetBranchAddress("lep2Pt",&lep2Pt);
   t1->SetBranchAddress("lep1Eta",&lep1Eta);
   t1->SetBranchAddress("lep1Pt",&lep1Pt);
   t1->SetBranchAddress("met",&met);
   t1->SetBranchAddress("mt",&mt);
   t1->SetBranchAddress("genWeight",&genWeight);
   t1->SetBranchAddress("eventWeight",&eventWeight);
   t1->SetBranchAddress("leptonWeight",&leptonWeight);
   t1->SetBranchAddress("triggerWeight",&triggerWeight);
   t1->SetBranchAddress("pileupWeight",&pileupWeight);
   t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
    t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);


// Create HISTOGRAM
            TH1F *Hmet = new TH1F("Hmet","met distribution SingleTOP",50,100,600);
            Hmet->Sumw2();

   Long64_t nentries = t1->GetEntries();
   cout<<"Number:  "<<nentries<<endl;
   for (Long64_t i=0;i<number;i++) {
     t1->GetEntry(i);

    // Fill_DATA_with_weight
    if
(mt>120&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))
                                    {
                                            Hmet-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                            if(met>600){Hmet-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                    }

  }

  TFile *fout = new TFile("SingleTop_output.root","RECREATE");
            Hmet->Write();
            fout->Close();
    f->Close();

}

void Z_Jets()
{

   TFile *f = new TFile("bkgtree_HardEle.root","READ");
   TTree *t1 = (TTree*)f->Get("SherpaZMassiveBC_NoSys");
   Float_t lep2Pt, lep1Pt, lep1Eta,  met, mt;
   Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
   Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;

   Double_t random;
   Int_t ev;
   t1->SetBranchAddress("lep1Eta",&lep1Eta);
   t1->SetBranchAddress("lep2Pt",&lep2Pt);
   t1->SetBranchAddress("lep1Pt",&lep1Pt);
   t1->SetBranchAddress("met",&met);
   t1->SetBranchAddress("mt",&mt);
   t1->SetBranchAddress("genWeight",&genWeight);
   t1->SetBranchAddress("eventWeight",&eventWeight);
   t1->SetBranchAddress("leptonWeight",&leptonWeight);
```

```
   t1->SetBranchAddress("triggerWeight",&triggerWeight);
   t1->SetBranchAddress("pileupWeight",&pileupWeight);
   t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
    t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);


// Create HISTOGRAM
            TH1F *Hmet = new TH1F("Hmet","met distribution Z",50,100,600);

   Long64_t nentries = t1->GetEntries();
   cout<<"Number:  "<<nentries<<endl;
   for (Long64_t i=0;i<number;i++) {
     t1->GetEntry(i);

    // Fill_DATA_with_weight
    if
(mt>120&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))
                                    {
                                                Hmet-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                                if(met>600){Hmet-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                    }

  }

  TFile *fout = new TFile("Z_output.root","RECREATE");
            Hmet->Write();
            fout->Close();
    f->Close();


}

void W_Jets()
{

   TFile *f = new TFile("bkgtree_HardEle.root","READ");
   TTree *t1 = (TTree*)f->Get("SherpaWMassiveBC_NoSys");
   Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
   Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
   Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;


   Double_t random;
   Int_t ev;
   t1->SetBranchAddress("lep2Pt",&lep2Pt);
   t1->SetBranchAddress("lep1Pt",&lep1Pt);
   t1->SetBranchAddress("lep1Eta",&lep1Eta);
   t1->SetBranchAddress("met",&met);
   t1->SetBranchAddress("mt",&mt);
   t1->SetBranchAddress("genWeight",&genWeight);
   t1->SetBranchAddress("eventWeight",&eventWeight);
   t1->SetBranchAddress("leptonWeight",&leptonWeight);
   t1->SetBranchAddress("triggerWeight",&triggerWeight);
   t1->SetBranchAddress("pileupWeight",&pileupWeight);
   t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
    t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);


// Create HISTOGRAM
            TH1F *Hmet = new TH1F("Hmet","met distribution W",50,100,600);
            Hmet->Sumw2();
  cout<<number<<endl;
```

```
  Long64_t nentries = t1->GetEntries();
  cout<<"Number:  "<<nentries<<endl;
  for (Long64_t i=0;i<number;i++) {
    t1->GetEntry(i);

   // Fill_DATA_with_weight
   if
(mt>120&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))
                                        {
                                                Hmet-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                                if(met>600){Hmet-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        }

  }

 TFile *fout = new TFile("W_output.root","RECREATE");
            Hmet->Write();
            fout->Close();
   f->Close();


}


void tree1r_Muo()
{
  //read the Tree generated by tree1w and fill two histograms

  //note that we use "new" to create the TFile and TTree objects !
  //because we want to keep these objects alive when we leave this function.
  TFile *f = new TFile("bkgtree_HardMuo.root","READ");
  TTree *t1 = (TTree*)f->Get("PowhegPythiaTTbar_NoSys");
  Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
  Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
  Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;


  Double_t random;
  Int_t ev;
  t1->SetBranchAddress("lep2Pt",&lep2Pt);
  t1->SetBranchAddress("lep1Eta",&lep1Eta);
  t1->SetBranchAddress("lep1Pt",&lep1Pt);
  t1->SetBranchAddress("met",&met);
  t1->SetBranchAddress("mt",&mt);
  t1->SetBranchAddress("genWeight",&genWeight);
  t1->SetBranchAddress("eventWeight",&eventWeight);
  t1->SetBranchAddress("leptonWeight",&leptonWeight);
  t1->SetBranchAddress("triggerWeight",&triggerWeight);
  t1->SetBranchAddress("pileupWeight",&pileupWeight);
  t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
   t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);



// Create HISTOGRAM
            TH1F *Hmet = new TH1F("Hmet","met distribution TTBAR",50,100,600);
            Hmet->Sumw2();



  //read all entries and fill the histograms
  Long64_t nentries = t1->GetEntries();
  for (Long64_t i=0;i<number;i++) {
    t1->GetEntry(i);
```

```
    //cout<<"mt: "<<mt<<"    meffInc25_JVF25pt50: "<<meffInc25_JVF25pt50<<"    lep2Pt: "<<lep2Pt<<endl;
    /*//Fill_WEIGHT
    HgenWeight->Fill(genWeight);
    HeventWeight->Fill(eventWeight);
    HleptonWeight->Fill(leptonWeight);
    HtriggerWeight->Fill(triggerWeight);
            HpileupWeight->Fill(pileupWeight);*/

   // Fill_DATA_with_weight
   if
(mt>120&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))
                                {
                                        Hmet-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        if(met>600){Hmet-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}//
                                        //Hlep1Pt-
>Fill(lep1Pt,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        //Hlep2Pt-
>Fill(lep2Pt,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        //Hmt-
>Fill(mt,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                }

  }

  TFile *fout = new TFile("Muo_TTBAR_output.root","RECREATE");
            Hmet->Write();
            fout->Close();
    f->Close();

}

void Single_Top_Muo()
{

   TFile *f = new TFile("bkgtree_HardMuo.root","READ");
   TTree *t1 = (TTree*)f->Get("SingleTop_NoSys");
   Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
   Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
   Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;

   Double_t random;
   Int_t ev;
   t1->SetBranchAddress("lep2Pt",&lep2Pt);
   t1->SetBranchAddress("lep1Eta",&lep1Eta);
   t1->SetBranchAddress("lep1Pt",&lep1Pt);
   t1->SetBranchAddress("met",&met);
   t1->SetBranchAddress("mt",&mt);
   t1->SetBranchAddress("genWeight",&genWeight);
   t1->SetBranchAddress("eventWeight",&eventWeight);
   t1->SetBranchAddress("leptonWeight",&leptonWeight);
   t1->SetBranchAddress("triggerWeight",&triggerWeight);
   t1->SetBranchAddress("pileupWeight",&pileupWeight);
   t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
    t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);


