Оценка эффективности моделирования физических сигналов с помощью модифицированного метода морфинга

Gregory Vorobyev

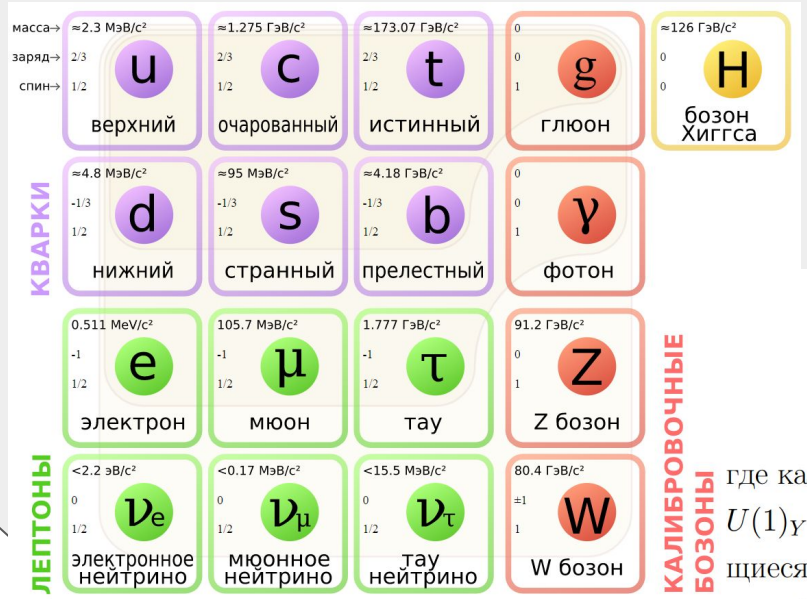
27.05.2022

теория всего

?

Новая физика?

Стандартная модель



EFT

(англ. Effective Field Theory)

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

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.

Morphing

$$T_{\text{out}}(\vec{g}_{\text{target}}) = \sum w_i(\vec{g}_{\text{target}}; \vec{g}_i) T_{\text{in}}(\vec{g}_i), \; = \vec{P}(\vec{g}) \cdot A\vec{T}$$

$$T_{\text{in},i}(g_{\text{SM},i}, g_{BSM,i}) \propto$$

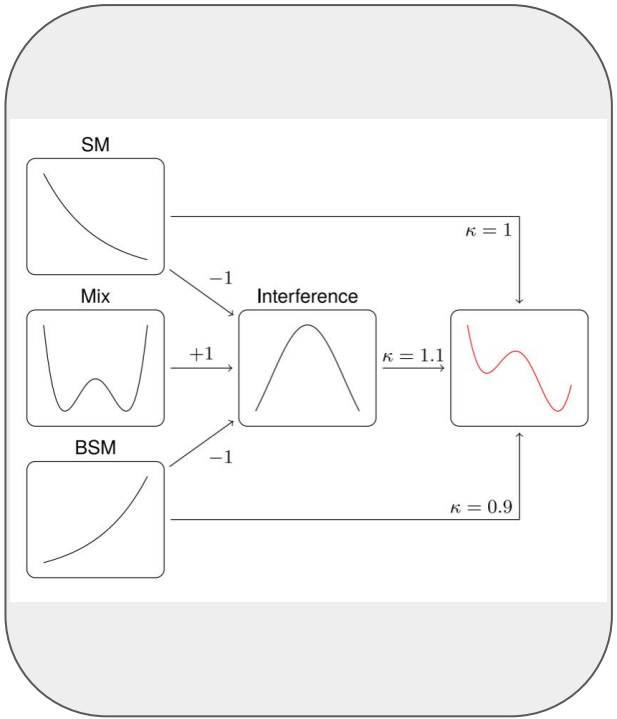
$$\propto g^2_{BSM,i}|\mathcal{O}_{BSM}|^2 + g^2_{\text{SM},i}|\mathcal{O}_{\text{SM}}|^2 + 2g_{\text{SM},i}g_{\text{SM},i}\mathcal{R}(\mathcal{O}^*_{\text{SM}}\mathcal{O}_{\text{SM}}), i = 1, \dots 3.$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} g^2_{\text{SM},1} & g^2_{\text{SM},2} & g^2_{\text{SM},3} \\ g^2_{BSM,1} & g^2_{BSM,2} & g^2_{BSM,3} \\ g_{\text{SM},1}g_{BSM,1} & g_{\text{SM},2}g_{BSM,2} & g_{\text{SM},3}g_{BSM,3} \end{pmatrix} = \mathbf{I}$$

