МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
(НИЯУ МИФИ)

ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ И ТЕХНОЛОГИЙ
КАФЕДРА №40 «ФИЗИКА ЭЛЕМЕНТАРНЫХ ЧАСТИЦ»

УДК 53.05, 53.07

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

ОЦЕНКА ЭФФЕКТИВНОСТИ МОДЕЛИРОВАНИЯ ФИЗИЧЕСКИХ СИГНАЛОВ С ПОМОЩЬЮ МОДИФИЦИРОВАННОГО МЕТОДА МОРФИНГА

Студент _____ Г. И. Воробьев

Научный руководитель,
к.ф.-м.н. _____ Н. Л. Беляев

Москва 2022

Содержание

Введение

Бозон Хиггса — частица теоретически предсказанная Питером Хиггсом и пятью другими учёными в 1964 году. Экспериментально с достаточной точностью она была обнаружена в 2012 году в экспериментах ATLAS и CMS на Большом адронном коллайдере (LHC) в ЦЕРНе. Эта частица особенна тем, что связанное с ней поле, добавляемое механизмом Хиггса, вызывает спонтанное нарушение симметрии из-за чего у некоторых частиц стандартной модели появляется масса. Тем самым эта частица и связанная с ней теория завершают стандартную модель (СМ), делая её полностью самостоятельной теорией, описывающей электромагнитное, слабое и сильное взаимодействия. На данный момент все её предсказания с высокой точностью подтверждаются экспериментально и успешно объясняют большую часть явлений в физике частиц.

Однако множество вопросов, в том числе описание гравитационного взаимодействия, иерархия частиц, осцилляции нейтрино, тёмные энергия и материя — остаются без ответа. Как следствие, главной задачей современной науки стал поиск «*новой физики*» — пределов применимости СМ. В то же время разрабатываются теории, основанные на других принципах или дополняющие, поправляющие СМ.

Целью данной работы является: Ознакомление с взаимодействиями в рамка и во вне Стандартной модели элементарных частиц в хиггсовском секторе; Ознакомление с методом «*морфинга*»; Усовершенствование метода для более широкого ряда условий использования.

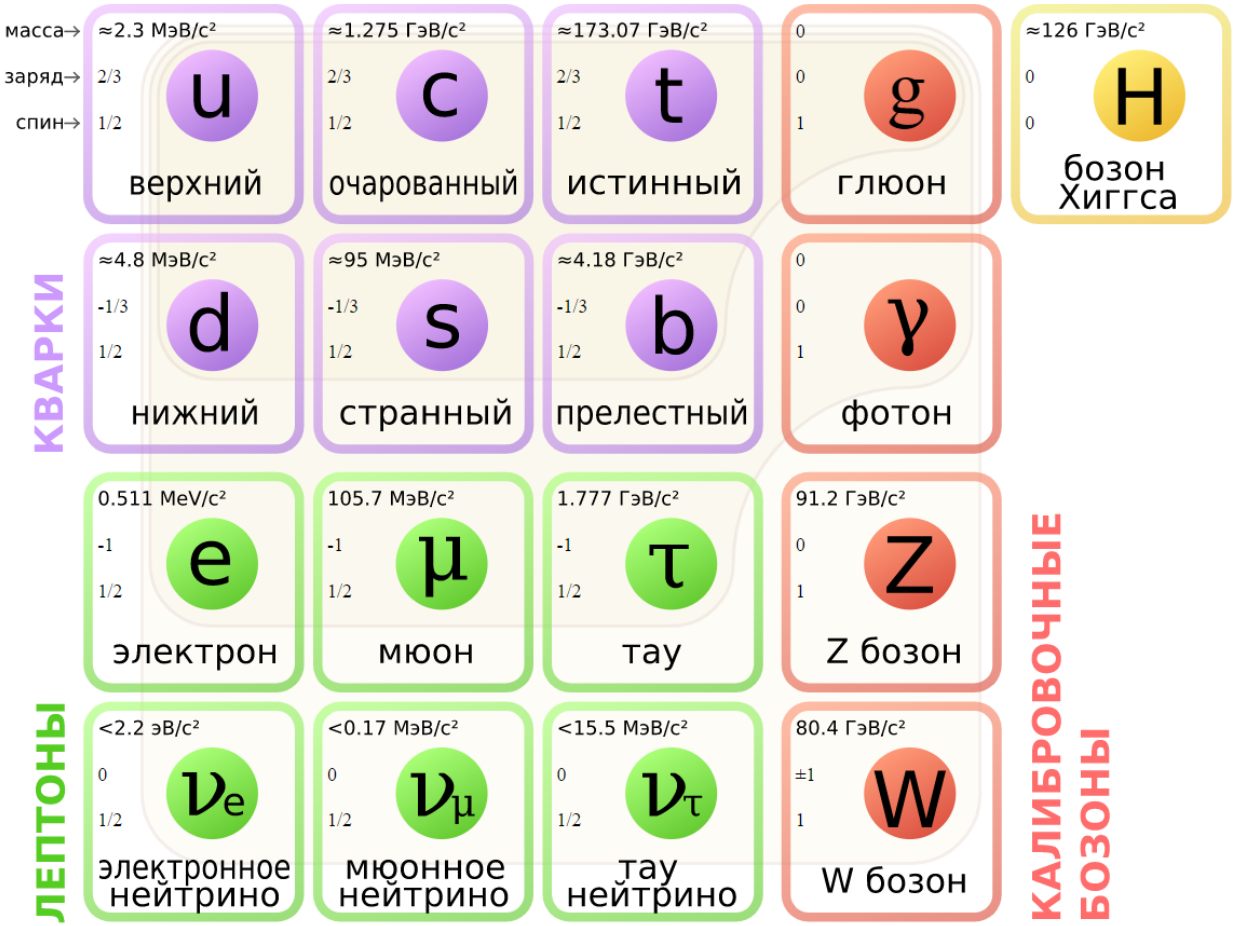

Рисунок 1 — 6 кварков, 6 лептонов и 5 бозонов Стандартной модели

1 Бозон Хиггса

1.1 Стандартная модель

Как было отмечено, стандартная модель (СМ) получила логическое завершение с открытием последней предсказываемой ею частицы - *бозон Хиггса*. Предпосылки к существованию этой частицы появились в связи с требованиями калибровочная симметрии, а точнее экспериментальным обнаружением её нарушения.