// Create HISTOGRAM
            TH1F *Hmet = new TH1F("Hmet","met distribution SingleTOP",50,100,600);
            Hmet->Sumw2();



   Long64_t nentries = t1->GetEntries();
   for (Long64_t i=0;i<number;i++) {
    t1->GetEntry(i);
```

```
   // Fill_DATA_with_weight
   if
(mt>120&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))
                                        {
                                                Hmet-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                                if(met>600){Hmet-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        }

  }

  TFile *fout = new TFile("Muo_SingleTop_output.root","RECREATE");
            Hmet->Write();
            fout->Close();
    f->Close();


}



void Z_Jets_Muo()
{

   TFile *f = new TFile("bkgtree_HardMuo.root","READ");
   TTree *t1 = (TTree*)f->Get("SherpaZMassiveBC_NoSys");
   Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
   Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
   Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;

   Double_t random;
   Int_t ev;
   t1->SetBranchAddress("lep2Pt",&lep2Pt);
   t1->SetBranchAddress("lep1Pt",&lep1Pt);
   t1->SetBranchAddress("lep1Eta",&lep1Eta);
   t1->SetBranchAddress("met",&met);
   t1->SetBranchAddress("mt",&mt);
   t1->SetBranchAddress("genWeight",&genWeight);
   t1->SetBranchAddress("eventWeight",&eventWeight);
   t1->SetBranchAddress("leptonWeight",&leptonWeight);
   t1->SetBranchAddress("triggerWeight",&triggerWeight);
   t1->SetBranchAddress("pileupWeight",&pileupWeight);
   t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
    t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);

// Create HISTOGRAM
            TH1F *Hmet = new TH1F("Hmet","met distribution Z",50,100,600);
            Hmet->Sumw2();
  cout<<number<<endl;

   Long64_t nentries = t1->GetEntries();
   for (Long64_t i=0;i<number;i++) {
     t1->GetEntry(i);

   // Fill_DATA_with_weight
   if
(mt>120&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))
                                        {
                                                Hmet-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                                if(met>600){Hmet-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        }
```

```
  }

  TFile *fout = new TFile("Muo_Z_output.root","RECREATE");
            Hmet->Write();
            fout->Close();
   f->Close();


}

void W_Jets_Muo()
{

   TFile *f = new TFile("bkgtree_HardMuo.root","READ");
   TTree *t1 = (TTree*)f->Get("SherpaWMassiveBC_NoSys");
   Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
   Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
   Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;


   Double_t random;
   Int_t ev;
   t1->SetBranchAddress("lep2Pt",&lep2Pt);
   t1->SetBranchAddress("lep1Eta",&lep1Eta);
   t1->SetBranchAddress("lep1Pt",&lep1Pt);
   t1->SetBranchAddress("met",&met);
   t1->SetBranchAddress("mt",&mt);
   t1->SetBranchAddress("genWeight",&genWeight);
   t1->SetBranchAddress("eventWeight",&eventWeight);
   t1->SetBranchAddress("leptonWeight",&leptonWeight);
   t1->SetBranchAddress("triggerWeight",&triggerWeight);
   t1->SetBranchAddress("pileupWeight",&pileupWeight);
   t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
    t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);
// Create HISTOGRAM
            TH1F *Hmet = new TH1F("Hmet","met distribution W",50,100,600);
            Hmet->Sumw2();
  cout<<number<<endl;
   Long64_t nentries = t1->GetEntries();
   for (Long64_t i=0;i<number;i++) {
     t1->GetEntry(i);

    // Fill_DATA_with_weight
    if
(mt>120&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))
                                    {
                                            Hmet-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                            if(met>600){Hmet-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                    }

  }

  TFile *fout = new TFile("Muo_W_output.root","RECREATE");
            Hmet->Write();
            fout->Close();
   f->Close();


}
void SUMM_HIST_MUO()
{
            TH1F *Hmet10 = new TH1F("Hmet10","Summ10",50,100,600);
            TH1F *Hmet20 = new TH1F("Hmet20","Summ20",50,100,600);
            TH1F *Hmet30 = new TH1F("Hmet30","Final_Muon_Summ",50,100,600);
            TFile *f1 = new TFile("Muo_W_output.root","READ");
```

```
	TH1F *Hmet1 = (TH1F *)f1->Get("Hmet");
	TFile *f2 = new TFile("Muo_Z_output.root","READ");
	TH1F *Hmet2 = (TH1F *)f2->Get("Hmet");
	TFile *f3 = new TFile("Muo_SingleTop_output.root","READ");
	TH1F *Hmet3 = (TH1F *)f3->Get("Hmet");
	TFile *f4 = new TFile("Muo_TTBAR_output.root","READ");
	TH1F *Hmet4 = (TH1F *)f4->Get("Hmet");
	Hmet10->Add(Hmet1,Hmet2);
	Hmet20->Add(Hmet3,Hmet4);
	Hmet30->Add(Hmet10,Hmet20);
	TFile *fout = new TFile("Hist_SUMM.root","RECREATE");
	Hmet30->Write();
	fout->Close();
	}
void SUMM_HIST_Ele()
{
	TH1F *Hmet10 = new TH1F("Hmet10","Summ10",50,100,600);
	TH1F *Hmet20 = new TH1F("Hmet20","Summ20",50,100,600);
	TH1F *Hmet30 = new TH1F("Hmet30","Final_Electron_Summ",50,100,600);

	TFile *f1 = new TFile("W_output.root","READ");
	TH1F *Hmet1 = (TH1F *)f1->Get("Hmet");
	TFile *f2 = new TFile("Z_output.root","READ");
	TH1F *Hmet2 = (TH1F *)f2->Get("Hmet");
	TFile *f3 = new TFile("SingleTop_output.root","READ");
	TH1F *Hmet3 = (TH1F *)f3->Get("Hmet");
	TFile *f4 = new TFile("output.root","READ");
	TH1F *Hmet4 = (TH1F *)f4->Get("Hmet");
	Hmet10->Add(Hmet1,Hmet2);
	Hmet20->Add(Hmet3,Hmet4);
	Hmet30->Add(Hmet10,Hmet20);
	TFile *fout = new TFile("Hist_SUMM_ele.root","RECREATE");
	Hmet30->Write();
	fout->Close();
	}
void SUMM_HIST_Final()
{

	TFile *f1 = new TFile("W_output.root","READ");
	TH1F *Hmet1 = (TH1F *)f1->Get("Hmet");
	TFile *f2 = new TFile("Z_output.root","READ");
	TH1F *Hmet2 = (TH1F *)f2->Get("Hmet");
	TFile *f3 = new TFile("SingleTop_output.root","READ");
	TH1F *Hmet3 = (TH1F *)f3->Get("Hmet");
	TFile *f4 = new TFile("output.root","READ");
	TH1F *Hmet4 = (TH1F *)f4->Get("Hmet");
	TFile *f1_Muo = new TFile("Muo_W_output.root","READ");
	TH1F *Hmet1_Muo = (TH1F *)f1_Muo->Get("Hmet");
	TFile *f2_Muo = new TFile("Muo_Z_output.root","READ");
	TH1F *Hmet2_Muo = (TH1F *)f2_Muo->Get("Hmet");
	TFile *f3_Muo = new TFile("Muo_SingleTop_output.root","READ");
	TH1F *Hmet3_Muo = (TH1F *)f3_Muo->Get("Hmet");
	TFile *f4_Muo = new TFile("Muo_TTBAR_output.root","READ");
	TH1F *Hmet4_Muo = (TH1F *)f4_Muo->Get("Hmet");
	TH1F *Hmet30 = new TH1F("Hmet30","Final_Electron+Muon_Summ",50,100,600);
	TH1F *prozZ = new TH1F("prozZ","prozZ",50,100,600);
	TH1F *prozST = new TH1F("prozST","prozZ",50,100,600);
	TH1F *prozW = new TH1F("prozW","prozW",50,100,600);
	TH1F *prozTTBAR = new TH1F("prozTTBAR","prozTTBAR",50,100,600);
	TFile *f1_Final = new TFile("Hist_SUMM_ele.root","READ");
	TH1F *Hmet10 = (TH1F *)f1_Final->Get("Hmet30");
	TFile *f2_Final = new TFile("Hist_SUMM.root","READ");
	TH1F *Hmet20 = (TH1F *)f2_Final->Get("Hmet30");

	prozZ->Add(Hmet2_Muo,Hmet2);
	prozW->Add(Hmet1_Muo,Hmet1);
	prozTTBAR->Add(Hmet4_Muo,Hmet4);
	prozST->Add(Hmet3_Muo,Hmet3);
```

