2 Higgs Doublet Model Evolver - Manual

Joel Oredsson†
Department of Astronomy and Theoretical Physics, Lund University,
Sölvegatan 14A, SE 223-62 Lund, Sweden

Abstract

2 Higgs Doublet Model Evolver (2HDMEv) is a C++ program that provides the functionality to perform renormalization group equation running of the general 2 Higgs Doublet Model at 2-loop order. We briefly describe the 2HDMEv’s structure, provide a demonstration of how to use it and list some of the most useful functions.
1 Introduction

The 2 Higgs Doublet Model (2HDM) is a very popular extension of the standard model. It offers a rich phenomenology and many new parameters; for a review see ref. [1]. A useful tool when investigating the 2HDM is to employ a renormalization group (RG) analysis. One can with such a method look for instabilities, fine-tuning and valid energy ranges in the 2HDM’s parameter space.

The purpose of 2 Higgs Doublet Model Evolver (2HDME) is to provide a fast C++ application programming interface (API) that can be used to evolve the 2HDM in renormalization scale. 2HDME works with the general, potentially complex and CP violating, 2HDM and both 1- and 2-loop RGEs are implemented.

In this manual, we give instructions and showcase some of 2HDME’s functionalities. The source code can be found at ref. [2] and we give some installation instructions in appendix A.
For a physics discussion, we refer to ref. [3]; which employ 2HDME to analyze $\mathbb{Z}_2$ breaking effects in the evolution of 2HDM.

We begin by briefly describing the structure of 2HDME in section 2 and a demonstration of how to use the API is given in section 3. Further details of the base classes and the functionality they provide are then given in section 4. The main class of 2HDME is THDM which is described in section 5, where we also give a short review of the physics of the 2HDM. Installation instructions are given in appendix A.

2 Structure of 2HDME

2HDME is written in C++11 and depends on GSL [4] for numerically solving the RGEs as well as on Eigen [5] for linear algebra operations. See appendix A for installation instructions. All source code should be fairly well documented with comments in the header files, which show the functionality of all the classes. 2HDME originated from an extension of 2HDMC [6] and hence share a similar structure.

The purpose of 2HDME is to provide an API that consists of methods to manipulate a 2HDM model; thus the idea is that the user should write their own executable code that uses the THDM class of 2HDM. A simple example that demonstrates how to use 2HDME is provided in src/demos/DemoRGE.cpp; which is explained in more detail in section 3.

The main class of the 2HDME is the THDM object that describes a general, potentially complex, 2HDM; see section 5 for a more detailed description of its functionality. For basic usage, one only needs to interact with the public methods of THDM (and the SM class to set boundary conditions). The framework to solve RGEs is contained in the RgeModel class, which THDM inherits from. The RgeModel, furthermore, inherits some basic functionality related to data and console output from the abstract class BaseModel.

To create a THDM, one needs to specify the Standard Model (SM) input. This is done through a SM class that describes the SM. It is by default initialized at the top mass scale, $\approx 173$ GeV, see appendix C for a detailed list of all the values. Similar to the THDM, the SM also inherits RGE functionality from the RgeModel class, with its own set of RGEs. Thus, if one wants the SM parameters at another renormalization scale, the SM can be evolved to an arbitrary energy scale. However, one should only run models above the top mass scale, since the RGEs specified for the SM include all particles.

After being given a SM object to setup the fermion masses, CKM matrix, gauge couplings and renormalization scale, the THDM needs a scalar potential and a Yukawa sector. The scalar sector can be specified with any of the 2HDM bases in section B; these bases are separate structs that are defined in THDM_bases.cpp. Note though that THDM internally works in the generic basis.

The Yukawa sector can be specified in three ways. One option is to impose a $\mathbb{Z}_2$ symmetry; type-I,-II,-III(X) or -IV(Y). Another is to use a flavor alignment ansatz, where the Yukawa matrices for the different Higgs fields are proportional to each other. Of course one can also set the Yukawa matrices manually as a third option.

To evolve the THDM or SM, simply use evolve(). The options for the RG evolution can be set with the RgeConfig struct.

1Running the SM downwards in energy below the top mass scale should incorporate some mechanism for integrating out particles at their corresponding mass threshold. This is currently not implemented in 2HDME.
2.1 2-loop RGEs

The 2-loop RG equations (RGEs) for massless parameters of any renormalizable quantum field theory in 4 dimensions were derived in the seminal papers by Machacek and Vaughn [7, 8, 9]. That work has been supplemented with the 2-loop RGEs of massive parameters in ref. [10], which is the source that we have used to derive the 1- and 2-loop RGEs for a general 2HDM. Note though, that when working with quantum field theories with multiple indistinguishable scalar fields one must be careful when interpreting their formulas, since the formulas in ref. [7, 8, 9, 10] are written for the case of an irreducible representation of the scalar fields. In the case of a general 2HDM, one gets non-diagonal anomalous dimensions that mixes the scalar fields during RG evolution.

The general 2-loop RGEs are very long and we will thus not write them down here, but are instead provided as supplementary material and are of course also available in C++ format in the source code of 2HDME.

3 Demonstration of usage

As an example of how to use the 2HDME API, we here go through the DemoRGE in src/demos, where a 2HDM is initialized at the top mass scale and then evolved up to the Planck scale. For instructions of how to install and run the demo, see appendix A.

The first thing to note is that the 2HDME is wrapped in a namespace, thus including using namespace THDME is convenient.

To initialize the THDM, we first have to create a SM. It is constructed at the top mass scale and we print its parameters to the console with

```cpp
SM sm;
sm.print_all();
```

This SM is used to initialize the CKM matrix, fermion mass \( \kappa_F \) matrices, gauge couplings, vacuum expectation value (VEV) \( v \) and the renormalization scale of THDM. For instructions of how to obtain a SM at another renormalization scale, see section 4.3.

To create a THDM and feed it a SM, use

```cpp
THDM thdm(sm);
```

Next up, we need to set the scalar potential. This can be done with any of the bases in section B. The bases are defined as struct objects which have member functions that can be used to convert one base to another. Here, we use the generic basis which we specify by

```cpp
BaseGeneric gen;
gen.beta = 1.46713;
gen.M12 = std::complex<double>(3132.85, 0.);
```

---

2. This is a subtlety that is also discussed in ref. [11, 12].
3. For more details about the RGEs of different renormalization schemes in theories with multiple indistinguishable scalar fields, we refer to ref. [13].
4. They are collected in separate header files in src/RGEs.
5. There is also the VEV phase \( \xi \) which is automatically initialized to zero; furthermore, it is fixed by the tadpole equations when actually setting the THDM potential.
gen.Lambda1 = 0.413702;
gen.Lambda2 = 0.263926;
gen.Lambda3 = 0.13313;
gen.Lambda4 = -0.0444794;
gen.Lambda5 = std::complex<double>(0.29586, 0.);
gen.Lambda6 = std::complex<double>(0., 0.);
gen.Lambda7 = std::complex<double>(0., 0.);

We set the potential with

```cpp
thdm.set_param_gen(gen);
```

Now that we have initialized the THDM and the SM, we can save them in SLHA-like text files with

```cpp
sm.write_slha_file();
thdm.write_slha_file(0);
```

The SLHA file from THDM can be used as an input file to SPheno and the argument for the THDM specifies the loop order that SPheno will use to calculate loop corrected masses; valid arguments are 0,1,2 corresponding to tree-level, 1- and 2-loop order.