Принцип калибровочной симметрии запрещает присутствие массового члена для калибровочных бозонов в лагранжиане, описывающем движение свободного дираковского фермиона. Фермионные массовые члены также отсутствуют, поскольку они будут смешивать левые и правые поля, которые имеют различные свойства преобразования, и, следовательно, могут привести к явному нарушению калибровочной симметрии. Таким об-

разом, лагранжиан теории на основе группы симметрии $SU(2)_L \otimes U(1)_Y$ содержит только безмассовые поля [1]. Однако экспериментально массы калибровочных бозонов $W^{\pm}$ и Z^0 обнаружены с отличными от нуля значениями [2].

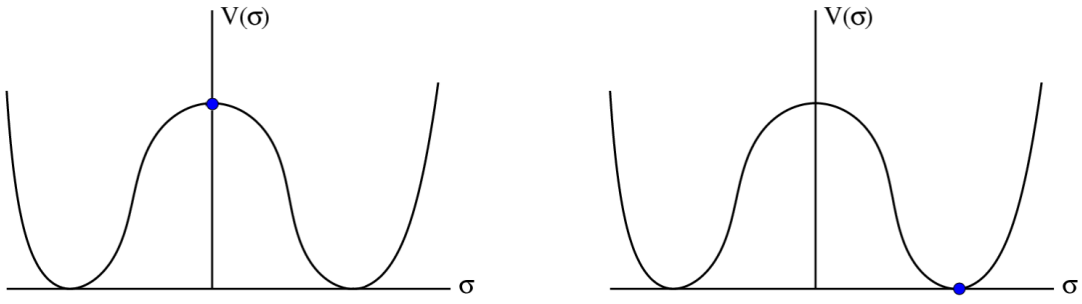

Рисунок 1.1 – Слева - состояние физической системы до спонтанного нарушения симметрии. Справа - после.

Эта дилемма была решена с помощью принципа *спонтанного нарушения симметрии*. Графики на рис. 1.1 иллюстрируют этот принцип. На левом графике изображено симметричное состояние лагранжиана, а на правом симметрия спонтанно нарушена. В квантовой теории поля существование вырожденных состояний (или соединяющих их эквипотенциальных поверхностей) с минимальной энергией подразумевает существование безмассовых степеней свободы. Таким образом механизм спонтанного нарушения электрослабой симметрии, который также называют *механизмом Хиггса*, позволяет элегантно решить проблему масс калибровочных бозонов.

1.2 Новая физика

Поскольку с Хиггс бозоном стандартная модель обрела цельность и была проверена во многих экспериментах – текущей задачей перед научным сообществом стали установление границ применимости СМ и поиск новой физики, которая либо даст толчок к развитию теорий расширения СМ или новых принципиально теорий. Далее мы рассмотрим одно из направлений этих поисков.

1.2.1 Эффективная теория поля

Эффективная теория поля (ЭТП) - это теория утверждающая возможность расширения лагранжиана СМ новыми операторами с каноническими размерностями D больше 4 при энергиях ниже массового масштаба новых частиц Λ([3]). В общем случае лагранжиан ЭТП может быть представлен в следующем виде:

$$\mathcal{L}_{EFT} = \mathcal{L}_{SM} + \sum_i \frac{c_i^{(5)}}{\Lambda}\mathcal{O}_i^{(5)} + \sum_i \frac{c_i^{(6)}}{\Lambda^2}\mathcal{O}_i^{(6)} + \sum_i \frac{c_i^{(7)}}{\Lambda^3}\mathcal{O}_i^{(7)} + \dots \qquad (1)$$

где каждый член $\mathcal{O}_i^{(D)}$ разложения представляет собой $SU(3)_C \otimes SU(2)_L \otimes U(1)_Y$ -инвариантный оператор размерности D, а параметры $c_i^{(D)}$, являющиеся множителями операторов в лагранжиане, называются коэффициентами Вильсона.

Данную модель можно использовать ограничившись разным набором операторов и в разных базисах. Исследование о их эквивалентности было приведено в работе [1]. В частности наша исследовательская группа в НИЯУ МИФИ преимущественно использует ЭТП с операторами размерности 6 в Хиггс базисе.

2 Морфинг

Для исследования тензорной структуры констант связи бозона Хиггса с калибровочными бозонами необходимо моделировать сигналы, как и для любого другого анализа. Однако в рамках этого исследования возникает много проблем со скоростью вычислений из-за огромного количества переменных, которые в том числе могут коррелировать между друг другом, при моделировании самих сигналов, дифференциальных распределений и других параметров модели. С целью разрешить возникающие трудности был разработан метод *морфинга* (англ. morphing).

2.1 Обзор метода

Модель сигнала, построенная с помощью метода морфинга, представляет собой линейную комбинацию **минимального набора** ортогональных базовых сигналов (элементов), охватывающих всё рассматриваемое пространство констант связи. Вес каждого элемента определяется на основе значений констант связи, присутствующих в специальной сигнальной матрице.