```
        Hmet30->Add(Hmet10,Hmet20);
        Hmet30->Sumw2();

        cout<<"Z%: "<<prozZ->Integral(0,50)/Hmet30->Integral(0,50)<<endl;
        cout<<"W%: "<<prozW->Integral(0,50)/Hmet30->Integral(0,50)<<endl;
        cout<<"TTBAR%: "<<prozTTBAR->Integral(0,50)/Hmet30->Integral(0,50)<<endl;
        cout<<"ST%: "<<prozST->Integral(0,50)/Hmet30->Integral(0,50)<<endl;


        TFile *fout = new TFile("Hist_SUMM_ele_muo.root","RECREATE");

        TCanvas  Hist_SUMM3("Hist_SUMM3_ele_muo","PDF", 800,600);
        Hist_SUMM3.SetFillColor(42);
        Hist_SUMM3.SetGrid();
        Hmet30->GetXaxis()->SetTitle("MeT, GeV");
        Hmet30->GetYaxis()->SetTitle("N");
  Hmet30->Draw();
        Hmet30->Write();

        Hist_SUMM3.Print("Final_SUMM_Ele_Muo_Background.pdf");
        fout->Close();
        }

void tree1() {
  //tree1w();
  tree1r();
  Single_Top();
  Z_Jets();
  W_Jets();
  tree1r_Muo();
  Single_Top_Muo();
  Z_Jets_Muo();
  W_Jets_Muo();
  SUMM_HIST_MUO();
  SUMM_HIST_Ele();
  SUMM_HIST_Final();
}
```

Приложение 3

Программа (сокращенная), с помощью которой проводится переоптимизация критерия отбора по MET.

```
void Muo()
{
Float_t calc, calc1;
Float_t err;

float Mgluino[npoints] = {425, 287, 345, 397, 465, 505, 477, 505, 545, 585, 625, 665, 557, 585, 625, 665, 705, 745, 785,
825, 865, 985, 905, 945, 1025, 1065, 637, 665, 705, 745, 785, 985, 825, 865, 905, 945, 1025, 1065, 1105, 1185, 1225, 717,
745, 825, 985, 865, 905, 945, 1025, 1065, 1105, 1145, 1185, 1225, 1265, 1305, 1345, 1385, 797, 825, 905, 985, 1025, 1105,
1065, 1145, 1185, 1225, 1265, 1305, 1345, 1385, 985, 1425, 1465, 1065, 1145, 1185, 1225, 1265, 1305, 1345, 1385, 1425,
1465, 1105, 1185, 1265, 1305, 1345, 1385, 1425, 1465, 1465};
float Mchargino[npoints] = {385, 275, 305, 385, 385, 385, 465, 465, 465, 465, 465, 465, 545, 545, 545, 545, 545, 545, 545,
545, 545, 545, 545, 545, 545, 545, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 705, 705, 705,
705, 705, 705, 705, 705, 705, 705, 705, 705, 705, 705, 705, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785,
785, 785, 785, 865, 785, 785, 865, 865, 865, 865, 865, 865, 865, 865, 865, 865, 945, 945, 945, 945, 945, 945, 945, 945,
1025};
float MLSP[npoints] = {345, 262, 265, 372, 305, 265, 452, 425, 385, 345, 305, 265, 532, 505, 465, 425, 385, 345, 305, 265,
225, 105, 185, 145, 65, 25, 612, 585, 545, 505, 465, 265, 425, 385, 345, 305, 225, 185, 145, 65, 25, 692, 665, 585, 425, 545,
505, 465, 385, 345,305, 265, 225, 185, 145, 105, 65, 25, 772, 745, 665, 585, 545, 465, 505, 425, 385, 345, 305, 265, 225,
185, 745, 145, 105, 665, 585, 545, 505, 465, 425, 385, 345, 305, 265, 785, 705, 625, 585, 545, 505, 465, 425, 585};

char treename[100];
TTree *tree;
TFile *f = new TFile("sigtree_HardMuo.root","READ");
for (int i; i<npoints; i++) {
```

```
sprintf(treename,"SM_GG2WWZZ_%d_%d_%d_NoSys",Mgluino[i],Mchargino[i],MLSP[i]);

  TTree *t1 = (TTree*)f->Get(treename);
  Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
  Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
  Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;
  Bool_t isQEDFSR, isNoCrackElectron, EF_mu24_j65_a4tchad_EFxe40_tclcw;
  Int_t AnalysisType;

  Double_t random;
  Int_t ev;
  t1->SetBranchAddress("isQEDFSR",&isQEDFSR);
  t1->SetBranchAddress("AnalysisType",&AnalysisType);
  t1->SetBranchAddress("isNoCrackElectron",&isNoCrackElectron);
  t1->SetBranchAddress("EF_mu24_j65_a4tchad_EFxe40_tclcw",&EF_mu24_j65_a4tchad_EFxe40_tclcw);

  t1->SetBranchAddress("lep2Pt",&lep2Pt);
  t1->SetBranchAddress("lep1Eta",&lep1Eta);
  t1->SetBranchAddress("lep1Pt",&lep1Pt);
  t1->SetBranchAddress("met",&met);
  t1->SetBranchAddress("mt",&mt);
  t1->SetBranchAddress("genWeight",&genWeight);
  t1->SetBranchAddress("eventWeight",&eventWeight);
  t1->SetBranchAddress("leptonWeight",&leptonWeight);
  t1->SetBranchAddress("triggerWeight",&triggerWeight);
  t1->SetBranchAddress("pileupWeight",&pileupWeight);
  t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
   t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
          t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);



          if (Mgluino[i]==1425 && MLSP[i]==465)
          {
          Long64_t nentries = t1->GetEntries();
          for (Long64_t j=0;j<nentries;j++)
                    {
                              t1->GetEntry(j);
                    // Fill_DATA_with_weight before selection
                              Hmet1425_Before-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                              Hmet1425_Before_WW->Fill(met);
                              if(met>600){Hmet1425_Before-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                              if(met>600){Hmet1425_Before_WW->Fill(590);}

                    // Fill_DATA_with_weight after selection
                              if ((EF_mu24_j65_a4tchad_EFxe40_tclcw ) &&mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25
pt50>750&& (abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                        {
                                                  if(met>600){Hmet1425_Full-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                                  if(met>600){Hmet1425_Full_WW->Fill(590);}
                                                  Hmet1425_Full-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                                  Hmet1425_Full_WW->Fill(met);
                                        }
                    }
          }

          if (Mgluino[i]==1225 && MLSP[i]==185)
          {
          Long64_t nentries = t1->GetEntries();
          for (Long64_t j=0;j<nentries;j++)
                    {
                              t1->GetEntry(j);
```

```
                // Fill_DATA_with_weight before selection
                        Hmet1225_Before-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                        Hmet1225_Before_WW->Fill(met);
                        if(met>600){Hmet1225_Before-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                        if(met>600){Hmet1225_Before_WW->Fill(590);}