To evolve the THDM, we need to configure the settings of the RG evolution. This is done by creating a RgeConfig struct and feeding it to the THDM like

```cpp
RgeConfig options;
options.dataOutput = true;
options.consoleOutput = true;
options.evolutionName = "DemoRGE";
options.twoloop = true;
options.perturbativity = true;
options.stability = false;
options.unitarity = false;
options.finalEnergyScale = 1e18;
options.steps = 100;
thdm.set_rgeConfig(options);
```

The different options are explained in section 4.2. One can print the options to the console with `options.print()`. Now, we are ready to evolve the THDM with

```cpp
thdm.evolve();
```

which should only take a couple of seconds at 2-loop order. The parameters as functions of the renormalization scale are listed in `output/DemoRGE/data` and basic plots are created in `output/DemoRGE/plots`, see figure 1 for an example.

Afterwards we can print the parameters at the energy scale where the RG running stopped with

```cpp
thdm.print_all();
```

To retrieve the scalar potential, one can for example use
Next up, we can evolve the THDM to another energy. First we save the THDM at the Planck scale with

```c++
thdm.write_slha_file(sphenoLoopOrder, "DemoRGE_evolvedThdm");
```

After the fist evolution, the THDM is specified at the Planck scale and to evolve the it downwards to $\mu = 1$ TeV, all one have to do is to change the `finalEnergyScale` of its `RgeConfig` before evolving, which is achieved with

```c++
options.finalEnergyScale = 1e3;
options.dataOutput = false;
thdm.set_rgeConfig( options);
thdm.evolve();
```

where we also changed `dataOutout` to `false`; thus preventing that the first plots are overwritten.

![Figure 1: The evolution of quartic couplings in the generic basis produced by DemoRGE.cpp.](image)

4 Classes

There are two main classes of 2HDM: THDM and SM. These inherit the features from the base classes `BaseModel` and `RgeModel` like

```latex
BaseModel \rightarrow RgeModel \rightarrow THDM,
BaseModel \rightarrow RgeModel \rightarrow SM.
```

We briefly describe the features of the `BaseModel`, `RgeModel` and `SM` in the following subsections, while the THDM is described in more detail in section 5.
4.1 BaseModel class

The BaseModel is an abstract class that offers some basic functionality such as input and output to data files as well as to the console.

The level of information printed to the console of the THDM and SM during computations can be set by `set_logLevel(LogLevel lvl)`, where `lvl` can be either one of the following:

- **LOG_INFO**: Prints information of calculations performed and status updates.
- **LOG_ERRORS**: Only prints error messages.
- **LOG_WARNINGS**: Prints error messages and warning messages.
- **LOG_DEBUG**: Prints all the information as well as additional debugging information.

4.2 RgeModel class

The RgeModel inherits the input/output functionality of the BaseModel and acts as a base class which offers a framework to incorporate RG evolution in derived classes; both THDM and SM are derived classes of RgeModel.

If one wants to create a new type of class that implements RGE functionality similar to THDM and SM, it is easy to use RgeModel. For example one might want to extend 2HDM with additional operators and consequently new parameters and RGEs. What one needs to do is

- First, make the new class inherit from RgeModel, i.e. override all its virtual functions.
- The new class’s RGEs must be provided as an ODE system like the functions in RGE.cpp.

See the header file RgeModel.h for a list of all member functions. Some of the most important ones are:

| Function                  | Description                                      | Returns    |
|---------------------------|--------------------------------------------------|------------|
| set_rgeConfig(RgeConfig)  | Sets up the options for the RG evolution.        | void       |
| get_rgeConfig()           | Retrieves rgeConfig                              | RgeConfig  |

The options for performing RG evolution are saved in the RgeConfig member variable rgeConfig. This RgeConfig has the following member variables:

- **bool twoloop**: If true, uses 2-loop RGEs; otherwise uses 1-loop.
- **bool perturbativity**: If true, the RG evolution stops when perturbativity is violated.
- **bool unitarity**: If true, the RG evolution stops when unitarity is violated.
- **bool stability**: If true, the RG evolution stops when stability is violated.
- **bool consoleOutput**: If true, prints information to the console during and after RG evolution.
- **string evolutionName**: Name of folder in output, where data and plots are stored.
• **bool dataOutput**: If true, creates data files in `output/"evolutionName"/data` folder. These files contain the parameters of the model at each step in the RG evolution. If `GNUPLOT` is enabled in the `Makefile`, simple plots are created in `output/"evolutionName"/plots`.

• **double finalEnergyScale**: The final energy scale, in GeV, for the RG running (from current renormalization scale). This can be both higher as well as lower than the current renormalization scale.

• **int steps**: Number of steps for the RG evolution; which are logarithmically distributed. Perturbativity, unitarity and stability are being checked at each step.

| Function                | Description                        | Returns       |
|-------------------------|------------------------------------|---------------|
| evolve()                | Evolves the model in renormalization scale. | true/false  |
| evolve_to(double)       | Evolves the model given scale       | true/false   |

Evolves the model according to the configuration set by `set_rgeConfig(RgeConfig)`. It returns false if the ODE solver runs into numerical problems, e.g. encounters a Landau pole. This does not usually happen if `perturbativity=true` in the `RgeConfig` since the RG running is stopped before the parameters become too large.

`evolve_to` first sets `_rgeConfig.finalEnergyScale` to the given argument and then evolves the model.

The result of the RG evolution is collected in a `RgeResults` struct. It can be retrieved with `get_rgeResults()` or simply printed to the console with `print_rgeResults()`. It stores any violation of perturbativity, unitarity or stability and at what energy scale it occurs.

| Function                | Description                        | Returns       |
|-------------------------|------------------------------------|---------------|
| save_current_state()    | Saves the current state internally | void          |
| reset_to_saved_state()  | Resets to a previously saved state | true/false   |

Since the `evolve()` function updates all the parameters, it can be useful to save the state of a model at a specific renormalization scale. The model can afterwards be restored to this state with `reset_to_saved_state()`.

Some other useful functions are:

| Function                      | Description                          | Returns       |
|-------------------------------|--------------------------------------|---------------|
| set_final_energy_scale(double) | Sets `finalEnergyScale` for RG evolution | void          |
| get_rgeResults(RgeResults)    | Retrieves results of RG evolution    | RgeResults    |
| print_rgeResults(RgeResults)  | Prints results of RG evolution to console | void          |
| set_renormalization_scale()   | Sets $\mu$                            | void          |
| get_renormalization_scale()   | Retrieves $\mu$                       | double        |

### 4.3 SM class

A class that describes the SM. It inherits the RGE functionality from `RgeModel`. The physics member variables\(^6\) that are evolved during RG evolution are

---

\(^6\)All member variables of classes are denoted with an underline as a first character.
- The three $SU(3)_c$, $SU(2)_L$ and $U(1)_Y$ gauge couplings: $g_1$, $g_2$, $g_3$.
- Quartic Higgs couplings $\lambda$: $\lambda$.
- Higgs VEV $v$: $v$.
- Complex 3 by 3 Yukawa matrices, $y_U$, $y_D$ and $y_L$.

In total these sum up to 59 real parameters. The Yukawa matrices are in the fermion weak eigenbasis and initialized with the CKM matrix and fermion masses:

$$Y^U = \frac{\sqrt{2}}{v} M_u,$$

$$Y^D = \frac{\sqrt{2}}{v} V_{CKM} M_d,$$

$$Y^L = \frac{\sqrt{2}}{v} M_\ell, \quad (1)$$

where $M_f$ are the diagonal fermion mass matrices. These parameters are at construction defined at the renormalization scale $\mu = 173.34$ GeV. For numerical values and conventions for the CKM matrix, see appendix C. There is no mechanism to generate neutrino masses implemented; hence the neutrinos are treated as being massless.

One should use the functions `evolve()` or `evolve_to(double)` of `RgeModel` to evolve the SM. In the RG evolution, the mass matrices $M_f$ and CKM matrix are calculated at each step by diagonalizing the $Y^F$ matrices. Note though that the RGEs for the SM are the full ones, with 6 quarks for example, and no decoupling is performed that should be done when running between energy scales that are below the top mass scale.