Морфинг - это **не** просто **метод интерполяции**, поскольку он не ограничен точками в диапазоне, охватываемом базовыми элементами. Фактически, выбор базовых элементов является произвольным, и любой набор базовых элементов, удовлетворяющий необходимым условиям для построения функции преобразования, будет охватывать всё пространство независимо от их местоположения в пространстве констант связи [4].

$$T_{\text{out}}(\vec{g}_{\text{target}}) = \sum_i w_i(\vec{g}_{\text{target}}; \vec{g}_i) T_{\text{in}}(\vec{g}_i), \tag{2}$$

Приведём схему построения функции морфинга для процессов с произвольным числом свободных констант связи в двух вершинах:

1. Построить квадрат общего матричного элемента (левый множитель отвечает за вершины рождения (p), правый, соответственно, – распад

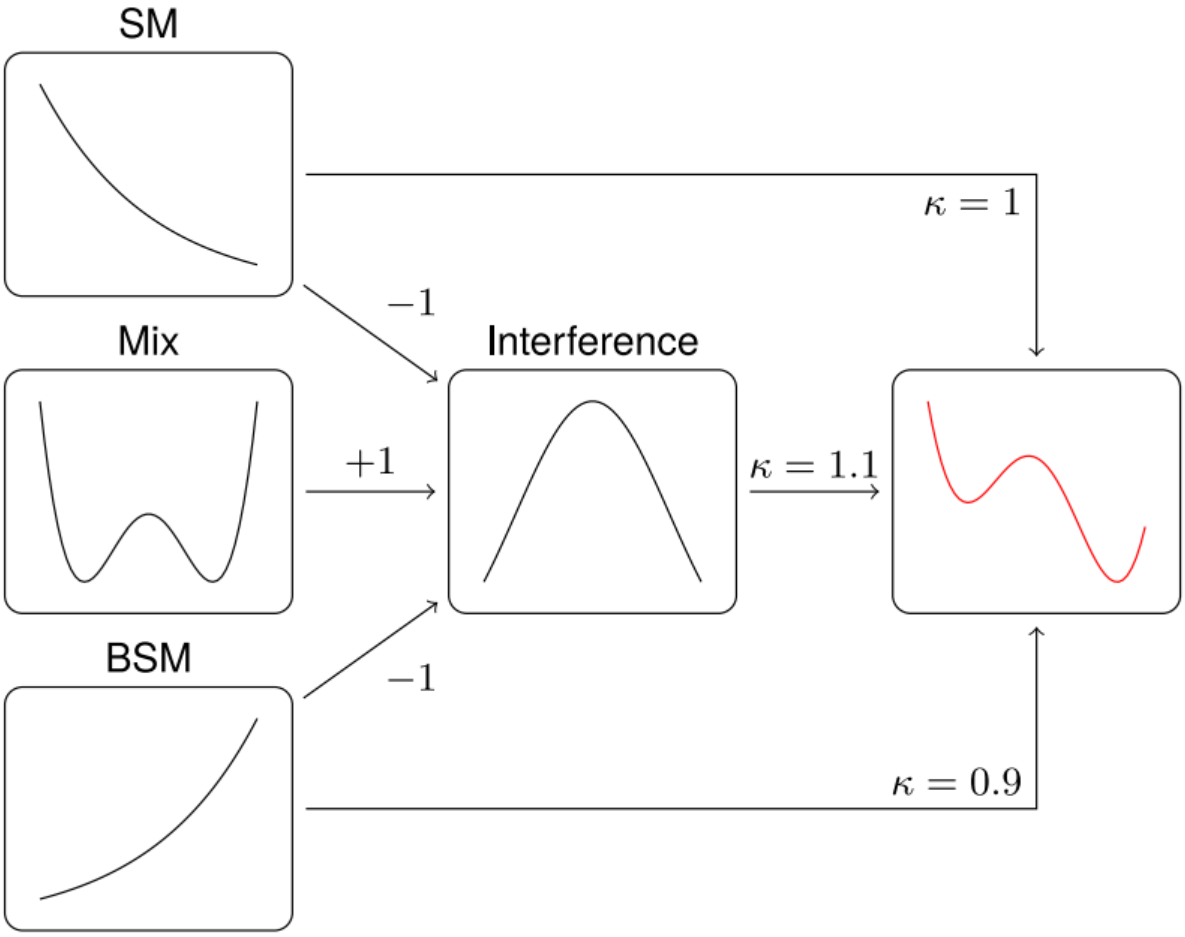

Рисунок 2.1 – Иллюстрация процедуры морфинга в простом примере

(d). Общие вершины отмечены индексом (s):

$$|\mathrm{M}E(\vec{g})|^2 = \Big(\sum_{x\in p,s} g_x \mathcal{O}(g_x)\Big)^2 \cdot \Big(\sum_{x\in d,s} g_x \mathcal{O}(g_x)\Big)^2, \tag{3}$$

Предполагая, что вершины рождения и распада некоррелированы, что имеет место для скалярной виртуальной частицы, переходим к следующему шагу

2. Разложить квадрат матричного элемента до полинома n-й степени (в данном случае 4) в терминах констант связи:

$$|\mathrm{M}E(\vec{g})|^2 = \sum_{i=1}^{N} X_i \cdot P_i(\vec{g}), \tag{4}$$

Здесь X_i является фактором, который будет представлен входным

распределением. В полиноме n-й степени $P_i(\vec{g}) = g_{a_1} g_{a_2} ... g_{a_n}$ по константам связи $\vec{g}$, одни и те же константы связи могут появляться несколько раз. Количество различных выражений в полученном многочлене равно числу базовых элементов N, необходимых для осуществления процедуры морфинга.

3. Генерация входных распределений $T_{\text{in},i}$ в произвольных, но фиксированных точках пространства констант связи $\vec{g}_i$:

$$T_{\text{in},i} \propto |\text{M}E(\vec{g}_i)|^2, \tag{5}$$

4. После чего построить функцию морфинга вида:

$$T_{\text{out},i}(\vec{g}) = \sum_{i=1}^{N} \Big(\sum_{j=1}^{N} A_{ij} P_j(\vec{g}) \Big) T_{\text{in},i} = \vec{P}(\vec{g}) \cdot A\vec{T} \tag{6}$$

где второе равнество аналогично первому, но преобразовано для матричной записи. Матрица A должна быть рассчитана для получения полной функции морфинга.

Также нужно нормализовать входное и выходное распределения и добавить условия на единственность решения. Более подробно алгоритм описан в статье [4].

Суммируя, метод морфинга сильно упрощает вычислительную задачу моделирования с интегральных методов Монте-Карло, до достаточно тривиальных операций над базисными гистограммами.