                // Fill_DATA_with_weight after selection
                        if ((EF_mu24_j65_a4tchad_EFxe40_tclcw ) && mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25
pt50>750&& (abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                {
                                        if(met>600){Hmet1225_Full-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        if(met>600){Hmet1225_Full_WW->Fill(590);}
                                        Hmet1225_Full-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        Hmet1225_Full_WW->Fill(met);
                                }
                }
        }

        if (Mgluino[i]==1065 && MLSP[i]==665)
        {
        Long64_t nentries = t1->GetEntries();
        for (Long64_t j=0;j<nentries;j++)
                {
                        t1->GetEntry(j);
                // Fill_DATA_with_weight before selection
                        Hmet1065_Before-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                        Hmet1065_Before_WW->Fill(met);
                        if(met>600){Hmet1065_Before-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                        if(met>600){Hmet1065_Before_WW->Fill(590);}

                // Fill_DATA_with_weight after selection
                        if ((EF_mu24_j65_a4tchad_EFxe40_tclcw ) && mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25
pt50>750&& (abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                {
                                        if(met>600){Hmet1065_Full-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        if(met>600){Hmet1065_Full_WW->Fill(590);}
                                        Hmet1065_Full-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        Hmet1065_Full_WW->Fill(met);
                                }
                }
        }

        if (Mgluino[i]==865 && MLSP[i]==385)
        {
        Long64_t nentries = t1->GetEntries();
        for (Long64_t j=0;j<nentries;j++)
                {
                        t1->GetEntry(j);
                // Fill_DATA_with_weight before selection
                        Hmet865_385_Before-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                        Hmet865_385_Before_WW->Fill(met);
                        if(met>600){Hmet865_385_Before-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                        if(met>600){Hmet865_385_Before_WW->Fill(590);}

                // Fill_DATA_with_weight after selection
```

```
                        if (( EF_mu24_j65_a4tchad_EFxe40_tclcw ) &&mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25
pt50>750&& (abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                {
                                        Hmet865_385-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        if(met>600){Hmet865_385_Full-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        if(met>600){Hmet865_385_Full_WW->Fill(590);}
                                        Hmet865_385_Full-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        Hmet865_385_Full_WW->Fill(met);
                                }
                        }
                }

                if (Mgluino[i]==985 && MLSP[i]==265)
                {
                Long64_t nentries = t1->GetEntries();
                for (Long64_t j=0;j<nentries;j++)
                        {
                                t1->GetEntry(j);

                        // Fill_DATA_with_weight before selection
                                Hmet985_Before-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                Hmet985_Before_WW->Fill(met);
                                if(met>600){Hmet985_Before-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                if(met>600){Hmet985_Before_WW->Fill(590);}

                        // Fill_DATA_with_weight
                                if ((EF_mu24_j65_a4tchad_EFxe40_tclcw ) && mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25
pt50>750&& (abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                {
                                        if(met>600){Hmet985_Full-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        if(met>600){Hmet985_Full_WW->Fill(590);}
                                        Hmet985_Full-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        Hmet985_Full_WW->Fill(met);
                                }
                        }
                }


                if (Mgluino[i]==1025 && MLSP[i]==225)
                {
                Long64_t nentries = t1->GetEntries();
                for (Long64_t j=0;j<nentries;j++)
                        {
                                t1->GetEntry(j);
                        // Fill_DATA_with_weight before selection
                                Hmet1025_Before-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                Hmet1025_Before_WW->Fill(met);
                                if(met>600){Hmet1025_Before-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                if(met>600){Hmet1025_Before_WW->Fill(590);}

                        // Fill_DATA_with_weight
                                if ((EF_mu24_j65_a4tchad_EFxe40_tclcw ) && mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25
pt50>750&& (abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                {
```

```
                                        if(met>600){Hmet1025_Full-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        if(met>600){Hmet1025_Full_WW->Fill(590);}
                                        Hmet1025_Full-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                        Hmet1025_Full_WW->Fill(met);
                                }
                }
        }

        if (Mgluino[i]==905 && MLSP[i]==505)
        {
        Long64_t nentries = t1->GetEntries();
        for (Long64_t j=0;j<nentries;j++)
                {
                        t1->GetEntry(j);

                // Fill_DATA_with_weight before selection
                        Hmet905_Before-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                        Hmet905_Before_WW->Fill(met);
                        if(met>600){Hmet905_Before-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                        if(met>600){Hmet905_Before_WW->Fill(590);}

                // Fill_DATA_with_weight
                        if ((EF_mu24_j65_a4tchad_EFxe40_tclcw ) && mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25
pt50>750&& (abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                {
                                        if(met>600){Hmet905_Full_WW->Fill(590);}
                                        if(met>600){Hmet905_Full-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        Hmet905_Full_WW->Fill(met);
                                        Hmet905_Full-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                }
                }
        }

        if (Mgluino[i]==865 && MLSP[i]==225)
        {
        Long64_t nentries = t1->GetEntries();
        for (Long64_t j=0;j<nentries;j++)
                {
                        t1->GetEntry(j);

                // Fill_DATA_with_weight before selection
                        Hmet865_Before-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                        Hmet865_Before_WW->Fill(met);
                        if(met>600){Hmet865_Before_WW->Fill(590);}
                        if(met>600){Hmet865_Before-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}

                // Fill_DATA_with_weight
                        if ((EF_mu24_j65_a4tchad_EFxe40_tclcw ) && mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25
pt50>750&& (abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                {
                                        if(met>600){Hmet865_Full_WW->Fill(590);}
                                        if(met>600){Hmet865_Full-
>Fill(590,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);}
                                        Hmet865_Full_WW->Fill(met);
                                        Hmet865_Full-
>Fill(met,genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300);
                                }
```

```
                }
        }
}

}
        void SUMM_HIST_Final()
{

        TH1F *Hmet905Summ_Full_WW = new TH1F("Hmet905Summ_Full_WW","Number after selection with weight
Ele+Muo ",50,100,600);
        TH1F *Hmet985Summ_Full_WW = new TH1F("Hmet985Summ_Full_WW","Number after selection with weight
Ele+Muo ",50,100,600);
        TH1F *Hmet865Summ_Full_WW = new TH1F("Hmet865Summ_Full_WW","Number after selection with weight
Ele+Muo ",50,100,600);
        TH1F *Hmet1025Summ_Full_WW = new TH1F("Hmet1025Summ_Full_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet1065Summ_Full_WW = new TH1F("Hmet1065Summ_Full_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet1225Summ_Full_WW = new TH1F("Hmet1225Summ_Full_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet1425Summ_Full_WW = new TH1F("Hmet1425Summ_Full_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet865_385Summ_Full_WW = new TH1F("Hmet865_385Summ_Full_WW","Number after selection
with weight Ele+Muo ",50,100,600);

        TH1F *Hmet905Summ_Before_WW = new TH1F("Hmet905Summ_Before_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet985Summ_Before_WW = new TH1F("Hmet985Summ_Before_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet865Summ_Before_WW = new TH1F("Hmet865Summ_Before_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet1065Summ_Before_WW = new TH1F("Hmet1065Summ_Before_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet1225Summ_Before_WW = new TH1F("Hmet1225Summ_Before_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet1425Summ_Before_WW = new TH1F("Hmet1425Summ_Before_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet1025Summ_Before_WW = new TH1F("Hmet1025Summ_Before_WW","Number after selection with
weight Ele+Muo ",50,100,600);
        TH1F *Hmet865_385Summ_Before_WW = new TH1F("Hmet865_385Summ_Before_WW","Number after
selection with weight Ele+Muo ",50,100,600);