**Functionality**

The SM can be saved to a SLHA-like text file with

| Function                  | Description                     | Returns   |
|---------------------------|---------------------------------|-----------|
| `write_slha_file(string)` | Creates SLHA file.              | void      |
| `set_from_slha_file(string)` | Sets the SM from SLHA file. | true/false |

Other useful functions are

| Function            | Description                      | Returns   |
|---------------------|----------------------------------|-----------|
| `get_v2()`          | Returns $v^2$                    | double    |
| `get_gauge_couplings()` | Returns $\{g_1, g_2, g_3\}$       | vector<double> |
| `get_mup()`         | Returns $\{m_u, m_c, m_t\}$     | vector<double> |
| `get_mdn()`         | Returns $\{m_d, m_s, m_b\}$     | vector<double> |
| `get_ml()`          | Returns $\{m_e, m_\mu, m_\tau\}$ | vector<double> |
| `get_vCkm()`        | Returns CKM matrix               | Matrix3cd |
| `get_lambda()`      | Returns Higgs quartic coupling   | double    |
| `print_all()`       | Prints parameters to console     | void      |
Changing renormalization scale

As previously mentioned, the SM is constructed at the top mass scale. It is however possible to obtain a SM defined at any other energy scale. For example, to get the SM at 1 TeV, one can evolve a constructed SM with `evolve_to(1e3)`.

If one wants to manually change all the parameters of the SM, the easiest solution is to do the following:

- Construct a SM and save the parameters to a SLHA file with `write_slha_file("smSlhaFile")`.
- The SLHA file contain all the parameters of the SM, including the renormalization scale. It is readable and thus provides an easy way to manually edit all the numerical values.
- Then use the edited SLHA file to set a SM object with `set_from_slha_file("smSlhaFile")`.

5 THDM class

THDM is the main class of 2HDM and describes a general, potentially complex, 2HDM. It inherits RGE functionality from RgeModel.

Here, we give a short summary of the general 2HDM and the parameterization of it inside THDM. For a thorough review of the 2HDM see ref. [1]. We use the notation of refs. [14, 15, 16] to describe the generic basis of the 2HDM.

5.1 Parameters of 2HDM

The 2HDM contains two hypercharge +1 complex scalar SU(2) doublets, \( \Phi_1 \) and \( \Phi_2 \). First of all, since the scalar fields have identical quantum numbers, one can always perform a field redefinition of the scalar fields, i.e. a non-singular complex transformation \( \Phi_a \rightarrow B_{ab} \Phi_b \). The Lagrangian of the 2HDM exhibits a \( U(2) \) Higgs flavor symmetry, \( \Phi_a \rightarrow U_{ab} \Phi_b \); since the Lagrangian keeps the same form after such a transformation. We will denote 2HDMs related by such Higgs flavor transformations as different bases of the 2HDM.

5.2 Generic basis

The most general 2HDM gauge invariant renormalizable scalar potential can be written

\[
-L_V = m_{11}^2 \Phi_1^\dagger \Phi_1 + m_{22}^2 \Phi_2^\dagger \Phi_2 - (m_{12}^2 \Phi_1^\dagger \Phi_2 + \text{h.c.}) + \frac{1}{2} \lambda_1 (\Phi_1^\dagger \Phi_1)^2 + \frac{1}{2} \lambda_2 (\Phi_2^\dagger \Phi_2)^2 \\
+ \lambda_3 (\Phi_1^\dagger \Phi_1) (\Phi_2^\dagger \Phi_2) + \lambda_4 (\Phi_1^\dagger \Phi_2) (\Phi_2^\dagger \Phi_1) \\
+ \left[ \frac{1}{2} \lambda_5 (\Phi_1^\dagger \Phi_2)^2 + \lambda_6 (\Phi_1^\dagger \Phi_1) (\Phi_1^\dagger \Phi_2) + \lambda_7 (\Phi_2^\dagger \Phi_2) (\Phi_1^\dagger \Phi_1) + \text{h.c.} \right],
\]

where \( m_{12}^2 \), \( \lambda_5 \), \( \lambda_6 \) and \( \lambda_7 \) are potentially complex while all the other parameters are real; resulting in a total of 14 degrees of freedom. However, three of these are fixed by the tadpole equations and one can be removed by a re-phasing of the second Higgs doublet. The bases in eq. (2) will be referred to as the generic basis; which is the internal basis used in the THDM class.
After electroweak symmetry breaking, $SU(2) \times U(1)_Y \rightarrow U(1)_{em}$, both the scalar fields acquire VEVs, which can be expressed in terms of a unit vector in the Higgs flavor space

$$\langle \Phi_a \rangle = \frac{v}{\sqrt{2}} \begin{pmatrix} 0 & 0 \\ \hat{v}_a & \hat{v}_a \end{pmatrix}, \quad \hat{v}_a \equiv \begin{pmatrix} c_\beta \\ s_\beta e^{i\xi} \end{pmatrix},$$

(3)

where the unit vector is normalized to $\hat{v}_a^* \hat{v}_a = 1$. By convention, we take $0 \leq \beta \leq \pi/2$ and $0 \leq \xi \leq 2\pi$. Here, we have used up all our gauge freedom, when setting the VEV in the lower component of the doublets with a $SU(2)$ transformation and removing any phase in the $\Phi_1$ VEV with a $U(1)_Y$ transformation. We also define $\hat{w}_b \equiv \hat{v}_a^* \epsilon_{ab}$, where $\epsilon_{12} = -\epsilon_{21} = 1$ and $\epsilon_{11} = \epsilon_{22} = 0$.

The angle $\beta$ can be defined in terms of the ratio of the Higgs fields,

$$\tan \beta \equiv |\langle \Phi_2 \rangle|/|\langle \Phi_1 \rangle|.$$  

(4)

The Yukawa interactions in the generic basis are

$$-\mathcal{L}_Y = \tilde{Q}_L^0 \Phi \eta \eta^*_a U^0_R + \tilde{Q}_L^0 \Phi \eta \eta^*_a D^0_R + \tilde{L}_L^0 \Phi \eta \eta^*_a L^0_R E^0_R + \text{h.c.},$$

(5)

where the left-handed fermion fields in the weak eigenbasis are

$$Q_L^0 \equiv \begin{pmatrix} U^0_R \\ D^0_R \end{pmatrix}, \quad L_L^0 \equiv \begin{pmatrix} \nu^0_L \\ E^0_L \end{pmatrix}$$

(6)

and $\Phi = i\sigma_2 \Phi^*$. The 129 parameters of the 2HDM are stored as member variables in THDM:

- The $SU(3)_c \times SU(2)_W \times U(1)_Y$ gauge couplings: $g_3$, $g_2$ and $g_1$ respectively.
- The Higgs VEV $v^2 = v_1^2 + v_2^2$. This is initialized when feeding a SM to the THDM.
- The potential parameters in the generic basis: base_generic; which also includes the angles $\beta$ and $\xi$. See appendix B for a detailed description.
- The 6 Yukawa matrices in the fermion weak eigenbasis: _eta1U, _eta2U, _eta1D, _eta2D, _eta1l, _eta2l.

These are the variables that have their RGEs defined in RGE.cpp.