Однако отсюда можно заметить и недостаток этого метода. Проблему представляет то, что как внутри, так в точке далеко от той области, где расположены базовые элементы, достоверность предсказания (в некоторых случаях внутри области и во всех случаях в достаточно далёких областях) может значительно ухудшаться. Причина состоит в том, что конечный результат морфинга складывается из суммы нескольких гистограмм, каждая из которых имеет свой вес. При этом в некоторых точках веса могут достигать больших значений по абсолютной величине, что, зачастую, приводит к уменьшению эффективной статистики и увеличению погрешностей. В дальних областях, так же играет роль информация об их характеристиках.

Но стоит заметить, что случаи с отдалёнными областями не представляет большой проблемы. В целях решить эту проблему была разработана улучшенная концепция данного алгоритма, которая получила название метода *расширенного морфинга*. Суть его состоит в использовании дополнительных базовых элементов, которые задействуются одновременно с основными элементами.

2.2 Переопределённый морфинг

Рассмотрим формулы 2 и 5. Их можно записать в более наглядном виде. В частности это упрощение видно и на рис. 2.1. Распишем эти формулы:

$$T_{\text{in},i}(g_{\text{SM},i}, g_{B\text{SM},i}) \propto$$
$$\propto g^2_{B\text{SM},i}|\mathcal{O}_{B\text{SM}}|^2 + g^2_{\text{SM},i}|\mathcal{O}_{\text{SM}}|^2 + 2g_{\text{SM},i}g_{\text{SM},i}\mathcal{R}(\mathcal{O}^*_{\text{SM}}\mathcal{O}_{\text{SM}}), i = 1, \dots 3. \quad (7)$$

$$\begin{aligned} T_{\text{out}}(g_{\text{SM}}, g_{B\text{SM}}) = (a_{11}g^2_{\text{SM}} + a_{12}g^2_{B\text{SM}} + a_{13}g_{B\text{SM}}g_{\text{SM}})T_{\text{in},1}(g_{\text{SM},1}, g_{B\text{SM},1}) \\ +(a_{21}g^2_{\text{SM}} + a_{22}g^2_{B\text{SM}} + a_{23}g_{B\text{SM}}g_{\text{SM}})T_{\text{in},2}(g_{\text{SM},2}, g_{B\text{SM},2}) \\ +(a_{31}g^2_{\text{SM}} + a_{32}g^2_{B\text{SM}} + a_{33}g_{B\text{SM}}g_{\text{SM}})T_{\text{in},3}(g_{\text{SM},3}, g_{B\text{SM},3}) \quad (8) \end{aligned}$$

Или ещё упрощая можно записать:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} g^2_{\text{SM},1} & g^2_{\text{SM},2} & g^2_{\text{SM},3} \\ g^2_{B\text{SM},1} & g^2_{B\text{SM},2} & g^2_{B\text{SM},3} \\ g_{\text{SM},1}g_{B\text{SM},1} & g_{\text{SM},2}g_{B\text{SM},2} & g_{\text{SM},3}g_{B\text{SM},3} \end{pmatrix} = \mathbf{I}$$
$$\Leftrightarrow A \cdot G = \mathbf{I} \quad (9)$$

Именно такая формулировка используется в текущей версии морфинга для случая **ggF** $np = 1, nd = 2, Nbase = Nmin = 3$. Однако если взять случай отличный, например **ggF** $np = 1, nd = 2, Nbase = 4, Nmin = 3$, то

выражение 9 примет вид:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{pmatrix} \cdot \begin{pmatrix} g^2_{\text{SM},1} & g^2_{\text{SM},2} & g^2_{\text{SM},3} & g^2_{\text{SM},4} \\ g^2_{B\text{SM},1} & g^2_{B\text{SM},2} & g^2_{B\text{SM},3} & g^2_{B\text{SM},4} \\ g_{\text{SM},1}g_{B\text{SM},1} & g_{\text{SM},2}g_{B\text{SM},2} & g_{\text{SM},3}g_{B\text{SM},3} & g_{\text{SM},4}g_{B\text{SM},4} \end{pmatrix} =$$

$$= \mathbf{I} \Leftrightarrow A \cdot G = \mathbf{I} \quad (10)$$

А т.к. одной из задач морфинга является нахождение параметров матрицы A, то используемый на данный момент метод поиска обратной матрицы не подходит, т.к. G^{-1} не существует.

Эту задачу в нашей группе именовали *переопределённым морфингом* (англ. Overdetermined Morphing). Решение подобной проблемы достаточно тривиально. В имеющейся программе необходимо заменить метод нахождения матрицы A. Новое решение можно вывести воспользовавшись методом наименьших квадратов:

Рассмотрим задачу $A\vec{x} = \vec{b}$, где $A = [m \times n], m >= n$. Обозначим $E(\vec{x}) = ||\vec{b} - A\vec{x}||^2, ||x_i||^2 = \sum\limits_i |x_i|^2$. Тогда:

$$E(\vec{x}) = (\vec{b} - A\vec{x})^T(\vec{b} - A\vec{x}) = (\vec{b})^T\vec{b} - 2(\vec{b})^T A\vec{x} + (\vec{x})^T A^T A\vec{x}$$

Методом наименьших квадратов:

$$\frac{\partial E(\vec{x})}{\partial \vec{x}} = -2A^T\vec{b} - 2A^T A\vec{x} = 0 \Rightarrow A^T A\vec{x} = A^T\vec{b} \Rightarrow \vec{x} = (A^T A)^{-1}A^T\vec{b}, \quad (11)$$

где $(A^T A)^{-1}A^T$ псевдо-инверсия A.

Гипотеза нашей группы состоит в следующем: поскольку появляется возможность добавлять семплы помимо базисного колличества, то достоверность предсказаний модели после внедрения этого метода улучшится по сравнению оригинальной версией. Внедрению этой новой функции и посвящена данная работа.

3 Инструменты ATLAS

Для внесения изменений необходимо сначала ознакомится с концептом программной реализации. Во-первых, за интересующую нас часть кода отвечает отдельная программа [5]. С оригинальной можно ознакомиться в источнике [6]. Во-вторых структура этих программ модульная. В ATLAS был разработан софт ATLAS Analysis Release, помогающий нам настроить среду и наше программное обеспечение для корректного и простого использования (например, версии, рекомендованные группами CP). Этот софт устанавливает подходящие компиляторы как стандартные, находит и устанавливает (с помощью сборки CMake проекта) библиотеки, используемые в проекте.