$$\Leftrightarrow A \cdot G = \mathbf{I}$$

- +: Из “базисных” сэмплов (распределений) получаем желаемое.
- -: “базис” надо оптимизировать

- Надо найти матрицу A



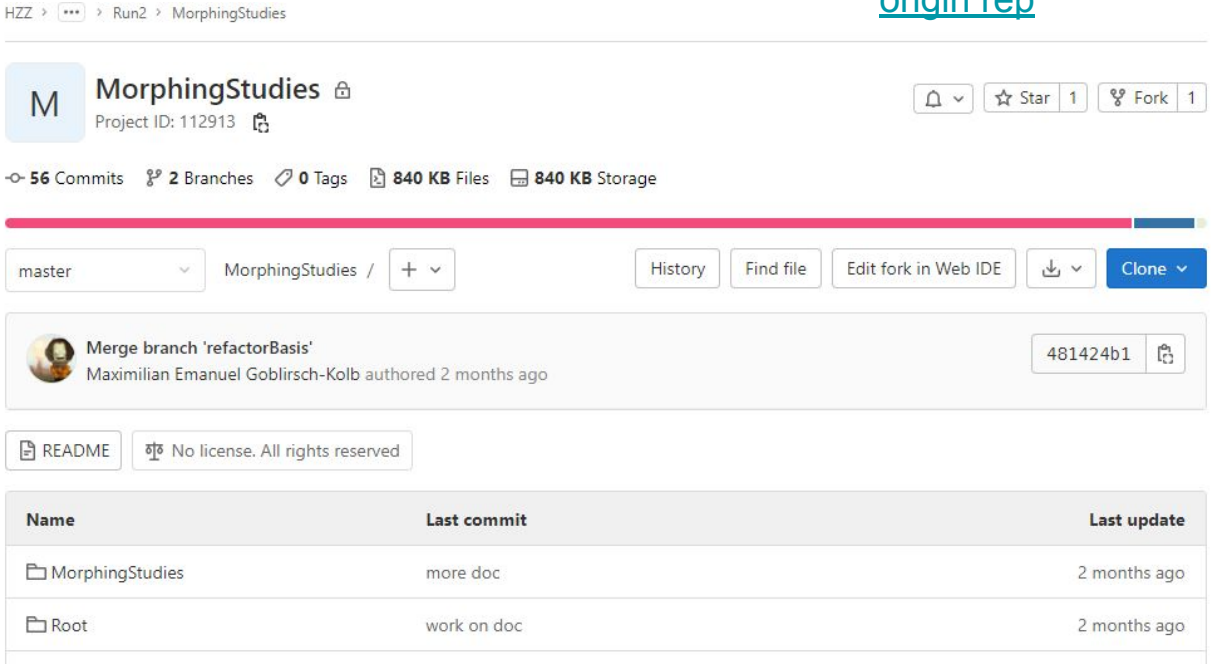
Цели

- Изучить структуру и использование
- Модифицировать классы для использования с дополнительными сэмплами (По умолчанию пакет использует минимальное или базисное кол-во сэмплов)

- Изучить эффективность метода (как количество дополнительных сэмплов качественно влияет на результирующие распределения)

MorphingStudies

sources: [my rep](#) (branch: *expand-sample-dim)*
[origin rep](#)



Суть модуля:

The package is intended to provide EFT morphing functionality in plain ROOT, outside RooFit. © *ReadMe*

Использовать базовые сэмплы классами алгебры морфинга, классами базисной оптимизации (основной интерес) и визуализации.