In addition to the member variables above, the THDM also stores the Yukawa sector in the fermion mass eigenbasis. To go to the fermion mass eigenbasis, the THDM first calculates the Yukawa matrices in the Higgs basis; which has the Lagrangian

$$-\mathcal{L}_Y = \tilde{Q}_L^0 \hat{H}_1 \kappa^{U,0} U^0_R + \tilde{Q}_L^0 \hat{H}_1 \kappa^{D,0} D^0_R + \tilde{L}_L^0 \hat{H}_1 \kappa^{L,0} E^0_R$$

$$+ \tilde{Q}_L^0 \hat{H}_2 \rho^{U,0} U^0_R + \tilde{Q}_L^0 \hat{H}_2 \rho^{D,0} D^0_R + \tilde{L}_L^0 \hat{H}_2 \rho^{L,0} E^0_R + \text{h.c.},$$

(7)

where only $H_1$ acquires a VEV. The $\kappa^{F,0}$ and $\rho^{F,0}$ matrices are given by

$$\kappa^{U,0} = \hat{v}_a^* \eta_a U^0_R, \quad \rho^{U,0} = \hat{w}_a \eta_a U^0_R,$$

$$\kappa^{D,0} = \hat{v}_a \eta_a D^0_R, \quad \rho^{D,0} = \hat{w}_a \eta_a D^0_R,$$

$$\kappa^{L,0} = \hat{v}_a \eta_a L^0_R, \quad \rho^{L,0} = \hat{w}_a \eta_a L^0_R.$$  

(8)
Note that $\hat{v}_a$ and $\hat{w}$ are defined in terms of the VEVs in the generic basis, which run during RG evolution; thus the transformation to the Higgs basis is $\mu$-dependent.

After going to the Higgs basis, \texttt{THDM} performs biunitary transformations to diagonalize the $\kappa^{F,0}$ matrices. The fermions in the mass eigenbasis are defined as

\begin{equation}
F_L \equiv V_L^F F^0_L, \quad F_R \equiv V_R^F F^0_R,
\end{equation}

where $F \in \{U, D, E\}$ is each fermion species. The the diagonal Yukawa matrices are

\begin{align}
\kappa^U &= V_L^U \kappa^{U,0} V_R^{U\dagger} = \frac{\sqrt{2}}{v} \text{diag}(m_u, m_c, m_t), \\
\kappa^D &= V_L^D \kappa^{D,0} V_R^{D\dagger} = \frac{\sqrt{2}}{v} \text{diag}(m_d, m_s, m_b), \\
\kappa^L &= V_L^L \kappa^{L,0} V_R^{L\dagger} = \frac{\sqrt{2}}{v} \text{diag}(m_e, m_\mu, m_\tau),
\end{align}

while $\rho^F = V_L^F \rho^{F,0} V_R^{F\dagger}$ are potentially non-diagonal; which would mean that tree-level FCNCs are present. The CKM matrix is composed out of the left-handed transformation matrices, $V_{CKM} = V_L^U V_L^{D\dagger}$.

To summarize, in addition to the parameters in the generic basis, the \texttt{THDM} stores

- The diagonal $\kappa^F$ matrices: $\_kU$, $\_kD$ and $\_kL$. The fermion masses are also stored as $\_mup[3]$, $\_mdn[3]$ and $\_ml[3]$.
- The non-diagonal $\rho^F$ matrices: $\_rU$, $\_rD$ and $\_rL$.
- The CKM matrix: $\_vCKM$

### 5.3 How to use \texttt{THDM}

To fully initialize the \texttt{THDM}, one needs to

- Construct a \texttt{THDM} object and feed it a \texttt{SM} object.
- Set the potential with any of the available bases.
- Fix the Yukawa sector. This can be done with a $\mathbb{Z}_2$ symmetry or a flavor ansatz, which produces a Yukawa sector without FCNCs. However, the Yukawa matrices can also be set manually.

The \texttt{SM} can be given at construction or with \texttt{set.sm(SM)}. This will set the VEV, gauge couplings, renormalization scale, $\kappa^F$ matrices and CKM matrix.

### 5.4 Setting the scalar potential

The scalar potential can be set with any of the bases described in section B. Internally though, the \texttt{THDM} uses the generic basis. The functions are
Returns

| set_param_gen(Base_generic,bool=true) | true/false |
| set_param_higgs(Base_higgs,bool=true) | true/false |
| set_param_invariant(Base_invariant,bool=true) | true/false |
| set_param_hybrid(Base_hybrid,bool=true) | true/false |

All of the functions set the parameters in the generic basis after transforming the basis that is given. The optional bool argument refers to if the tree-level tadpole equations should be enforced; which they are by default. If $\tan \beta \neq 0$, the eqs.(A4,A5,A7) of ref. [14] are used, which fix $m_{11}^2$, $m_{22}^2$ and $\xi$. Otherwise the Higgs basis tadpole equations are used, which fix $m_{11}^2 = -v^2 \lambda_1/2$ and $m_{12}^2 = -v^2 \lambda_6/2$. These functions will return false if the tree-level Higgs masses are imaginary or if the tadpole equations could not be set.

5.5 Setting the Yukawa sector

Note that $\tan \beta$ must be set before fixing the Yukawa sector. After that has been done, it can initialized with

```
set_yukawa_type(Z2symmetry)
```

| set_yukawa_type(Z2symmetry) | Description |
|-----------------------------|-------------|
| set_yukawa_aligned(double,double,double) | Fixes all Yukawa matrices from a $Z_2$ symmetry. |
| set_yukawa_manual(Matrix3cd, ...) | Sets the Yukawa matrices from a flavor ansatz. |
| set_yukawa_eta(Matrix3cd, ...) | Sets all $\eta^{F,0}$ manually. |

The $Z_2$ is specified by a enum, Z2symmetry, which can be set to either NO_SYM, TYPE_I, TYPE_II, TYPE_III or TYPE_IV. Imposing a $Z_2$ makes the Yukawa matrices proportional to each other,

$$\rho^F = a^F \kappa^F,$$

(11)

where the coefficients $a^F$ are fixed by $\beta$ as in table 1. These $a^F$ coefficients can also be set manually with set_yukawa_aligned(aU,aD,aL).

| Type | $U_R$ | $D_R$ | $L_R$ | $a^U$ | $a^D$ | $a^L$ |
|------|-------|-------|-------|-------|-------|-------|
| I    | +     | +     | +     | cot $\beta$ | cot $\beta$ | cot $\beta$ |
| II   | +     | -     | -     | cot $\beta$ | $-\tan \beta$ | $-\tan \beta$ |
| Y    | +     | -     | +     | cot $\beta$ | $-\tan \beta$ | cot $\beta$ |
| X    | +     | +     | -     | cot $\beta$ | cot $\beta$ | $-\tan \beta$ |

Table 1: Different $Z_2$ symmetries that can be imposed on the 2HDM. $\Phi_1$ is odd(−1) and $\Phi_2$ is even(+1). For every type of $Z_2$ symmetry, the $\rho^F$ matrices become proportional to the diagonal mass matrices, $\rho^F = a^F \kappa^F$.

---

7Using set_yukawa_type(NO_SYM) does nothing in terms of fixing the $\rho^F$ matrices.
5.6 Checks

Some of the checks that are implemented are:

| Function                             | Returns          |
|--------------------------------------|------------------|
| is_perturbative()                    | true/false       |
| is_unitary()                         | true/false       |
| is_stable()                          | true/false       |
| is_cp_conserved()                    | true/false       |

The perturbativity limit is reached when any of the $\lambda_i$ parameters is larger than $4\pi$. The 2HDM is unitary if the constraints laid out in appendix D are satisfied. Likewise, the stability conditions are given in appendix E.