Практическая часть работы началась с изучения программного обеспечения ATLAS Analysis Release. Для работы с модулями такого типа обычно делят рабочую дерикторию на 3-и папки: *source*, *run* и *build*. В папку *source* клонируются репозитории с необходимыми для работы модулями. В папке *build* будет производиться установка модуля и окружения. А в папке *run* можно запускать функции модуля, для более удобной работы с файлами вывода.

Алгоритм настройки среды:

1. После клонирования модулей в *source* - в этой же папке создаётся специальный CMakeLists.txt файл, описанный в источнике [7].

2. Далее вводим команды инициализирующие настройку среды с помощью инструкций описанных в CMakeLists.txt файле.

```
1    cd build/
2    asetup AnalysisBase 21.2.189
3    cmake ../source/
4    make
```

После этого мы можем использовать функции модуля, а в случае внесения изменений перекомпилировать их и запускать заново:

```
1    make && func_name
```

4 Программа оптимизации базиса Морфинга

4.1 Модуль «MorphingStudies»

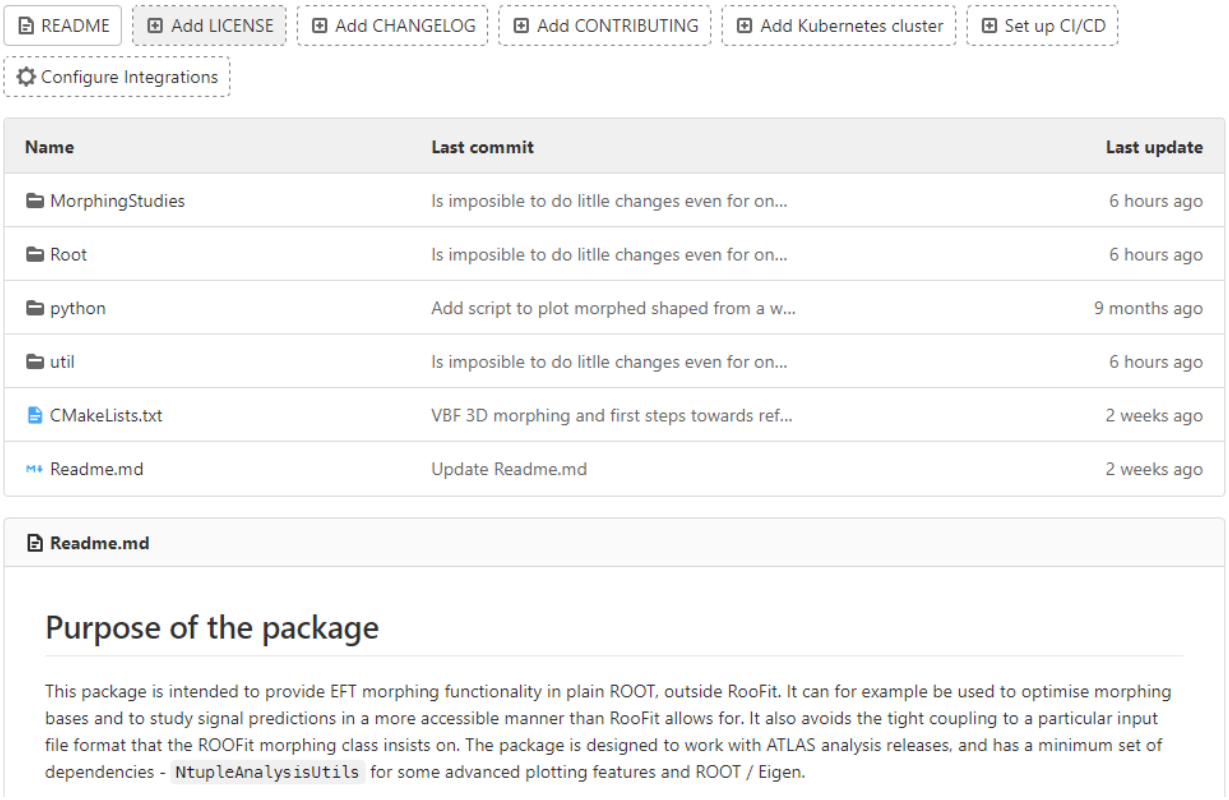

Рисунок 4.1 – Файловая структура программы

Опишем модуль и некоторые его классы и методы. Файловая структура (рис. 4.1) соответствует стандартному модульному проекту ATLASAnalysisBase. В папке «*MorphingStudies*» содержатся файлы с разрешением *.header* и *.ixx*. Здесь объявляются все функции, классы и их методы модуля. В папке «*Root*» хранятся некоторые вспомогательные скрипты, которые либо расшифровывают часть методов объявленных в *MorphingStudies*, либо настраивают дополнительные шаблоны работы программы. Папка «textitPython» предназначена для версии первой модуля на языке Python. Это независимая папка (находящаяся в ней скрипты никак не связаны с остальным модулем) и её скрипты приложены исключительно в ознакомительных целях. И последняя папка «*util*» используется, как рабочая область для хранения и разработки скриптов.

Принцип работы с модулем: Используя классы, их методы и шаблоны пишется скрипт для конкретной задачи с использованием внешних

анализируемых данных.

В основе проекта лежит класс *MorphingCalculator* в нём задаётся алгебра морфинга - методы для создания матрицы морфинга и различных параметров использующих для своего расчёта операции над этой матрицей. Основной внешний параметр - это матрица типа `std::array<std::array<float,nCoefficients>,nBasisPoints>`, где *nCoefficients* - это сумма шаблонных параметров *nProdOnly, nDecayOnly, nProdAndDecay* отвечающих за число вершин рождения, распада и двойных (одновременно распад и рождение) соответственно, а *nBasisPoints* - это комбинация тех же параметров, рассчитывающая необходимое (минимальное или базисное) количество сэмплов для данной конфигурации вершин по формуле (27-30) доступной в источнике [4].