Скрипт оптимизации базиса и создания графиков: Загрузка

```cpp
///In this scrip we have two sample sets VBF 2D (then nessesary number of points is 15)
std::array<std::shared_ptr<TH1F>,15> generateDummyHistograms(){
    std::vector<float> yields {
        0.41306, 3.148, 0.81035, 2.4365, 15.237, 4.2566, 4.0504, 40.74, 35.234, 3.0092, 10.486, 72.584, 79.002, 67.42, 215.78
    };

    std::array<std::shared_ptr<TH1F>,15> output={};
    for (size_t k = 0; k < output.size(); ++k){
        output[k] = std::make_shared<TH1F>(Form("basisHisto_%lu",k), ";total rate;rate",1,0.,1.);
        output[k]->SetBinContent(1,yields[k]);
        output[k]->SetBinError(1,0.01 * yields[k]); // this is the 1% stat errors
        MorphingMetrics::enforceNeffTotal(output[k].get(), 100.e3); // definition in ../Root/Morph
        // generally it change each error to error/sqrt(100.e3/(centralVal²/uncertainty²)),
        // where centralVal is integratedYield and uncertainty is integratedErr (../MorphingStudie
    }
    return output;
}
```

В этом примере:
VBF 2D (0,0,2)

```cpp
// get the basis histos
std::array<std::shared_ptr<TH1F>,15> dummyHistos = generateDummyHistograms();
    std::array<std::array<float,2>, 15> sampleLocations{
    std::array<float,2>{0.07, 0.07},
    std::array<float,2>{1.67, -1.3},
    std::array<float,2>{-1.86, 0.09},
    std::array<float,2>{3.72,  -0.32},
    std::array<float,2>{-4.43, 1.59},
    std::array<float,2>{-0.87, 1.78},
    std::array<float,2>{-0.39, -1.57},
    std::array<float,2>{0.05,  4.1},
    std::array<float,2>{1.67, -4.41},
    std::array<float,2>{1.66, 0.98},
    std::array<float,2>{-4.2, -1.06},
    std::array<float,2>{4.98,  2.52},
    std::array<float,2>{4.92, -4.01},
    std::array<float,2>{-3.63, 4.87},
    std::array<float,2>{-3.2,  -4.91},
};

/// Now, we can construct a morphing basis!
/// We give it our locations (couplings) and the corresponding histos (cross-sect
/// We also need to tell it what we would like to morph (here: TH1F objects)
/// The numbers are the number of prod / decay / prod-and-decay coefficients to m
// 0,0,2 - means nProd=0, nDecey=0 nProdAndDecey=2 (VBF 2D)
MorphingTools::MorphingBasis<TH1F,0,0,2> theBasis(sampleLocations, dummyHistos);

/// and we are done, print this basis
theBasis.printBasis();
```

util/Examples/Custom_basis_Optimizing_and_Plotting.cxx

`MorphingTools::getNBasisPoints(nProd,nDesey,nBoth)` - возвращает число базисных сэмплов из *nProd,nDecay,nBoth* - целые

взято из примера *util/Examples/Loading_Custom_Basis.cxx*

Оптимизация базиса и создание графиков: Оптимизация

Главные составляющие оптимизации:

- генератор смещённых базисов
- размерная сетка
- метод оценки качества

```
BasisGenerator<0,0,2> myGenerator{
    /// step sizes: 0.03 + 0.03 * nfailed in each direction
    {
        std::pair<float,float>{0.03,0.03},
        std::pair<float,float>{0.03,0.03},
    },
    /// allow each coordinate to run up to 6 at most (abs max of couplings meaning)
    // Just range of the generator
    {
        6,6
    },
    /// We emulate that each generated basis has N_eff = 100.e3
    100.e3
};
// 5x5 test points in -1...1
ScanGrid<0,0,2> myGrid;
myGrid.addRectangle({
    std::make_pair(5, 1.f),
    std::make_pair(5, 1.f),
});
// And then let's add a coarser grid for large couplings, to also
// include these points at a lower priority
myGrid.addRectangle({
    std::make_pair(5, 5.f),
    std::make_pair(5, 5.f),
});
```