        TH1F *Hmet905Summ_Full = new TH1F("Hmet905Summ_Full","Number after selection with weight Ele+Muo
",50,100,600);
        TH1F *Hmet985Summ_Full = new TH1F("Hmet985Summ_Full","Number after selection with weight Ele+Muo
",50,100,600);
        TH1F *Hmet865Summ_Full = new TH1F("Hmet865Summ_Full","Number after selection with weight Ele+Muo
",50,100,600);
        TH1F *Hmet1025Summ_Full = new TH1F("Hmet1025Summ_Full","Number after selection with weight Ele+Muo
",50,100,600);
        TH1F *Hmet1065Summ_Full = new TH1F("Hmet1065Summ_Full","Number after selection with weight Ele+Muo
",50,100,600);
        TH1F *Hmet1225Summ_Full = new TH1F("Hmet1225Summ_Full","Number after selection with weight Ele+Muo
",50,100,600);
        TH1F *Hmet1425Summ_Full = new TH1F("Hmet1425Summ_Full","Number after selection with weight Ele+Muo
",50,100,600);
        TH1F *Hmet865_385Summ_Full = new TH1F("Hmet865_385Summ_Full","Number after selection with weight
Ele+Muo ",50,100,600);

        TH1F *Hmet905Summ_Before = new TH1F("Hmet905Summ_Before","Number after selection with weight
Ele+Muo ",50,100,600);
        TH1F *Hmet985Summ_Before = new TH1F("Hmet985Summ_Before","Number after selection with weight
Ele+Muo ",50,100,600);
        TH1F *Hmet865Summ_Before = new TH1F("Hmet865Summ_Before","Number after selection with weight
Ele+Muo ",50,100,600);
        TH1F *Hmet1065Summ_Before = new TH1F("Hmet1065Summ_Before","Number after selection with weight
Ele+Muo ",50,100,600);
        TH1F *Hmet1225Summ_Before = new TH1F("Hmet1225Summ_Before","Number after selection with weight
```

```
Ele+Muo ",50,100,600);
	TH1F *Hmet1425Summ_Before = new TH1F("Hmet1425Summ_Before","Number after selection with weight
Ele+Muo ",50,100,600);
	TH1F *Hmet1025Summ_Before = new TH1F("Hmet1025Summ_Before","Number after selection with weight
Ele+Muo ",50,100,600);
	TH1F *Hmet865_385Summ_Before = new TH1F("Hmet865_385Summ_Before","Number after selection with
weight Ele+Muo ",50,100,600);


	TFile *f1 = new TFile("Muo_output.root","READ");
	TFile *f3 = new TFile("Hist_SUMM_ele_muo.root","READ");
	TH1F *Hist_SUMM_Background = (TH1F *)f3->Get("Hmet30");
	Hmet865Summ_Full_WW->Add(Hmet865_Muo_Full_WW, Hmet865_Ele_Full_WW);
	Hmet905Summ_Full_WW->Add(Hmet905_Muo_Full_WW, Hmet905_Ele_Full_WW);
	Hmet985Summ_Full_WW->Add(Hmet985_Muo_Full_WW, Hmet985_Ele_Full_WW);
	Hmet865_385Summ_Full_WW->Add(Hmet865_385_Muo_Full_WW,Hmet865_385_Ele_Full_WW);
	Hmet1025Summ_Full_WW->Add(Hmet1025_Muo_Full_WW,Hmet1025_Ele_Full_WW);
	Hmet1065Summ_Full_WW->Add(Hmet1065_Muo_Full_WW,Hmet1065_Ele_Full_WW);
	Hmet1225Summ_Full_WW->Add(Hmet1225_Muo_Full_WW,Hmet1225_Ele_Full_WW);
	Hmet1425Summ_Full_WW->Add(Hmet1425_Muo_Full_WW,Hmet1425_Ele_Full_WW);
	Hmet865Summ_Before_WW->Add(Hmet865_Muo_Before_WW, Hmet865_Ele_Before_WW);
	Hmet905Summ_Before_WW->Add(Hmet905_Muo_Before_WW, Hmet905_Ele_Before_WW);
	Hmet985Summ_Before_WW->Add(Hmet985_Muo_Before_WW, Hmet985_Ele_Before_WW);
	Hmet865_385Summ_Before_WW->Add(Hmet865_385_Muo_Before_WW,Hmet865_385_Ele_Before_WW);
	Hmet1025Summ_Before_WW->Add(Hmet1025_Muo_Before_WW,Hmet1025_Ele_Before_WW);
	Hmet1065Summ_Before_WW->Add(Hmet1065_Muo_Before_WW,Hmet1065_Ele_Before_WW);
	Hmet1225Summ_Before_WW->Add(Hmet1225_Muo_Before_WW,Hmet1225_Ele_Before_WW);
	Hmet1425Summ_Before_WW->Add(Hmet1425_Muo_Before_WW,Hmet1425_Ele_Before_WW);
	Hmet865Summ_Full->Add(Hmet865_Muo_Full, Hmet865_Ele_Full);
	Hmet905Summ_Full->Add(Hmet905_Muo_Full, Hmet905_Ele_Full);
	Hmet985Summ_Full->Add(Hmet985_Muo_Full, Hmet985_Ele_Full);
	Hmet865_385Summ_Full->Add(Hmet865_385_Muo_Full,Hmet865_385_Ele_Full);
	Hmet1025Summ_Full->Add(Hmet1025_Muo_Full,Hmet1025_Ele_Full);
	Hmet1065Summ_Full->Add(Hmet1065_Muo_Full,Hmet1065_Ele_Full);
	Hmet1225Summ_Full->Add(Hmet1225_Muo_Full,Hmet1225_Ele_Full);
	Hmet1425Summ_Full->Add(Hmet1425_Muo_Full,Hmet1425_Ele_Full);
	Hmet865Summ_Before->Add(Hmet865_Muo_Before, Hmet865_Ele_Before);
	Hmet905Summ_Before->Add(Hmet905_Muo_Before, Hmet905_Ele_Before);
	Hmet985Summ_Before->Add(Hmet985_Muo_Before, Hmet985_Ele_Before);
	Hmet865_385Summ_Before->Add(Hmet865_385_Muo_Before,Hmet865_385_Ele_Before);
	Hmet1025Summ_Before->Add(Hmet1025_Muo_Before,Hmet1025_Ele_Before);
	Hmet1065Summ_Before->Add(Hmet1065_Muo_Before,Hmet1065_Ele_Before);
	Hmet1225Summ_Before->Add(Hmet1225_Muo_Before,Hmet1225_Ele_Before);
	Hmet1425Summ_Before->Add(Hmet1425_Muo_Before,Hmet1425_Ele_Before);

	const int n=50;
	Double_t Signif865[n+1];
	Double_t Summ[n+1];
	Double_t Signif1025[n+1];
	Double_t Signif1065[n+1];
	Double_t Signif1225[n+1];
	Double_t Signif1425[n+1];
	Double_t Signif865_385[n+1];
	Double_t Signif905[n+1];
	Double_t Signif985[n+1];
	Double_t Nbackgr[n];
	Double_t met[n+1];
	Double_t metGeV[n+1];
	met[0]=0;
	metGeV[0]=100;
	Double_t Test[n];



	for (int i =0; i<n; i++)
		{
			met[i+1]=1 +met[i];
			metGeV[i]=10*met[i]+metGeV[0];
			cout<<"met[i] "<<met[i]<<endl;
```

```
                        cout<<"metGeV[i]: "<<metGeV[i]<<endl;
                        Nbackgr[i]=sqrt(1.2*Hist_SUMM_Background->Integral(met[i], 50));
                        cout<<"Backgr "<<i<<" = "<<Nbackgr[i]<<endl;
                        cout<<endl;
                        //cout<<"sqrt1.2*Backgr "<<i<<" = "<<sqrt(1.2*Hist_SUMM_Background->Integral(met[i],
met[i+1]))<<endl;
                        Test[i]=Hmet905Summ_Full->Integral(met[i], 50);
                        if(Nbackgr[i]!=0)
                        {
                        Signif905[i]=Test[i]/Nbackgr[i];
                        Signif985[i] =Hmet985Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                        Signif905[i] =Hmet905Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                        Signif865[i] =Hmet865Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                        Signif1025[i] =Hmet1025Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                        Signif1225[i] =Hmet1225Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                        Signif1425[i] =Hmet1425Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                        Signif1065[i] =Hmet1065Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                        Signif865_385[i] =Hmet865_385Summ_Full->Integral(met[i], 50)/Nbackgr[i];
        Summ[i]=Signif865_385[i]+Signif1065[i]+Signif1425[i]+Signif1225[i]+Signif1025[i]+Signif865[i]+Signif905[i]+
Signif985[i];