Далее по важности идут классы *MorphingBasis*, наследующий *MorphingCalculator* и вспомогательный *BasisOptimization*. Первый упрощает использование класса алгебры морфинга и добавляет более общие методы для работы с результатами вычислений как со стороны пользователя, так и со стороны других классов. Второй же выполняет задачу оптимизации полученного базиса. Доступны разные методы оптимизации базиса для получения более однородных по числу вхождений данных. Используя стандартные методы можно наблюдать следующие распределения для коэффициентов $cH\tilde{B}$ и $cH\tilde{W}$ d случае 2 мерного VBF (две пары рождения и распада) рис.4.2.

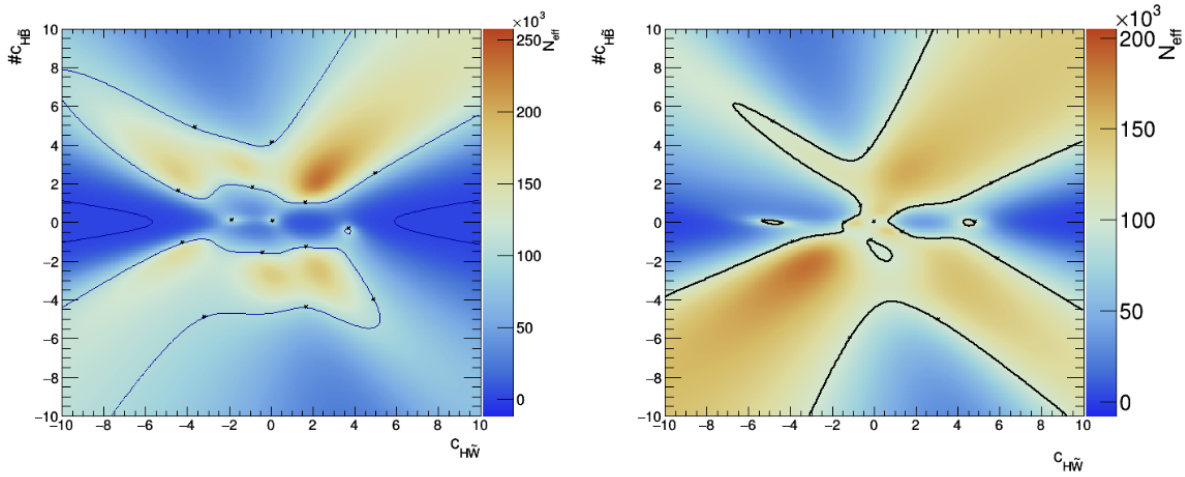

Рисунок 4.2 – Слева - распределение 2D VBF $cH\tilde{B}$ и $cH\tilde{W}$ построенное по базовым сэмплам без оптимизации. Справа - к полученном базису была применена стандартная оптимизация.

Остальные модули нужны преимущественно для работы этих троих основных или для вспомогательных методов, которые позволяют строить графики или получать стандартные базисы, находить стандартные базисы из данных моделирующих пакетов.

Однако наша задача не просто написать скрипт, а расширить возможности модуля.

Модуль написан на языке C++ с поддержкой библиотек ROOT, Eigen (библиотека линейной алгебры) и дополнительного модуля NtupleAnalysisUtils (Используется как вспомогательный метод при построении графиков).

4.2 Расширение возможностей модуля для реализации метода переопределённого морфинга

Вернёмся к алгоритму нахождения функции морфинга, описанного в разделе 2.1 и формулам (9 -11) раздела 2.2. Как уже было описано суть алгоритма в нахождении матрицы A. Чтобы это сделать нужно сначала определить полиномы $\vec{P}(\vec{g})$. Вопрос о том, как выбирать $\vec{g}$ базовых элементов, чтобы ожидаемая неопределённость выходного сигнала была минимальна в пределах некоторой интересующей области параметра, является нетривиальным и подробно рассмотрен в статье [4].

Общий подход к реализации поиска матрицы A в модуле «MorphingStudies» можно описать следующим образом:

1. Создаётся класс

```
template <int nProdOnly, int nDecayOnly, int
  ↪ nProdAndDecay> class MorphingCalculator{
```

2. Его конструктор

```
MorphingCalculator(const std::array<std::array<
  ↪ float, nCoefficients>, nBasisPoints> &
  ↪ basisCouplings);
```

где basisCouplings - это описываемая выше матрица со входными положениями точек сэмплов.

3. Можно так же воспользоваться другими методами например класса *H4lCPHelpers.h* для создания класса MorphingBasis, который по наследованию определяет и MorphingCalculator.

4. Далее с помощью внешних для этого класса функций и внутренних методов (getNBasisPoints, populateCouplingPolys toCouplingPolynomials,buildMorphingMatrix) рассчитывается матрица морфинга.

5. Конструктор класса и классовая функция buildMorphingMatrixм используют строго инициализированные массивы.

```
        Eigen::Matrix<float, nBasisPoints, nBasisPoints> m
          ↪ _morphMatrix;    /// Weight matrix used in
          ↪ the morphing
            Eigen::Matrix<float, nBasisPoints,
              ↪ nBasisPoints> m_weightMatrix;    ///
              ↪ Weight matrix used in the morphing
            std::array<std::array<float, nCoefficients>,
              ↪ nBasisPoints> m_basisPoints ={};  ///
              ↪ Basis points
            std::array<std::array<int, nCoefficients+1>,
              ↪ nBasisPoints> m_couplingsPowers = {};
```

6. Финальным действием эти массивы и полиномы (populateCouplingPolys toCouplingPolynomials) собираются в матрицу G в обозначениях формул (9-10) (или $\vec{P}(\vec{g})$ формулы (6)). Эти действия описаны строками 3-4 фрагмента кода ниже. И после этого необходимо получить саму искомую матрицу A. В исходной версии кода это решается методом стандартного морфингом (инверсией матрицы G, строка 8 кода ниже):

```
1       template <int nProdOnly, int nDecayOnly, int
          ↪ nProdAndDecay>
2       void MorphingCalculator<nProdOnly,nDecayOnly,
          ↪ nProdAndDecay>::buildMorphingMatrix(){
3         for (size_t col = 0; col < m_basisPoints.size
            ↪ (); ++col){
4           m_auxArr = toCouplingPolynomials<nProdOnly
              ↪ , nDecayOnly, nProdAndDecay>(m_
              basisPoints[col]);
```

```
5                   Eigen::Map<Eigen::Matrix<float,
                        ↪ nBasisPoints,1>> inputSample (&m_
                        auxArr[0]);
6                   m_morphMatrix.col(col) = inputSample.col
                        ↪ (0);
7               }
8               m_weightMatrix = m_morphMatrix.inverse();
9       }
```