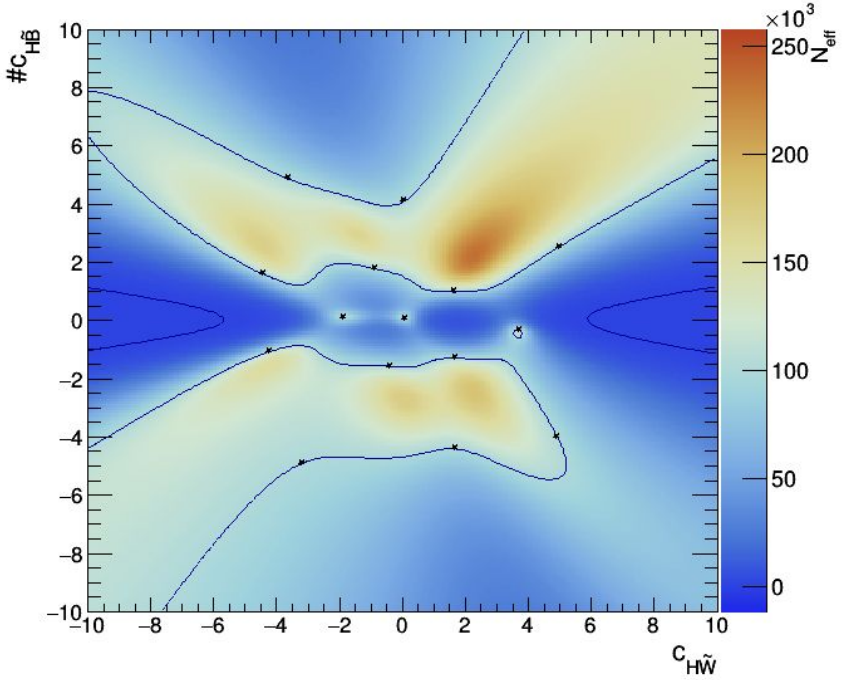
```
/// Now we still need to set up how to score a basis. A higher score is better.
BasisScorer<TH1F, 0,0,2> myScorer{
    [](const Plot<TH1F> h, const std::array<float,2> & point){
        return  MorphingMetrics::getNeffTotal(h()) / (1. + 0.02 * (point[0]*point[0]+point[1]*point[1]));
    },
    // as an example, we return the n_effective per point, and upweigh those points
    // close to the SM (to further enhance their importance).

    /// @tparam h - is a our histogramm. We calculate NeffTotal as (centralVal²/uncertainty²),
    // where <centralVal> is integratedYield and <uncertainty> is integratedErr
    //(../MorphingStudies/MorphingMetrics.h line 11).

    10.
    // this assigns a relative weight against the condition number (which acts as a tie breaker)
};

/// instantiate the tuner
BasisTuner<TH1F,0,0,2> MyTuner(
    theBasis,
    myGrid,
    myGenerator,
    myScorer,
    5000     /// here, we just run 5000 iterations
    // so if you want see only score, then boot only 1 iteration
);

// run the tuning
MyTuner.runTuning();

// and print the outcome.
// MyTuner.getCurrentBasis().printBasis();

// The printout will display a NEGATIVE score - this is intended, because the implementation
// behind the scenes minimises a score including the negative of the myScorer output.
```