                        }
                        else ()
                        {
                                cout<<"background=0"<<endl;
                                }
                        }
        TLegend* leg_1 = new TLegend(0.1,0.7,0.4,0.9);
        TCanvas *c1 = new TCanvas("c1","Significance",200,10,700,500);
        TMultiGraph *mg = new TMultiGraph();
        gr_Summ = new TGraph(n,metGeV,Summ);
        gr_Summ->SetLineColor(7);
        gr_Summ->SetLineWidth(5);
        gr_Summ->SetMarkerColor(7);
        gr_Summ->SetMarkerStyle(21);
        gr_11 = new TGraph(n,metGeV,Signif1065);
        gr_12 = new TGraph(n,metGeV,Signif1425);
        gr_13 = new TGraph(n,metGeV,Signif1225);
        gr8 = new TGraph(n,metGeV,Signif865_385);
        gr9 = new TGraph(n,metGeV,Signif985);
        gr6 = new TGraph(n,metGeV,Signif1025);

        gr2 = new TGraph(n,metGeV,Signif865);

        gr = new TGraph(n,metGeV,Signif905);

        gr_Summ->SetTitle("Significance");
        gr_Summ->GetXaxis()->SetTitle("Met");
        gr_Summ->GetYaxis()->SetTitle("Z");
        mg->Add(gr);
        mg->Add(gr2);
        mg->Add(gr6);
        mg->Add(gr8);
        mg->Add(gr9);
        mg->Add(gr_11);
        mg->Add(gr_12);
        mg->Add(gr_13);
        mg->Add(gr_Summ);
        mg->Draw("ALP");
        leg_1->Draw("same");
        c1->Update();
        c1->GetFrame()->SetFillColor(21);
        c1->GetFrame()->SetBorderSize(12);
        c1->Modified();
        c1.Print("Signif.pdf");

        TFile *fout = new TFile("Summ_muo_ele.root","RECREATE");
        TCanvas  Hist_SUMM31("Summ_muo_ele","PDF", 800,600);
        gr->Write();
```

```
Hist_SUMM31.Print("Summ_muo_ele.pdf");
fout->Close();

TLegend* leg_2 = new TLegend(0.1,0.7,0.4,0.9);
TCanvas *c2 = new TCanvas("c2","N",200,10,700,500);
c2.SetFillColor(42);
c2.SetGrid();

Hist_SUMM_Background->Scale(0.1);

Hist_SUMM_Background->Draw();
leg_2->AddEntry(Hist_SUMM_Background,"Background","l");

                    Signif985[i] =Hmet985Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                    Signif905[i] =Hmet905Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                    Signif865[i] =Hmet865Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                    Signif1025[i] =Hmet1025Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                    Signif1225[i] =Hmet1225Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                    Signif1425[i] =Hmet1425Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                    Signif1065[i] =Hmet1065Summ_Full->Integral(met[i], 50)/Nbackgr[i];
                    Signif865_385[i] =Hmet865_385Summ_Full->Integral(met[i], 50)/Nbackgr[i];

  leg_2->Draw("same");

  Double_t Maximum_Signif985=0, Maximum_Met985=0;
  Double_t Maximum_Signif905=0, Maximum_Met905=0;
  Double_t Maximum_Signif1025=0, Maximum_Met1025=0;
  Double_t Maximum_Signif1065=0, Maximum_Met1065=0;
  Double_t Maximum_Signif1225=0, Maximum_Met1225=0;
  Double_t Maximum_Signif1425=0, Maximum_Met1425=0;
  Double_t Maximum_Signif865_385=0, Maximum_Met865_385=0;
  Double_t Maximum_Signif865=0, Maximum_Met865=0;
  Double_t Maximum_SignifSumm=0, Maximum_MetSumm=0;


  for (int j=0; j<50; j++){if
(Signif985[j]>Maximum_Signif985){Maximum_Signif985=Signif985[j];Maximum_Met985=j;}}
  for (int j=0; j<50; j++){if
(Signif905[j]>Maximum_Signif905){Maximum_Signif905=Signif905[j];Maximum_Met905=j;}}
  for (int j=0; j<50; j++){if
(Signif865[j]>Maximum_Signif865){Maximum_Signif865=Signif865[j];Maximum_Met865=j;}}
  for (int j=0; j<50; j++){if
(Signif865_385[j]>Maximum_Signif865_385){Maximum_Signif865_385=Signif865_385[j];Maximum_Met865_385=j;}}
  for (int j=0; j<50; j++){if
(Signif1025[j]>Maximum_Signif1025){Maximum_Signif1025=Signif1025[j];Maximum_Met1025=j;}}
  for (int j=0; j<50; j++){if
(Signif1065[j]>Maximum_Signif1065){Maximum_Signif1065=Signif1065[j];Maximum_Met1065=j;}}
  for (int j=0; j<50; j++){if
(Signif1225[j]>Maximum_Signif1225){Maximum_Signif1225=Signif1225[j];Maximum_Met1225=j;}}
  for (int j=0; j<50; j++){if
(Signif1425[j]>Maximum_Signif1425){Maximum_Signif1425=Signif1425[j];Maximum_Met1425=j;}}
  for (int j=0; j<50; j++){if
(Summ[j]>Maximum_SignifSumm){Maximum_SignifSumm=Summ[j];Maximum_MetSumm=j;}}
  cout<<"Maximum_SignifSumm: "<<Maximum_SignifSumm<<endl;
  cout<<"Maximum_MetSumm: "<<100+Maximum_MetSumm*10<<endl;

        c2.Print("N_all.pdf");
        f1->Close();
        f2->Close();
        }
```

Приложение 4

Построение двумерной гистограммы статистической значимости с учетом переоптимизированного критерия по MET.

```
#include "TROOT.h"
#include "TFile.h"
#include "TTree.h"
```

```
#include "TBrowser.h"
#include "TH2.h"
#include "TRandom.h"
#include "TMath.h"
Int_t number =1000000;

Float_t calc, calc1;
Float_t err;

Float_t calc_Before, calc1_Before;
Float_t err_Before;

void D2_Met_Summ_Z() {

Ele();
Muo();
SUMM_HIST_Final();