5.7 Miscellaneous functions

There are a number of functions that returns useful quantities from the THDM:

| Function                                                         | Returns                                                                 |
|-----------------------------------------------------------------|-------------------------------------------------------------------------|
| get_param_gen()                                                  | Base_generic                                                            |
| get_param_higgs()                                                | Base_higgs                                                              |
| get_param_invariant()                                            | Base_invariant                                                          |
| get_param_hybrid()                                               | Base_hybrid                                                             |
| get_yukawa_type()                                               | Z2symmetry                                                              |
| get_aF()                                                        | $\{|a_U^F|, |a_D^F|, |a_L^F|\}$                                            |
| get_v2()                                                        | $v^2$                                                                   |
| get_gauge_couplings()                                           | $\{g_1, g_2, g_3\}$                                                   |
| get_mup()                                                       | $\{m_u, m_c, m_t\}$                                                   |
| get_ml()                                                        | $\{m_e, m_\mu, m_\tau\}$                                              |
| get_vCKM()                                                      | $V_{CKM}$                                                              |
| get_yukawa_eta()                                                | all $\{\eta^F_i\}$                                                   |
| get_vevs()                                                      | $\{v \cos \beta, v \sin \beta e^{i\xi}\}$                            |
| get_higgs_treeLvl_masses()                                       | $\{m_h, m_{H^\pm}\}$                                                 |
| get_largest_diagonal_rF()                                        | $\max(\rho^F_i)$                                                      |
| get_largest_nonDiagonal_rF()                                     | $\max(\rho^F_{i\neq j})$                                             |
| get_largest_lambda()                                            | $\max(\lambda^F_i)$                                                   |
| get_largest_nonDiagonal_lamF()                                   | $\max(\lambda^F_{i\neq j})$                                          |
| get_lamF_element(FermionSector, i, j)                           | $\lambda^F_{ij}$                                                      |
| get_lamF(FermionSector)                                          | $\lambda^F_i$                                                         |

Note that the aligned parameters $a^F$ are only meaningful when the Yukawa matrices are diagonal, which may change during RG evolution.

The get_largest_nonDiagonal_lamF() and get_lamF_element(FermionSector, i, j) functions returns the $\rho^F$ Yukawa matrices in terms of the Cheng-Sher ansatz defined by $\lambda^F_{ij} \equiv v \sqrt{\frac{\rho^F_{ij}}{2m_i m_j}}$, where FermionSector is either UP, DOWN or LEPTON.

There are also a bunch of functions that print information to the console:
If SPheno is enabled, one can run

```
Function | Returns
---|---
run_spheno(int) | true/false
```

The argument is the loop order used when calculating the loop corrected masses; 0 = Tree-level, 1 = 1-loop for all masses, 2 = Include 2-loop corrections to the neutral Higgs. It first creates a temporary SLHA file, which is then given to SPheno; it returns false if it fails these tasks. The results are stored in THDM and some of them can be printed to the console with

```
Function | Description
---|---
print_spheno_results() | Prints SPheno results to console.
```

If one wants to calculate the results of performing a RG evolution of the 2HDM without updating its parameters, one can use

```
Function | Description | Returns
---|---|---
calc_rgeResults() | Calculates RG evolution | void
```

The results can be printed to the console with the RgeModel’s function print_rgeResults.

One can create SLHA-like files and setting THDM objects with

```
Function | Description | Returns
---|---|---
write_slha_file(int sphenoLoopLvl, string file) | Creates SLHA file | void
set_from_slha_file(string file) | Sets THDM from SLHA file | true/false
```

These files contain all the information of the THDM and can be used as an input file to SPheno. The sphenoLoopLvl sets the loop order configuration for SPheno.

6 RG evolution summary

Here, we briefly describe the procedure that is used when evolving a 2HDM. As an example, we initialize the 2HDM at the top mass scale and evolve upwards in energy\(^8\).

First one must initialize the THDM with a SM. Then one must set the scalar potential and Yukawa sector\(^9\). The options for the RG evolution are specified by a RgeConfig, which is

\(^8\)Note that evolving a THDM downwards in energy is also possible. One must then, however, fix the high scale boundary condition first in some way.

\(^9\)Alternatively, one can set the THDM from a SLHA file instead of performing the first few steps.
given to the THDM with `set_rgeConfig`. After that, the THDM can be evolved in energy with `evolve()`. During the RG evolution, the following is happening:

- At each intermediate step as specified in the `RgeConfig`, perturbativity, unitarity and stability are checked.
- The parameters are evolved in the generic basis, but the other bases are calculated with the $\mu$-dependent $\tan \beta$ and $\xi$ at each step.
- If `dataOutput=true`, the THDM stores the parameters as a function of $\mu$ in text files in `output/"evolutionName"`. If `GNUPLOT` is enabled, it also creates simple afterwards.
- When the RG evolution stops depends on the settings in `RgeConfig`. By default, it stops when perturbativity is violated; which is very close to a Landau pole.

**Acknowledgments**

2HDME originated from an extension of 2HDMC [6] in collaboration with Johan Rathsman, who also provided useful feedback on this manuscript.

This work is supported in part by the Swedish Research Council grants contract numbers 621-2013-4287 and 2016-05996 and by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 668679).
A Installation instructions

Source code
The source code is available at https://github.com/jojelen/2HDME.

Dependencies
The 2HDME requires the following to be installed:

- A C++11 compiler such as gcc.
- 2HDME requires the library Eigen[5] to perform linear algebra operations. If you have installed it, but the compiler does not find it, you can add the path to its directory with CFLAGS+=-I/.../eigen3 in the Makefile.
- To solve the RGEs, 2HDME uses the GNU GSL library[4], which is usually included in GNU/Linux distributions. See https://www.gnu.org/software/gsl/ for details.

Additional dependencies
These dependencies are optional and can be enabled/disabled by commenting the relevant lines in the Makefile:

- The 2HDME can automatically create simple plots of the RG running of the parameters with the help of GNUPLOT, see http://www.gnuplot.info/ for details.
- There is the possibility to link 2HDME with SPheno [spheno.hepforge.org]. See the instructions below of how to install it.

Compilation
First, make sure that all the requirements are properly installed. One might need to configure Makefile to link all the libraries if they are not installed in the usual location. After that, one can proceed to compile: In terminal, type

```
make
```

in the main directory that contains the Makefile. Please note that the RGEs in RGE.cpp are not written in an optimal form. However, the compiler with optimization level -03 will take care of this. This takes some time and compiling RGE.cpp takes roughly 10 min on a laptop.

Run Demo
To see that everything works, the demo in section 3 is included. The demo evolves a $Z_2$ symmetric CP conserved 2HDM from the top mass scale to the Planck scale. The source code is located in src/demos/DemoRGE.cpp. To run it, type

```
bin/DemoRGE
```

in the terminal. If the GNUPLOT functionality hasn’t been disabled (by commenting out lines in the Makefile), plots of the parameters should have been created in output/DemoRGE/plots.
**SPheno linkage**

To enable SPheno, one must have a working installation of SPheno [spheno.hepforge.org] in the folder of 2HDME. The folder's name is by default SPheno-4.0.3, but can be changed by modifying `SPheno_PATH` in `src/SPheno.cpp`.

Next up, the model files for 2HDM must be compiled with your SPheno. The needed model files have been created with SARAH [sarah.hepforge.org] and are included in `SPhenoModelFiles/THDM_GEN`. Copy `THDM_GEN` to the SPheno directory and run `make` to compile it.

When SPheno and the 2HDM model files have been compiled, the SPheno functionality in 2HDME can be enabled by removing the relevant comments in the Makefile.

See DemoRGE.cpp for an example of how to compute loop corrected masses with SPheno.

Please cite the relevant sources for SPheno and SARAH if you use this functionality of 2HDME.

**B Bases of 2HDM’s potential**

There are four bases for the most general scalar potential of the 2HDM implemented in 2HDME as well as one basis that describes the CP conserved 2HDM with a softly broken $Z_2$ symmetry. The base struct for these bases is `ThdmBasis`, which has the member variables that define the VEV in eq. (3), i.e. $\beta$ and $\xi$.

More details about these bases and their relations to each other can be found in refs. [14, 15]. Note that THDM works in the generic basis internally.