Рассмотрев процесс в оригинальной версии кода теперь мы можем сказать, какие изменения необходимо сделать, чтобы реализовать метод переопределённого морфинга. А именно, заменить вычисление матрицы A на выражение полученное в формуле (11), что описывается следующей строкой кода:

```
m_weightMatrix = m_morphMatrix.transpose()*m_morphMatrix.
    ↪ inverse()*m_morphMatrix.transpose();
```

Однако данное решение будет не точным. И с ростом размера матрицы погрешность будет очень сильно расти. Чтобы решить эту проблемы была разработана функция псевдоинверсии матрицы с помощью метода сингулярного разложения. Данная Функция была построена используя методы библиотеки Eigen.

Стоит заметить, что этого не достаточно. Очевидно, что новая матрица будет размерности отличной от тех, что мы получали при использовании метода стандартного морфинга. Чтобы внесённые изменения в вычисления матрицы можно было наблюдать (добавить сэпмлов к базисной размерности матрицы) нужно переписать все шаблоны и шаблонные функции, которые были упомянуты в этом разделе или изменить типы данных со строгих к размеру типов std:array на динамические std:vector . Помимо определений в файлах *.header* необходимо обновить их использование в каждом другом файле модуля.

Данная задача была сначала выполнена для класса MorphingCalculator первым методом. Были проведены тесты и получены обратные матрицы рис.4.3.

Позже было решено всё же изменить подход фундаментально, чтобы не вносить неточность с физические параметры, а так же потенциально решить проблему с последствиями изменений в классе MorphingCalculator

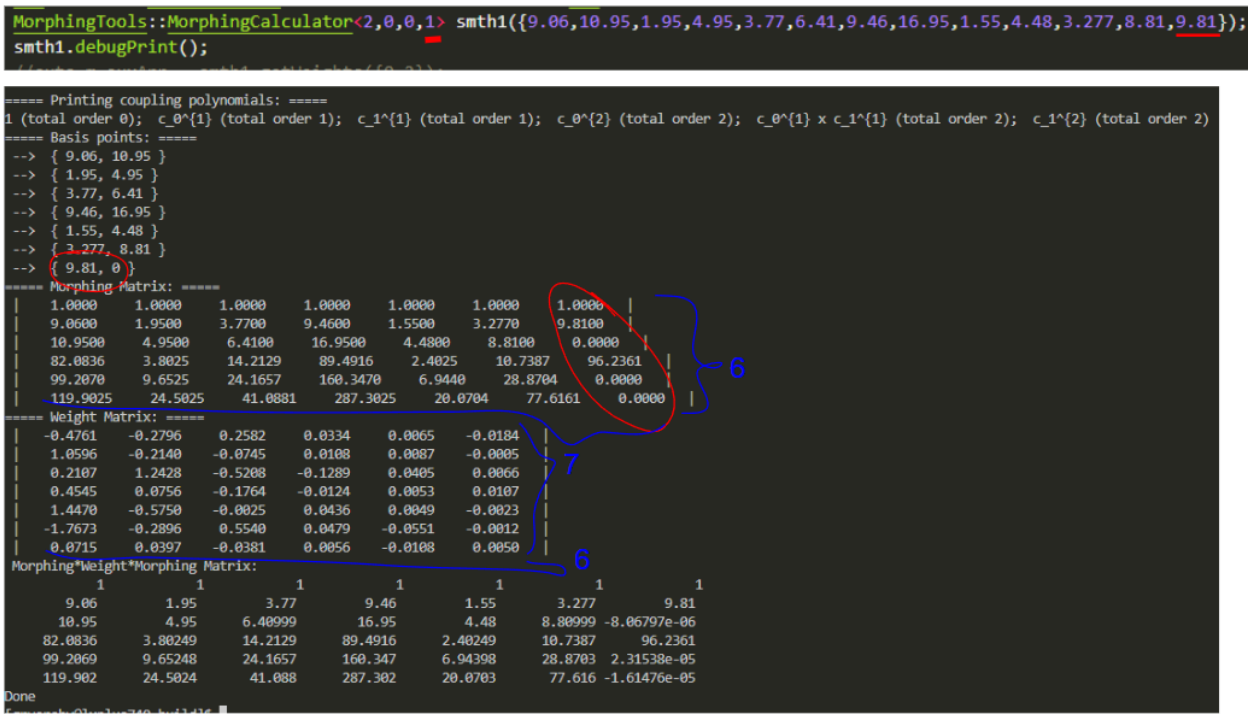

Рисунок 4.3 – Пример задания класса и дебаг принт вычисленной не квадратичной матрицы для случая 2D ggf (2 вершины рождения,0 распада,0 обеих одновременно и 1 дополнительны сэмпл

на другие классы. Однако в конечном итоге стало понятно, что изменять физически размеры массивов типа std::array некорректно в C++ и может привести к серьёзным проблемам с повреждением памяти вычислительного устройства. Метод был полностью реализован и проверен для всех классов. На данный момент компиляция проекта проходит успешно, однако класс оптимизации базиса при вызове оптимизации выдаёт ошибку пакета ROOT. Она связана с путаницей в указателях используемых в в классе MorphingCalculator и на данный момент ещё не решена.

5 Результаты работы скрипта и их анализ

Поскольку оптимизация базиса вызывает ошибки - результатов на данный момент получено не было. Однако независимым от этого модуля скриптом, разработанным одним из членом нашей группы, были получены частные подтверждения, что дополнительные сэмплы улучшают достоверность моделирования. С двумерными графиками этих частных случаев можно ознакомится на рис 5.1.