}

void Muo()
{

TH2F *Hmet2 = new TH2F("Hmet2","Number without weight ",100,300,1500,100,0,900);
TH2F *Hmet3 = new TH2F("Hmet3","Error",100,300,1500,100,0,900);
TH2F *Hmet4 = new TH2F("Hmet4","Number without weight",100,300,1500,100,0,900);
TH2F *Hmet2_Before = new TH2F("Hmet2_Before","Number without weight ",100,300,1500,100,0,900);
TH2F *Hmet3_Before = new TH2F("Hmet3_Before","Error",100,300,1500,100,0,900);
TH2F *Hmet4_Before = new TH2F("Hmet4_Before","Number without weight",100,300,1500,100,0,900);
float Mgluino[npoints] = {425, 287, 345, 397, 465, 505, 477, 505, 545, 585, 625, 665, 557, 585, 625, 665, 705, 745, 785, 825, 865, 985, 905,
945, 1025, 1065, 637, 665, 705, 745, 785, 985, 825, 865, 905, 945, 1025, 1065, 1105, 1185, 1225, 717, 745, 825, 985, 865, 905, 945, 1025,
1065, 1105, 1145, 1185, 1225, 1265, 1305, 1345, 1385, 797, 825, 905, 985, 1025, 1105, 1065, 1145, 1185, 1225, 1265, 1305, 1345, 1385,
985, 1425, 1465, 1065, 1145, 1185, 1225, 1265, 1305, 1345, 1385, 1425, 1465, 1105, 1185, 1265, 1305, 1345, 1385, 1425, 1465, 1465};
float Mchargino[npoints] = {385, 275, 305, 385, 385, 385, 465, 465, 465, 465, 465, 465, 545, 545, 545, 545, 545, 545, 545, 545, 545, 545,
545, 545, 545, 545, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 705, 705, 705, 705, 705, 705, 705, 705, 705,
705, 705, 705, 705, 705, 705, 705, 705, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 865, 785, 785, 865, 865, 865,
865, 865, 865, 865, 865, 865, 865, 945, 945, 945, 945, 945, 945, 945, 945, 945, 1025};
float MLSP[npoints] = {345, 262, 265, 372, 305, 265, 452, 425, 385, 345, 305, 265, 532, 505, 465, 425, 385, 345, 305, 265, 225, 105, 185,
145, 65, 25, 612, 585, 545, 505, 465, 265, 425, 385, 345, 305, 225, 185, 145, 65, 25, 692, 665, 585, 425, 545, 505, 465, 385, 345,305, 265,
225, 185, 145, 105, 65, 25, 772, 745, 665, 585, 545, 465, 505, 425, 385, 345, 305, 265, 225, 185, 745, 145, 105, 665, 585, 545, 505, 465,
425, 385, 345, 305, 265, 785, 705, 625, 585, 545, 505, 465, 425, 585};

char treename[100];
TTree *tree;
TFile *f = new TFile("sigtree_HardMuo.root","READ");

for (int i; i<npoints; i++) {
        //cout<<Mgluino[i]<<"     "<<Mchargino[i]<<"      "<<MLSP[i]<<endl;

sprintf(treename,"SM_GG2WWZZ_%d_%d_%d_NoSys",Mgluino[i],Mchargino[i],MLSP[i]);
//cout <<treename<<endl;
        calc1=0;
        err=0;
  calc = 0;
        calc1_Before=0;
        err_Before=0;
  calc_Before = 0;
  TTree *t1 = (TTree*)f->Get(treename);
  Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
  Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
  Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;
  Bool_t isQEDFSR, isNoCrackElectron, EF_mu24_j65_a4tchad_EFxe40_tclcw;
  Int_t AnalysisType;

  Double_t random;
  Int_t ev;
  t1->SetBranchAddress("isQEDFSR",&isQEDFSR);
  t1->SetBranchAddress("AnalysisType",&AnalysisType);
  t1->SetBranchAddress("isNoCrackElectron",&isNoCrackElectron);
  t1->SetBranchAddress("EF_mu24_j65_a4tchad_EFxe40_tclcw",&EF_mu24_j65_a4tchad_EFxe40_tclcw);
  t1->SetBranchAddress("lep2Pt",&lep2Pt);
  t1->SetBranchAddress("lep1Eta",&lep1Eta);
  t1->SetBranchAddress("lep1Pt",&lep1Pt);
  t1->SetBranchAddress("met",&met);
  t1->SetBranchAddress("mt",&mt);
  t1->SetBranchAddress("genWeight",&genWeight);
  t1->SetBranchAddress("eventWeight",&eventWeight);
  t1->SetBranchAddress("leptonWeight",&leptonWeight);
```

```
   t1->SetBranchAddress("triggerWeight",&triggerWeight);
   t1->SetBranchAddress("pileupWeight",&pileupWeight);
   t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
    t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
                t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);
   Long64_t nentries = t1->GetEntries();
  // cout<<"NumberAll:  "<<nentries<<endl;
   for (Long64_t j=0;j<nentries;j++) {
    t1->GetEntry(j);
                                                calc1_Before+=1;

err_Before+=genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300*genWeight*eventWeight*leptonWeight*triggerW
eight*pileupWeight*20300;
                                                calc_Before+=1*genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300;
    // Fill_DATA_with_weight
    if (( EF_mu24_j65_a4tchad_EFxe40_tclcw )&&met>190&&mt>120&& AnalysisType==2 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                        {
                                                calc1+=1;

err+=genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300*genWeight*eventWeight*leptonWeight*triggerWeight*pil
eupWeight*20300;
                                                calc+=1*genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300;
                                        }
  }


  Hmet2->Fill(Mgluino[i],MLSP[i],calc1);
  Hmet4->Fill(Mgluino[i],MLSP[i],calc);
  Hmet2_Before->Fill(Mgluino[i],MLSP[i],calc1_Before);
  Hmet4_Before->Fill(Mgluino[i],MLSP[i],calc_Before);

}
}
    f->Close();
                }

void Ele()
{
TH2F *Hmet2 = new TH2F("Hmet2","Number without weight ",100,300,1500,100,0,900);
TH2F *Hmet4 = new TH2F("Hmet4","Number with weight ",100,300,1500,100,0,900);
TH2F *Hmet3 = new TH2F("Hmet3","Delta N / N  ",100,300,1500,100,0,900);
TH2F *Hmet2_Before = new TH2F("Hmet2_Before","Number without weight ",100,300,1500,100,0,900);
TH2F *Hmet3_Before = new TH2F("Hmet3_Before","Error",100,300,1500,100,0,900);
TH2F *Hmet4_Before = new TH2F("Hmet4_Before","Number without weight",100,300,1500,100,0,900);

float Mgluino[npoints] = {425, 287, 345, 397, 465, 505, 477, 505, 545, 585, 625, 665, 557, 585, 625, 665, 705, 745, 785, 825, 865, 985, 905,
945, 1025, 1065, 637, 665, 705, 745, 785, 985, 825, 865, 905, 945, 1025, 1065, 1105, 1185, 1225, 717, 745, 825, 985, 865, 905, 945, 1025,
1065, 1105, 1145, 1185, 1225, 1265, 1305, 1345, 1385, 797, 825, 905, 985, 1025, 1105, 1065, 1145, 1185, 1225, 1265, 1305, 1345, 1385,
985, 1425, 1465, 1065, 1145, 1185, 1225, 1265, 1305, 1345, 1385, 1425, 1465, 1105, 1185, 1265, 1305, 1345, 1385, 1425, 1465, 1465};
float Mchargino[npoints] = {385, 275, 305, 385, 385, 385, 465, 465, 465, 465, 465, 465, 545, 545, 545, 545, 545, 545, 545, 545, 545, 545,
545, 545, 545, 545, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 625, 705, 705, 705, 705, 705, 705, 705, 705, 705,
705, 705, 705, 705, 705, 705, 705, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 785, 865, 785, 785, 865, 865, 865,
865, 865, 865, 865, 865, 865, 865, 945, 945, 945, 945, 945, 945, 945, 945, 945, 1025};
float MLSP[npoints] = {345, 262, 265, 372, 305, 265, 452, 425, 385, 345, 305, 265, 532, 505, 465, 425, 385, 345, 305, 265, 225, 105, 185,
145, 65, 25, 612, 585, 545, 505, 465, 265, 425, 385, 345, 305, 225, 185, 145, 65, 25, 692, 665, 585, 425, 545, 505, 465, 385, 345,305, 265,
225, 185, 145, 105, 65, 25, 772, 745, 665, 585, 545, 465, 505, 425, 385, 345, 305, 265, 225, 185, 745, 145, 105, 665, 585, 545, 505, 465,
425, 385, 345, 305, 265, 785, 705, 625, 585, 545, 505, 465, 425, 585};

char treename[100];
TTree *tree;
TFile *f = new TFile("sigtree_HardEle.root","READ");

for (int i; i<npoints; i++) {
                //cout<<Mgluino[i]<<"     "<<Mchargino[i]<<"      "<<MLSP[i]<<endl;

sprintf(treename,"SM_GG2WWZZ_%d_%d_%d_NoSys",Mgluino[i],Mchargino[i],MLSP[i]);
//cout <<treename<<endl;
                calc1=0;
                err=0;
    calc = 0;
    calc1_Before=0;
                err_Before=0;
   calc_Before = 0;
   TTree *t1 = (TTree*)f->Get(treename);
   Float_t lep2Pt, lep1Pt, lep1Eta, met, mt;
```