**Base generic**

The generic basis of 2HDM is described in section 5.2. It consists of the additional parameters:

| Parameter $m_{ij}^2$, $M_{ij}$, $\lambda_i$ | Base generic |
|---------------------------------------------------|--------------|
| $m_{11}^2$                                        | $M_{112}$    |
| $m_{22}^2$                                        | $M_{222}$    |
| $m_{12}^2$                                        | $M_{12}$     |
| $\lambda_1$                                       | Lambda1      |
| $\lambda_2$                                       | Lambda2      |
| $\lambda_3$                                       | Lambda3      |
| $\lambda_4$                                       | Lambda4      |
| $\lambda_5$                                       | Lambda5      |
| $\lambda_6$                                       | Lambda6      |
| $\lambda_7$                                       | Lambda7      |

A generic basis can be converted to other bases with the functions:

| Function                        | Returns          |
|---------------------------------|------------------|
| convert_to_compact()            | Base_compact     |
| convert_to_higgs()              | Base_higgs       |
| convert_to_invariant(double v2) | Base_invariant   |
The compact basis is defined by
\[ -\mathcal{L}_V = Y_{ab} \Phi_a^\dagger \Phi_b + \frac{1}{2} Z_{abcd} (\Phi_a^\dagger \Phi_b)(\Phi_c^\dagger \Phi_d). \]  \hfill (12)

All these parameters are stored as complex numbers in **Base_compact**:

| Parameter | **Base_compact** |
|-----------|------------------|
| \( Y_{ab} \) | Y[2] [2] |
| \( Z_{abcd} \) | Z[2] [2] [2] [2] |

The Higgs basis is defined by the basis where only one scalar doublet acquires a VEV,
\[ \langle H_1 \rangle = \frac{v}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \langle H_2 \rangle = 0. \]  \hfill (13)

The Lagrangian takes the form
\[ -\mathcal{L}_V = Y_1 H_1^\dagger H_1 + Y_2 H_2^\dagger H_2 + \left( Y_3 H_1^\dagger H_2 + \text{h.c.} \right) + \frac{1}{2} Z_1 (H_1^\dagger H_1)^2 + \frac{1}{2} Z_2 (H_2^\dagger H_2)^2 + \frac{1}{2} Z_3 (H_1^\dagger H_1)(H_2^\dagger H_2) + \frac{1}{2} Z_4 (H_1^\dagger H_2)(H_2^\dagger H_1) \]
\[ + \left\{ \frac{1}{2} Z_5 (H_1^\dagger H_2)^2 + \left[ Z_6 (H_1^\dagger H_1) + Z_7 (H_2^\dagger H_2) \right] H_1^\dagger H_2 + \text{h.c.} \right\}. \]  \hfill (14)

The \( Y_1, Y_2, Z_{1,2,3,4} \) are invariants under a Higgs flavor \( U(2) \) transformation, while \( Y_3, Z_{5,6,7} \) are pseudoinvariants that transform as
\[ \{ Y_3, Z_6, Z_7 \} \rightarrow (\det U)^{-1} \{ Y_3, Z_6, Z_7 \}, \]  \hfill (15)
\[ Z_5 \rightarrow (\det U)^{-2} Z_5. \]  \hfill (16)

This effectively means that the Higgs basis is unique up to a rephasing of the \( H_2 \) field. The parameters of **Base_higgs** are:

| Parameter | **Base_higgs** |
|-----------|----------------|
| \( Y_1 \) | Y1 |
| \( Y_2 \) | Y2 |
| \( Y_3 \) | Y3 |
| \( Z_1 \) | Z1 |
| \( Z_2 \) | Z2 |
| \( Z_3 \) | Z3 |
| \( Z_4 \) | Z4 |
| \( Z_5 \) | Z5 |
| \( Z_6 \) | Z6 |
| \( Z_7 \) | Z7 |
And it can be transformed to other bases with the functions:

| Function                                              | Returns     |
|-------------------------------------------------------|-------------|
| convert_to_generic()                                  | Base_generic|
| convert_to_compact()                                  | Base_compact|
| convert_toInvariant(double v2)                        | Base_invariant|

**Base invariant**

The invariant basis describes the general 2HDM with only Higgs flavor $U(2)$ invariant quantities.

Four invariant quantities are the tree-level masses of the Higgs bosons. The charged Higgs boson mass is given by

$$m_{H^\pm}^2 = Y_2 + \frac{1}{2}Z_3v^2$$

and the three neutral Higgs bosons’ masses are given by the mass matrix in the Higgs basis,

$$\mathcal{M} \equiv v^2 \begin{pmatrix} Z_1 & \text{Re}(Z_6) & -\text{Im}(Z_6) \\ \text{Re}(Z_6) & \frac{1}{2} [Z_3 + Z_4 + \text{Re}(Z_5)] + Y_2/v^2 & -\frac{1}{2} \text{Im}(Z_5) \\ -\text{Im}(Z_6) & -\frac{1}{2} \text{Im}(Z_5) & \frac{1}{2} [Z_3 + Z_4 - \text{Re}(Z_5)] + Y_2/v^2 \end{pmatrix}.$$  

(18)

This mass matrix can be diagonalized with the rotation matrix

$$R \equiv \begin{pmatrix} c_{13}c_{12} & -c_{23}s_{12} & -c_{12}s_{13}s_{23} \\ c_{13}s_{12} & c_{12}c_{23} & -s_{12}s_{13}s_{23} \\ s_{13} & s_{13} & c_{13}c_{23} \end{pmatrix},$$  

(19)

to produce $\mathcal{M}_D = R\mathcal{M}R^T = \text{diag}(m_{h_1}^2, m_{h_2}^2, m_{h_3}^2)$. We will, without loss of generality, assume ordered masses, $m_{h_1} < m_{h_2} < m_{h_3}$. The eigenvalues of the mass matrix are invariant under Higgs flavor transformations, even though the matrix elements are not. Consequently, the rotation matrix is not invariant either. While the angles $\theta_{12}$ and $\theta_{13}$ are invariant, $\theta_{23}$ changes so that $e^{i\theta_{23}} \rightarrow (\det U)e^{i\theta_{23}}$ is a pseudo-invariant quantity [15]. In ref. [15], they define a $U(2)$ invariant mass matrix,

$$\tilde{\mathcal{M}} \equiv v^2 \begin{pmatrix} Z_1 & \text{Re}(Z_6e^{-i\theta_{23}}) & -\text{Im}(Z_6e^{-i\theta_{23}}) \\ \text{Re}(Z_6e^{-i\theta_{23}}) & \text{Re}(Z_5e^{-2i\theta_{23}}) + A^2/v^2 & -\frac{1}{2} \text{Im}(Z_5e^{-2i\theta_{23}}) \\ -\text{Im}(Z_6e^{-i\theta_{23}}) & -\frac{1}{2} \text{Im}(Z_5e^{-2i\theta_{23}}) & A^2/v^2 \end{pmatrix},$$  

(20)

where $A^2 = Y_2 + \frac{1}{2} [Z_3 + Z_4 - \text{Re}(Z_5e^{-2i\theta_{23}})]v^2$. This matrix is diagonalized with the rotation matrix

$$\tilde{R} \equiv \begin{pmatrix} c_{12}c_{13} & -s_{12} & -c_{12}s_{13} \\ c_{13}s_{12} & c_{12} & -s_{12}s_{13} \\ s_{13} & 0 & c_{13} \end{pmatrix},$$  

(21)

such that $\mathcal{M}_D = \tilde{R}\tilde{\mathcal{M}}\tilde{R}^T$. The angles lie in the range $-\pi/2 \leq \theta_{12}, \theta_{13} < \pi/2$ in general.

Now, we can construct a $U(2)$ invariant basis with the 8 real parameters $\{m_{h_i}\}, m_{H^\pm}, \theta_{12}, \theta_{13}, Z_2, Z_3$ and the complex parameter $Z_7e^{-i\theta_{23}}$. 