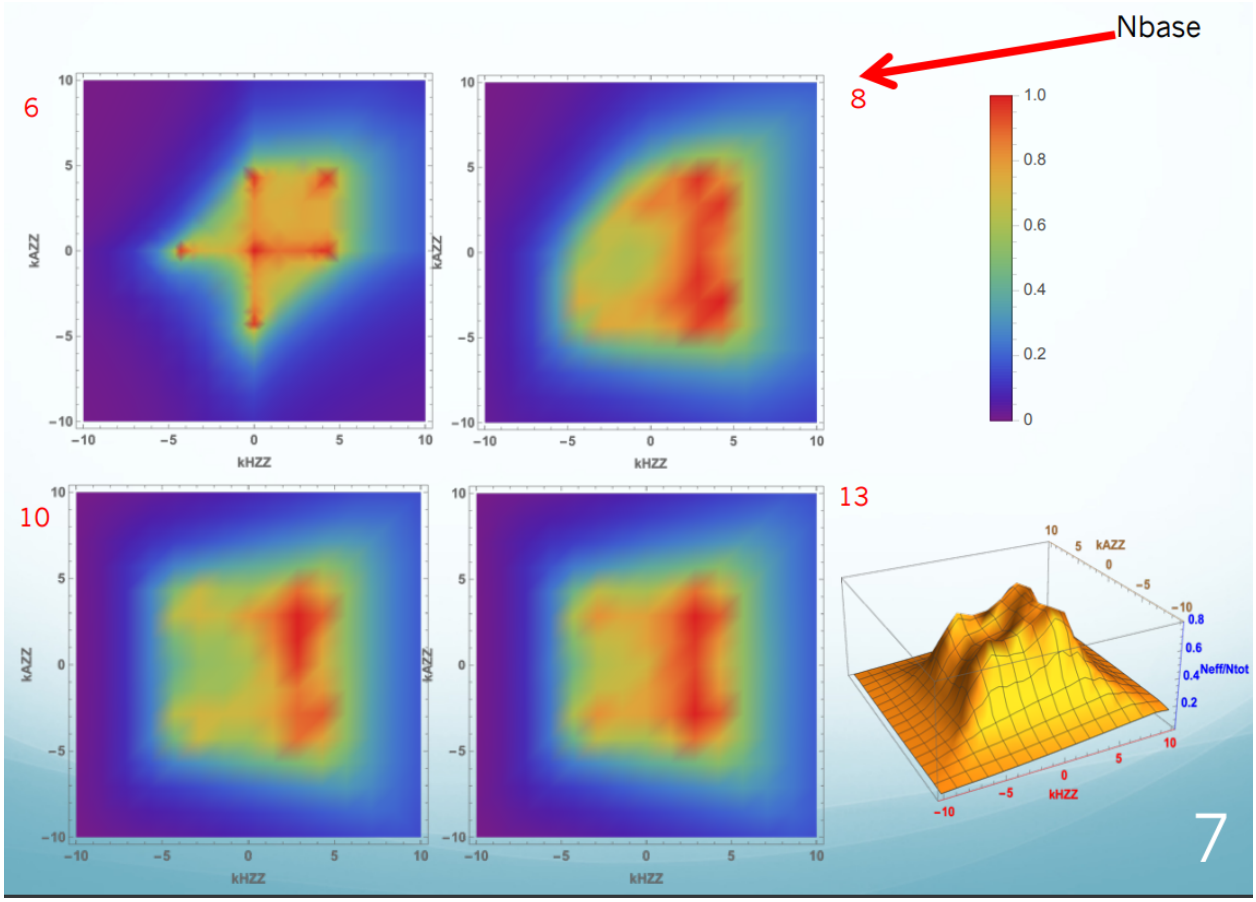

Рисунок 5.1 – Двумерные тепловые графики для разного количества сэмплов (при базисном для 2-ух коэффициентов Вильсона = 6)

6 Заключение

В данной работе был проведён небольшой обзор на проблемы Стандартной модели и современной физики частиц, и в частности рассмотрено направление исследования бозона Хиггса, эффективная теория поля, как метод поиска новой физики, а также метод морфинга, разработкой и усовершенствованием которого занимается наша группа в ATLAS. Метод упрощает процедуру моделирования разных параметров их корреляцию.

В рамках научной работы были продолжены разработка скрипта на языке C++ и усовершенствование для более общих случаев существующего модуля «MorphingStudies», рассчитывающего матрицу морфинга, необходимую моделирования параметров (например распределений коэффициентов Вильсона).

В работе приведены частные косвенные подтверждения, что новый метод переопределённого морфинга должен улучшить достоверность моделирования.

Отдельно стоит отметить, что были улучшены навыки программирования на языке C++ с использованием математических библиотек, а также опыт работы с модулями стандарта ATLAS Analysis Release и применение вычислительных методов в языке C++ с помощью библиотеки Eigen .

Список использованных источников

1. *Беляев Н.* Эффекты CP-нарушения и аномальные взаимодействия в хиггсовском секторе. — Диссертация на соискание учёной степени кандидата физико-математических наук, 2020.
2. *Pich A.* The Standard model of electroweak interactions. — https://arxiv.org/abs/0705.4264 : P. 1—49., arXiv, 2008.
3. *Belyaev N.*, *de Florian D.*, *et al.* Handbook of LHC Higgs Cross Sections: 4. Deciphering the Nature of the Higgs Sector. — CERN Monography, 2016.
4. *Kaluza A.*, *Belyaev N.*, *et al.* A morphing technique for signal modelling in a multidimensional space of coupling parameters. — 2015.
5. *Belyaev N. L.*, *Vorobyev G. I.* MorphingStudies. — https://gitlab.cern.ch/grvoroby/MorphingStudies, 2021.
6. *Goblirsch-Kolb M. E.*, *Yang Y. L.* MorphingStudies. — https://gitlab.cern.ch/HZZ/HZZSoftware/HZZCPStudy/Run2/MorphingStudies, 2021.
7. *Analysis Software Group.* ATLAS Software Documentation. — https://atlassoftwaredocs.web.cern.ch/ABtutorial/release_setup/, 2022.