```
  Float_t genWeight, eventWeight, leptonWeight, triggerWeight, pileupWeight;
  Float_t jet1Pt_JVF25pt50, jet7Pt_JVF25pt50, meffInc25_JVF25pt50;
  Bool_t isQEDFSR, isNoCrackElectron, EF_e60_medium1, EF_e24vh_medium1_EFxe35_tclcw;
  Int_t AnalysisType;

Double_t random;
  Int_t ev;
  t1->SetBranchAddress("isQEDFSR",&isQEDFSR);
  t1->SetBranchAddress("AnalysisType",&AnalysisType);
  t1->SetBranchAddress("isNoCrackElectron",&isNoCrackElectron);
  t1->SetBranchAddress("EF_e60_medium1",&EF_e60_medium1);
  t1->SetBranchAddress("EF_e24vh_medium1_EFxe35_tclcw",&EF_e24vh_medium1_EFxe35_tclcw);

  t1->SetBranchAddress("lep2Pt",&lep2Pt);
  t1->SetBranchAddress("lep1Eta",&lep1Eta);
  t1->SetBranchAddress("lep1Pt",&lep1Pt);
  t1->SetBranchAddress("met",&met);
  t1->SetBranchAddress("mt",&mt);
  t1->SetBranchAddress("genWeight",&genWeight);
  t1->SetBranchAddress("eventWeight",&eventWeight);
  t1->SetBranchAddress("leptonWeight",&leptonWeight);
  t1->SetBranchAddress("triggerWeight",&triggerWeight);
  t1->SetBranchAddress("pileupWeight",&pileupWeight);
  t1->SetBranchAddress("meffInc25_JVF25pt50",&meffInc25_JVF25pt50);
   t1->SetBranchAddress("jet1Pt_JVF25pt50",&jet1Pt_JVF25pt50);
            t1->SetBranchAddress("jet7Pt_JVF25pt50",&jet7Pt_JVF25pt50);


  Long64_t nentries = t1->GetEntries();
 // cout<<"NumberAll:  "<<nentries<<endl;
  for (Long64_t j=0;j<nentries;j++) {
    t1->GetEntry(j);
                                        calc1_Before+=1;

err_Before+=genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300*genWeight*eventWeight*leptonWeight*triggerW
eight*pileupWeight*20300;
                                        calc_Before+=1*genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300;
   // Fill_DATA_with_weight
   if (( EF_e24vh_medium1_EFxe35_tclcw || EF_e60_medium1  )&&met>190&&mt>120&& AnalysisType==1 &&
isNoCrackElectron==1&&lep1Pt>25&&lep2Pt<10&&jet7Pt_JVF25pt50>25&&jet1Pt_JVF25pt50>80&&meffInc25_JVF25pt50>750&&
(abs(lep1Eta)<1.37 || 1.52<abs(lep1Eta)))

                                {
                                        calc1+=1;

err+=genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300*genWeight*eventWeight*leptonWeight*triggerWeight*pil
eupWeight*20300;
                                        calc+=1*genWeight*eventWeight*leptonWeight*triggerWeight*pileupWeight*20300;
                                }
 }

 if (calc1!=0&&calc1!=1)
 {

 Hmet2->Fill(Mgluino[i],MLSP[i],calc1);
 Hmet4->Fill(Mgluino[i],MLSP[i],calc);
 Hmet2_Before->Fill(Mgluino[i],MLSP[i],calc1_Before);
 Hmet4_Before->Fill(Mgluino[i],MLSP[i],calc_Before);
}
}


   f->Close();

            }

            void SUMM_HIST_Final()
{
            TH2F *Hmet_Eff_WW= new TH2F("Hmet31","Z without weight Ele+Muo ",100,300,1500,100,0,900);
            TH2F *Hmet_Eff= new TH2F("Hmet31","Z with weight Ele+Muo ",100,300,1500,100,0,900);
            TH2F *Hmet33 = new TH2F("Hmet33","Number with weight Ele+Muo ",100,300,1500,100,0,900);
            TH2F *Hmet31 = new TH2F("Hmet31","Number without weight Ele+Muo ",100,300,1500,100,0,900);
            TH2F *Hmet32 = new TH2F("Hmet32","Error ",100,300,1500,100,0,900);
            TH2F *Hmet31_Before = new TH2F("Hmet31","Number without weight Ele+Muo ",100,300,1500,100,0,900);
            TH2F *Hmet33_Before = new TH2F("Hmet33","Number with weight Ele+Muo ",100,300,1500,100,0,900);

            TFile *f1 = new TFile("2D_HIST_Ele.root","READ");
            TH2F *Hmet11 = (TH2F *)f1->Get("Hmet2");
```

```
	TH2F *Hmet12 = (TH2F *)f1->Get("Hmet3");
	TH2F *Hmet13 = (TH2F *)f1->Get("Hmet4");
	TH2F *Hmet11_Before = (TH2F *)f1->Get("Hmet2_Before");
	TH2F *Hmet13_Before = (TH2F *)f1->Get("Hmet4_Before");

	TFile *f2 = new TFile("2D_HIST_Muo.root","READ");
	TH2F *Hmet21 = (TH2F *)f2->Get("Hmet2");
	TH2F *Hmet22 = (TH2F *)f2->Get("Hmet3");
	TH2F *Hmet23 = (TH2F *)f2->Get("Hmet4");
	TH2F *Hmet21_Before = (TH2F *)f2->Get("Hmet2_Before");
	TH2F *Hmet23_Before = (TH2F *)f2->Get("Hmet4_Before");
	Hmet31->Add(Hmet11,Hmet21);
	Hmet32->Add(Hmet12,Hmet22);
	Hmet33->Add(Hmet13,Hmet23);
	Hmet31_Before->Add(Hmet11_Before,Hmet21_Before);
	Hmet33_Before->Add(Hmet13_Before,Hmet23_Before);
		Hmet_Eff_WW->Divide(Hmet31,Hmet31_Before);
		Hmet_Eff->Divide(Hmet33,Hmet33_Before);

	TFile *fout = new TFile("2D_Met_Summ_Z.root","RECREATE");

	TCanvas  Hist_SUMM31("Efficiency without weight Ele+Muo_2D","PDF", 800,600);

	Hmet_Eff_WW->GetXaxis()->SetTitle("Mgluino");
Hmet_Eff_WW->GetYaxis()->SetTitle("MLSP");
Hmet_Eff_WW->Draw("TEXT");
	Hmet_Eff_WW->Write();

	TCanvas  Hist_SUMM32("Error_2D","PDF", 800,600);

	Hmet32->GetXaxis()->SetTitle("Mgluino");
Hmet32->GetYaxis()->SetTitle("MLSP");
Hmet32->Draw("TEXT");
	Hmet32->Write();

	TCanvas  Hist_SUMM33("Z_Ele+Muo_2D","PDF", 800,600);

		Hmet_Eff->GetXaxis()->SetTitle("Mgluino");
	Hmet_Eff->GetYaxis()->SetTitle("MLSP");
	Hmet_Eff->Draw("TEXT");
		Hmet_Eff->Write();

	Hist_SUMM32.Print("Error_2D.pdf");
	Hist_SUMM33.Print("Z_Ele+Muo.pdf");
	fout->Close();
	f1->Close();
	f2->Close();
	}
```