19
| Parameter             | Description                              | Base_invariant |
|-----------------------|------------------------------------------|----------------|
| \( \{m_{h_1}, m_{h_2}, m_{h_3}\} \) | Ordered neutral Higgs boson masses      | \( mh[3]\) |
| \( m_{H^\pm} \)       | Charged Higgs boson mass                 | \( m_Hc \)   |
| \( s_{12} \in [-1,1] \) | Mixing angle of neutral Higgs mass matrix| \( s_{12} \) |
| \( c_{13} \in [0,1] \) | Mixing angle of neutral Higgs mass matrix| \( c_{13} \) |
| \( \{Z_2, Z_3\} \)    | Real quartic couplings                   | \( Z_{2,3} \) |
| \( Z_7 e^{-i\theta_{23}} \) | Complex quartic coupling                 | \( Z_{7 \text{inv}} \) |

The invariant basis can be converted to other bases with

| Returns                                      | Base                                      |
|----------------------------------------------|-------------------------------------------|
| \textproc{convert\_to\_generic(double v2)}  | Base\_generic                            |
| \textproc{convert\_to\_compact(double v2)} | Base\_compact                            |
| \textproc{convert\_to\_higgs(double v2)}  | Base\_higgs                              |

**Base\_hybrid**

The hybrid basis presented in ref. [17] is describing a \( CP \) conserved 2HDM with a softly broken \( Z_2 \) symmetry. It consists of a combination of tree-lvl masses and quartic couplings:

| Parameter      | Description                              | Base_invariant |
|----------------|------------------------------------------|----------------|
| \( m_h \)     | Lightest \( CP \) even Higgs boson       | \( mh \)       |
| \( m_H \)     | Heaviest \( CP \) even Higgs boson       | \( m_H \)      |
| \( \cos(\beta - \alpha) \) | Mixing angle of \( CP \) even Higgs mass matrix | \( cba \) |
| \( \tan \beta \) | Ratio of Higgs VEVs in generic basis  | \( \tan b \) |
| \( \{Z_4, Z_5, Z_7\} \) | Real quartic couplings               | \( Z_{4,5,7} \) |

and can be converted to the general bases with

| Returns                                      | Base                                      |
|----------------------------------------------|-------------------------------------------|
| \textproc{convert\_to\_generic(double v2)}  | Base\_generic                            |
| \textproc{convert\_to\_higgs(double v2)}  | Base\_higgs                              |
| \textproc{convert\_to\_invariant(double v2)} | Base\_invariant                          |

**C \ SM input**

The SM is defined at the top quark mass scale, \( M_t = 173.34 \text{ GeV} \) [18]. See section 4.3 for instructions of how to create a SM object at another renormalization scale. At construction, we use the following input to fix its parameters:

- The SM Higgs VEV is taken to be \( v = (\sqrt{2}G_F)^{-1/2} = 246.21971 \text{ GeV} \) [19].
- The fermion masses are used to fix the Yukawa matrix elements in the fermion mass eigenbasis. We use the ones from ref. [20]:

\[
\begin{align*}
    m_u &= 1.22 \text{ MeV}, & m_c &= 0.590 \text{ GeV}, & m_t &= 162.2 \text{ GeV}, \\
    m_d &= 2.76 \text{ MeV}, & m_s &= 52 \text{ MeV}, & m_b &= 2.75 \text{ GeV}, \\
    m_e &= 0.485289396 \text{ MeV}, & m_\mu &= 0.1024673155 \text{ GeV}, & m_\tau &= 1.74215 \text{ GeV}.
\end{align*}
\] (22)
• Gauge couplings from ref. [19]:

\[
g_1 = 0.3583 \\
g_2 = 0.64779 \\
g_3 = 1.1666
\]  
(23)

for \(U(1)_Y\), \(SU(2)_W\) and \(SU(3)_c\) respectively.

• For the CKM matrix, we use the standard parametrization

\[
V_{CKM} = \begin{pmatrix}
    c_{12}c_{13} & s_{12}c_{13} & s_{13}e^{-i\delta} \\
    -s_{12}c_{23} - c_{12}s_{23}s_{13}e^{i\delta} & c_{12}c_{23} - s_{12}s_{23}s_{13}e^{i\delta} & s_{23}c_{13} \\
    s_{12}s_{23} - c_{12}c_{23}s_{13}e^{i\delta} & -c_{12}s_{23} - s_{12}c_{23}s_{13}e^{i\delta} & c_{23}c_{13}
\end{pmatrix},
\]  
(24)

where the angles in terms of the Wolfenstein parameters are

\[
\begin{align*}
s_{12} &= \lambda, \\
s_{23} &= A\lambda^2, \\
s_{13}e^{i\delta} &= \frac{A\lambda^3(\bar{\rho} + i\bar{\eta})\sqrt{1 - A^2\lambda^4}}{\sqrt{1 - \lambda^2}[1 - A^2\lambda^4(\bar{\rho} + i\bar{\eta})]}.
\end{align*}
\]  
(25)

The numerical values

\[
\begin{align*}
\lambda &= 0.22453, \\
A &= 0.836, \\
\bar{\rho} &= 0.122, \\
\bar{\eta} &= 0.355,
\end{align*}
\]  
(26)

are extracted from the PDG [21].

### D Tree-level unitarity conditions

The tree-level unitarity conditions for a general 2HDM have been worked out in ref. [22]. There, they work out the following scattering matrices:

\[
\Lambda_{21} \equiv \begin{pmatrix}
    \lambda_1 & \lambda_5 & \sqrt{2}\lambda_6 \\
    \lambda_5^* & \lambda_2 & \sqrt{2}\lambda_7^* \\
    \sqrt{2}\lambda_6^* & \sqrt{2}\lambda_7 & \lambda_3 + \lambda_4
\end{pmatrix},
\]  
(27)

\[
\Lambda_{20} \equiv \lambda_3 - \lambda_4,
\]  
(28)

\[
\Lambda_{01} \equiv \begin{pmatrix}
    \lambda_1 & \lambda_4 & \lambda_6 & \lambda_6^* \\
    \lambda_4^* & \lambda_2 & \lambda_7 & \lambda_7^* \\
    \lambda_6^* & \lambda_7^* & \lambda_3 & \lambda_5 \\
    \lambda_6 & \lambda_7 & \lambda_5 & \lambda_3
\end{pmatrix},
\]  
(29)

\[
\Lambda_{00} \equiv \begin{pmatrix}
    3\lambda_1 & 2\lambda_3 + \lambda_4 & 3\lambda_6 & 3\lambda_6^* \\
    2\lambda_3 + \lambda_4^* & 3\lambda_2 & 3\lambda_7 & 3\lambda_7^* \\
    3\lambda_6^* & 3\lambda_7^* & \lambda_3 + 2\lambda_4 & 3\lambda_5^* \\
    3\lambda_6 & 3\lambda_7 & 3\lambda_5 & \lambda_3 + 2\lambda_4
\end{pmatrix}.
\]  
(30)
In the end, the unitarity constraint put upper limits on the absolute value of the eigenvalues, \( \Lambda_i \), of these matrices,
\[
|\Lambda_i| < 8\pi. \tag{31}
\]

## E Stability

Here, we give the conditions for the scalar potential to be bounded from below, as worked out in ref. [23, 24].