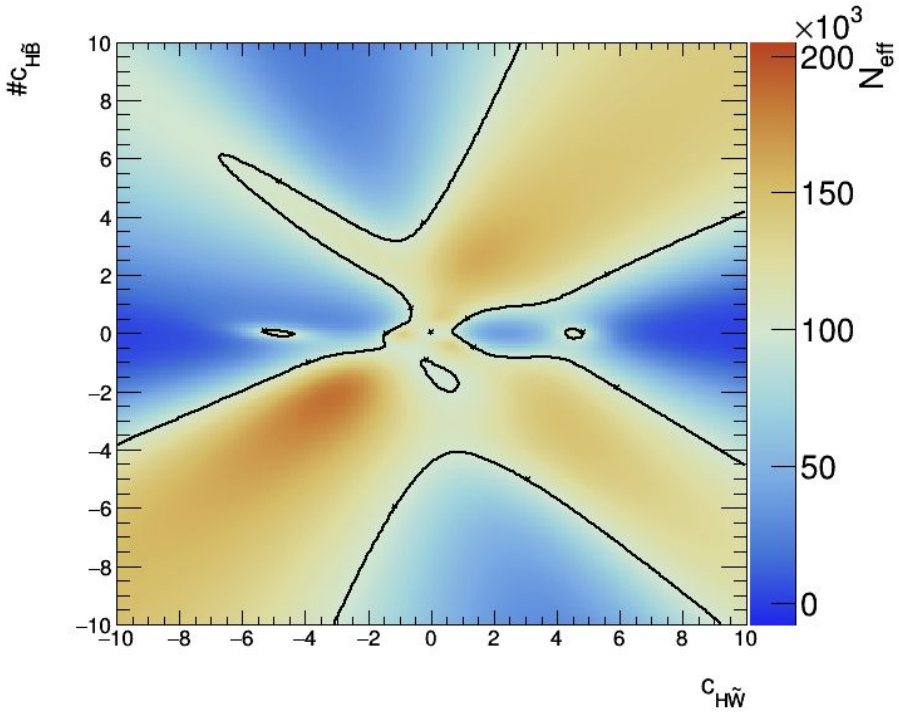
Оптимизация базиса и создание графиков: Графики

```
ProjectionPlane<2,2> scanPlane_cHWtil_cHBtil  {
    {
        ProjectionHelpers::ProjectionAxis<2>{{1.,0.}, "c_{H#tilde{W}}", 10},     ///x axis of 2D plot
        ProjectionHelpers::ProjectionAxis<2>{{0.,1.}, "#c_{H#tilde{B}}", 10},     // y axis of 2D plot
    },
    {0.,0.}  // origin of 2D plot
};


/// then we can call the "plot2D" method, giving it our basis, the projection, the metric we would like,
/// and an output file name
/// Argument 4 is an optional handle for multi-page-PDF file writing.
/// Argument 5 is a set of contour levels to superimpose.
//  The last argument steers the visuals of the canvas.
CanvasOptions myCanvasOpts = CanvasOptions().ColorPalette(kLightTemperature).RightMargin(0.25).TopMargin(0.07).OutputFormats({"png"});
myCanvasOpts.ZAxis.modify().Title("N_{eff}");

// option for creating common file with plots
// use  auto mpc = PlotUtils::startMultiPagePdfFile("BasisTuning_VBF_3D");  for creating common pdf file
auto mpc = nullptr;
//for MorphingPlots::metricToPlot:: <your_parametr> are few options: NeffTotal, NeffMin, SigmaRelMax, Yield
//{100.e3,1.e3,10,1} - level lines will be writen on this meanings of <your_parametr>
MorphingPlots::plot2D<TH1F,2>(theBasis, scanPlane_cHWtil_cHBtil, MorphingPlots::metricToPlot::NeffTotal,"Neff_cHWtil_cHBtil_startBasis",mpc,{100.e3,1.e3,10,1}, myCanvasOpts);
MorphingPlots::plot2D<TH1F,2>(newBasis, scanPlane_cHWtil_cHBtil, MorphingPlots::metricToPlot::NeffTotal,"Neff_cHWtil_cHBtil_newBasis",mpc,{100.e3,1.e3,10,1}, myCanvasOpts);
return 0;
```

Оптимизация базиса и создание графиков: Графики

2D VBF (исходный => новый)





Overdetermined Morphing:

Исходная программа (стандартный Morphing) использует **только базисные** (**минимальное** количество) сэмплы и игнорирует любые дополнительные.

Overdetermined Morphing использует дополнительные сэмплы и решает связанные с ними некоторые математические трудности.

Standard Morphing case:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} g^2_{\mathrm{SM},1} & g^2_{\mathrm{SM},2} & g^2_{\mathrm{SM},3} \\ g^2_{B\mathrm{SM},1} & g^2_{B\mathrm{SM},2} & g^2_{B\mathrm{SM},3} \\ g_{\mathrm{SM},1}g_{B\mathrm{SM},1} & g_{\mathrm{SM},2}g_{B\mathrm{SM},2} & g_{\mathrm{SM},3}g_{B\mathrm{SM},3} \end{pmatrix} = \mathbf{I}$$

$$\Leftrightarrow A \cdot G = \mathbf{I} \Rightarrow A = G^{-1} \quad (1)$$

Overdetermined Morphing case:

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

$$= \mathbf{I} \Leftrightarrow A \cdot G = \mathbf{I}\text{ , but } G^{-1} \text{ doesn't exist, } A = ? \quad (2)$$

solution:

$$\vec{x} = (G^T G)^{-1} G^T \vec{b}\text{ , where } (G^T G)^{-1} G^T \text{ is the pseudo-inverse of } G.$$

$$A = (G^T G)^{-1} G^T \quad (3)$$

Overdetermined Morphing: nExtraPoints

```
template <int nProdOnly, int nDecayOnly, int nProdAndDecay, int nExtraPoints>
    void MorphingCalculator<nProdOnly,nDecayOnly,nProdAndDecay,nExtraPoints>::buildMorphingMatrix(){
        // perform column-wise copy of the input array into the morphing matrix (needs translation to polynomials)
        for (size_t col = 0; col < m_basisPoints.size(); ++col){ //@ for OM: m_basisPoints.size() will be increased by addSamplesPoints
            //  convert each sample
            m_auxArr = toCouplingPolynomials<nProdOnly, nDecayOnly, nProdAndDecay>(m_basisPoints[col]);
            // then plug it into an eigen vector (no re-allocation)
            Eigen::Map<Eigen::Matrix<float,nBasisPoints,1>> inputSample (&m_auxArr[0]);
            // and add it to the matrix
            m_morphMatrix.col(col) = inputSample.col(0);
        }
        // now, obtain the weight matrix by inverting
        m_weightMatrix = pseudoInverse(m_morphMatrix); //@ for OM: SVD-based solution
        //@ for OM: m_morphMatrix.inverse() ->
            //  1) m_morphMatrix.transpose()*m_morphMatrix).inverse()*m_morphMatrix.transpose(); - not precise method
            //  2) m_morphMatrix.completeOrthogonalDecomposition().pseudoInverse();  - doesn`t work correct any time
}
```

polynomial

matrix of polynomials

```
===== Morphing Matrix: =====
|    1.0000     1.0000     1.0000   |
|    9.0600    10.9500     1.9500   |
|   82.0836   119.9025     3.8025   |
===== Weight Matrix: =====
|   -1.6068     0.9708    -0.0754   |
|    1.0503    -0.6545     0.0594   |
|    1.5674    -0.3166     0.0189   |
```

И каждый следующий класс, метод и функции класса должны быть изменены:

`template <int nProdOnly, int nDecayOnly, int nProdAndDecay>` to
`template <int nProdOnly, int nDecayOnly, int nProdAndDecay, int nExtraPoints>`

(Если в этом есть необходимость)