When working out these conditions, ref. [23, 24] constructed a Minkowskian formalism of the 2HDM that uses gauge-invariant field bilinears,
\[
\begin{align*}
    r^0 &\equiv \Phi_1^\dagger \Phi_1 + \Phi_2^\dagger \Phi_2, \\
    r^1 &\equiv 2 \text{Re} \left( \Phi_1^\dagger \Phi_2 \right), \\
    r^2 &\equiv 2 \text{Im} \left( \Phi_1^\dagger \Phi_2 \right), \\
    r^3 &\equiv \Phi_1^\dagger \Phi_1 - \Phi_2^\dagger \Phi_2.
\end{align*}
\]
These can be used to create a four-vector \( r^\mu = (r^0, \vec{r}) \); where one can raise and lower the indices as usual with the flat Minkowski metric \( \eta^\mu_\nu = \text{diag}(1, -1, -1, -1) \). In this formalism, the scalar potential is conveniently written as
\[
V = -M_\mu r^\mu + \frac{1}{2} r^\mu \Lambda^\nu_\mu r^\nu, \tag{36}
\]
where
\[
M_\mu = \left(-\frac{1}{2}(Y_1 + Y_2), \text{Re} (Y_3), -\text{Im} (Y_3), -\frac{1}{2}(Y_1 - Y_2)\right), \tag{37}
\]
and
\[
\Lambda^\nu_\mu = \frac{1}{2} \begin{pmatrix}
    \frac{1}{2}(Z_1 + Z_2) + Z_3 & -\text{Re} (Z_6 + Z_7) & \text{Im} (Z_6 + Z_7) & -\frac{1}{2}(Z_1 - Z_2) \\
    \text{Re} (Z_6 + Z_7) & -Z_4 - \text{Re} (Z_5) & \text{Im} (Z_5) & -\text{Re} (Z_6 - Z_7) \\
    -\text{Im} (Z_6 + Z_7) & \text{Im} (Z_5) & -Z_4 + \text{Re} (Z_5) & \text{Im} (Z_6 - Z_7) \\
    \frac{1}{2}(Z_1 - Z_2) & -\text{Re} (Z_6 - Z_7) & \text{Im} (Z_6 - Z_7) & -\frac{1}{2}(Z_1 + Z_2) + Z_3
\end{pmatrix}. \tag{38}
\]
The scalar potential is bounded from below if and only if all of the below requirements are fulfilled:

- All the eigenvalues of \( \Lambda^\nu_\mu \) are real.
- There exists a largest eigenvalue that is positive, \( \Lambda_0 > \{\Lambda_1, \Lambda_2, \Lambda_3\} \) and \( \Lambda_0 > 0 \).
- There exist four linearly independent eigenvectors; one \( V^{(a)} \) for each eigenvalue \( \Lambda_a \).
- The eigenvector to the largest eigenvalue is timelike, while the others are spacelike,
\[
\begin{align*}
    V^{(0)} \cdot V^{(0)} &= \left(V^{(0)}_0\right)^2 - \sum_{i=1}^3 \left(V^{(0)}_i\right)^2 > 0, \tag{39} \\
    V^{(i)} \cdot V^{(i)} &= \left(V^{(i)}_0\right)^2 - \sum_{j=1}^3 \left(V^{(i)}_j\right)^2 < 0. \tag{40}
\end{align*}
\]
References

[1] G. Branco, P. Ferreira, L. Lavoura, M. Rebelo, M. Sher, et. al., Theory and phenomenology of two-Higgs-doublet models, Phys.Rept. 516 (2012) 1–102, [arXiv:1106.0034].

[2] J. Oredsson, 2 Higgs Doublet Model Evolver, https://github.com/jojelen/2HDME.

[3] J. Oredsson and J. Rathsman, $Z_2$ breaking effects in 2-loop RG evolution of 2HDM, arXiv:1810.02588.

[4] M. G. et.al., GNU Scientific Library Reference Manual (3rd Ed.), ISBN 0954612078 [http://www.gnu.org/software/gsl/].

[5] G. Guennebaud, B. Jacob, et. al., “Eigen v3.” http://eigen.tuxfamily.org, 2010.

[6] D. Eriksson, J. Rathsman, and O. Stål, 2HDMC: Two-Higgs-Doublet Model Calculator Physics and Manual, Comput. Phys. Commun. 181 (2010) 189–205, [arXiv:0902.0851].

[7] M. E. Machacek and M. T. Vaughn, Two Loop Renormalization Group Equations in a General Quantum Field Theory. 1. Wave Function Renormalization, Nucl. Phys. B222 (1983) 83–103.

[8] M. E. Machacek and M. T. Vaughn, Two Loop Renormalization Group Equations in a General Quantum Field Theory. 2. Yukawa Couplings, Nucl. Phys. B236 (1984) 221–232.

[9] M. E. Machacek and M. T. Vaughn, Two Loop Renormalization Group Equations in a General Quantum Field Theory. 3. Scalar Quartic Couplings, Nucl. Phys. B249 (1985) 70–92.

[10] M.-x. Luo, H.-w. Wang, and Y. Xiao, Two loop renormalization group equations in general gauge field theories, Phys. Rev. D67 (2003) 065019, [hep-ph/0211440].

[11] A. V. Bednyakov, On Three-loop RGE for the Higgs Sector of 2HDM, arXiv:1809.04527.

[12] I. Schienbein, F. Staub, T. Steudtner, and K. Svirina, Revisiting RGEs for general gauge theories, arXiv:1809.06797.

[13] J. Bijnens, J. Oredsson, and J. Rathsman, Scalar Kinetic Mixing and the Renormalization Group, arXiv:1810.04483.

[14] S. Davidson and H. E. Haber, Basis-independent methods for the two-Higgs-doublet model, Phys. Rev. D72 (2005) 035004, [hep-ph/0504050]. [Erratum: Phys. Rev.D72,099902(2005)].

[15] H. E. Haber and D. O’Neil, Basis-independent methods for the two-Higgs-doublet model. II. The Significance of $\tan\beta$, Phys. Rev. D74 (2006) 015018, [hep-ph/0602242]. [Erratum: Phys. Rev.D74,no.5,059905(2006)].
[16] H. E. Haber and D. O’Neil, *Basis-independent methods for the two-Higgs-doublet model III: The CP-conserving limit, custodial symmetry, and the oblique parameters S, T, U*, Phys. Rev. D83 (2011) 055017, [arXiv:1011.6188].

[17] H. E. Haber and O. Stål, *New LHC benchmarks for the CP-conserving two-Higgs-doublet model*, Eur. Phys. J. C75 (2015), no. 10 491, [arXiv:1507.04281]. [Erratum: Eur. Phys. J.C76,no.6,312(2016)].

[18] ATLAS, CDF, CMS, D0 Collaboration, *First combination of Tevatron and LHC measurements of the top-quark mass*, arXiv:1403.4427.

[19] D. Buttazzo, G. Degrassi, P. P. Giardino, G. F. Giudice, F. Sala, A. Salvio, and A. Strumia, *Investigating the near-criticality of the Higgs boson*, JHEP 12 (2013) 089, [arXiv:1307.3536].

[20] Z.-z. Xing, H. Zhang, and S. Zhou, *Updated Values of Running Quark and Lepton Masses*, Phys. Rev. D77 (2008) 113016, [arXiv:0712.1419].

[21] Particle Data Group Collaboration, M. T. et al., *Review of Particle Physics*, Phys. Rev. D98 (2018) 030001.

[22] I. F. Ginzburg and I. P. Ivanov, *Tree-level unitarity constraints in the most general 2HDM*, Phys. Rev. D72 (2005) 115010, [hep-ph/0508020].

[23] I. P. Ivanov, *Minkowski space structure of the Higgs potential in 2HDM*, Phys. Rev. D75 (2007) 035001, [hep-ph/0609018]. [Erratum: Phys. Rev.D76,039902(2007)].

[24] I. P. Ivanov, *Minkowski space structure of the Higgs potential in 2HDM. II. Minima, symmetries, and topology*, Phys. Rev. D77 (2008) 015017, [arXiv:0710.3490].