Overdetermined Morphing: Псевдоинверсия

```
// now, obtain the weight matrix by inverting
m_weightMatrix = pseudoInverse(m_morphMatrix); //@ for OM: SVD-based solution
//@ for OM: m_morphMatrix.inverse() ->
    //  1) m_morphMatrix.transpose()*m_morphMatrix).inverse()*m_morphMatrix.transpose(); - not precise method
    //  2) m_morphMatrix.completeOrthogonalDecomposition().pseudoInverse();  - doesn`t work correct any time
```

```
namespace MorphingTools{
    ///  SVD-based solution for method to compute the (Moore-Penrose) pseudo inverse (http://eigen.tuxfamily.org/index.php?title=FAQ)
    template<typename MatType>
    using PseudoInverseType = Eigen::Matrix<typename MatType::Scalar, MatType::ColsAtCompileTime, MatType::RowsAtCompileTime>;

    template<typename MatType>
    PseudoInverseType<MatType> pseudoInverse(const MatType &a, double epsilon = std::numeric_limits<double>::epsilon()){
        using WorkingMatType = Eigen::Matrix<typename MatType::Scalar, Eigen::Dynamic, Eigen::Dynamic, 0, MatType::MaxRowsAtCompileTime, MatType::MaxColsAtCompileTime>;
        Eigen::BDCSVD<WorkingMatType> svd(a, Eigen::ComputeThinU | Eigen::ComputeThinV);
        svd.setThreshold(epsilon*std::max(a.cols(), a.rows()));
        Eigen::Index rank = svd.rank();
        Eigen::Matrix<typename MatType::Scalar, Eigen::Dynamic, MatType::RowsAtCompileTime,
                0, Eigen::BDCSVD<WorkingMatType>::MaxDiagSizeAtCompileTime, MatType::MaxRowsAtCompileTime>
        tmp = svd.matrixU().leftCols(rank).adjoint();
        tmp = svd.singularValues().head(rank).asDiagonal().inverse() * tmp;
        return svd.matrixV().leftCols(rank) * tmp;
    } // for some reason this func slow down cmake compilation a lot.
```

Overdetermined Morphing: Результаты

```
MorphingTools::MorphingCalculator<2,0,0,1> smth1({9.06,10.95,1.95,4.95,3.77,6.41,9.46,16.95,1.55,4.48,3.277,8.81,9.81});
smth1.debugPrint();
```

```
===== Printing coupling polynomials: =====
1 (total order 0);  c_0^{1} (total order 1);  c_1^{1} (total order 1);  c_0^{2} (total order 2);  c_0^{1} x c_1^{1} (total order 2);  c_1^{2} (total order 2)
===== Basis points: =====
 -->  { 9.06, 10.95 }
 -->  { 1.95, 4.95 }
 -->  { 3.77, 6.41 }
 -->  { 9.46, 16.95 }
 -->  { 1.55, 4.48 }
 -->  { 3.277, 8.81 }
 -->  { 9.81, 0 }
===== Morphing Matrix: =====
|    1.0000     1.0000     1.0000     1.0000     1.0000     1.0000     1.0000  |
|    9.0600     1.9500     3.7700     9.4600     1.5500     3.2770     9.8100  |
|   10.9500     4.9500     6.4100    16.9500     4.4800     8.8100     0.0000  |
|   82.0836     3.8025    14.2129    89.4916     2.4025    10.7387    96.2361  |
|   99.2070     9.6525    24.1657   160.3470     6.9440    28.8704     0.0000  |
|  119.9025    24.5025    41.0881   287.3025    20.0704    77.6161     0.0000  |
===== Weight Matrix: =====
|   -0.4761    -0.2796     0.2582     0.0334     0.0065    -0.0184  |
|    1.0596    -0.2140    -0.0745     0.0108     0.0087    -0.0005  |
|    0.2107     1.2428    -0.5208    -0.1289     0.0405     0.0066  |
|    0.4545     0.0756    -0.1764    -0.0124     0.0053     0.0107  |
|    1.4470    -0.5750    -0.0025     0.0436     0.0049    -0.0023  |
|   -1.7673    -0.2896     0.5540     0.0479    -0.0551    -0.0012  |
|    0.0715     0.0397    -0.0381     0.0056    -0.0108     0.0050  |
Morphing*Weight*Morphing Matrix:
           1          1          1          1          1          1          1
        9.06       1.95       3.77       9.46       1.55      3.277       9.81
       10.95       4.95    6.40999      16.95       4.48    8.80999 -8.06797e-06
     82.0836    3.80249    14.2129    89.4916    2.40249    10.7387    96.2361
     99.2069    9.65248    24.1657     160.347    6.94398    28.8703 2.31538e-05
      119.9    24.5024       41.088     287.302    20.0703     77.616 -1.61476e-05
Done
```

6

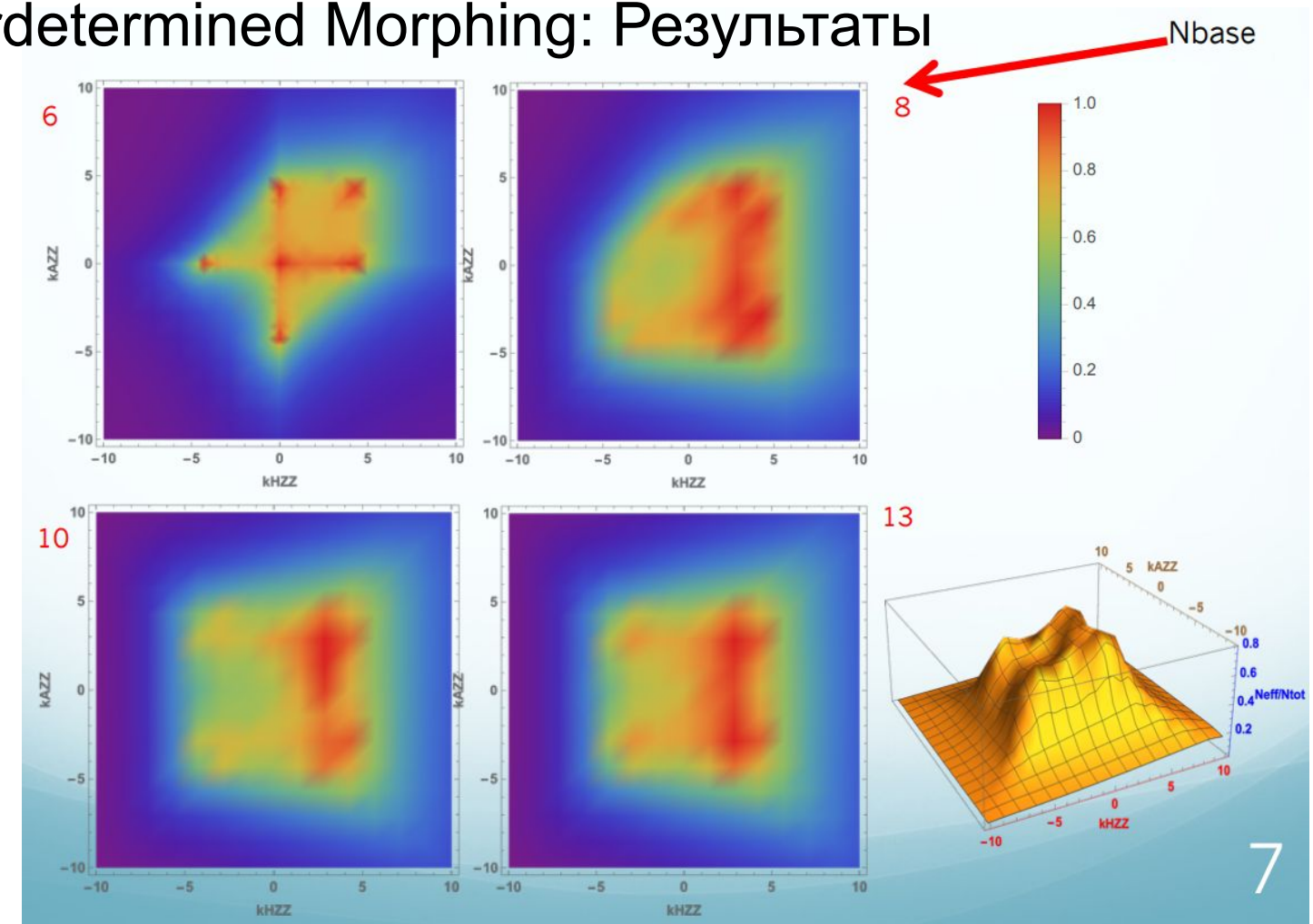
7

6

2D ggf

n.Prod=2,
n.Decay=0,
n.Both=0.
n.Extra
Samples = 1

Overdetermined Morphing: Результаты



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 начаты 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 .

Overdetermined Morphing

Для неквадратных матриц и вырожденных матриц обратных матриц не существует:

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

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

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