DEUTSCHES ELEKTRONEN-SYNCHROTRON

Ein Forschungszentrum der Helmholtz-Gemeinschaft

HEPfit: A Code for the Combination of Indirect and Direct Constraints on High Energy Physics Models

J. de Blas et al.

NOTKESTRASSE 85 - 22607 HAMBURG

To be sure that your reports and preprints are promptly included in the
HEP literature database
send them to (if possible by air mail):

DESY	DESY
Zentralbibliothek	Bibliothek
Notkestraße 85	Platanenallee 6
22607 Hamburg	15738 Zeuthen
Germany	Germany

`HEPfit`: a Code for the Combination of Indirect and Direct Constraints on High Energy Physics Models

J. de Blas,[a,b] **D. Chowdhury,**[c,d] **M. Ciuchini,**[e] **A. M. Coutinho,**[f] **O. Eberhardt,**[g] **M. Fedele,**[h] **E. Franco,**[i] **G. Grilli di Cortona,**[j] **V. Miralles,**[g] **S. Mishima,**[k] **A. Paul,**[l,m] **A. Peñuelas,**[g] **M. Pierini,**[n] **L. Reina,**[o] **L. Silvestrini,**[i,p] **M. Valli,**[q] **R. Watanabe**[e] **and N. Yokozaki**[r]

[a]*Dipartimento di Fisica e Astronomia "Galileo Galilei", Università di Padova, Via Marzolo 8, I-35131 Padova, Italy*

[b]*INFN, Sezione di Padova, Via Marzolo 8, I-35131 Padova, Italy*

[c]*Centre de Physique Théorique, CNRS, École Polytechnique, IP Paris, 91128 Palaiseau, France*

[d]*Laboratoire de Physique Théorique (UMR8627), CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91405 Orsay, France*

[e]*INFN, Sezione di Roma Tre, Via della Vasca Navale 84, I-00146 Roma, Italy*

[f]*Paul Scherrer Institut, CH-5232 Villigen PSI, Switzerland*

[g]*IFIC, Universitat de València - CSIC, Apt. Correus 22085, E-46071, València, Spain*

[h]*Dept. de Física Quàntica i Astrofísica, Institut de Ciències del Cosmos (ICCUB), Universitat de Barcelona, Martí Franquès 1, E-08028 Barcelona, Spain*

[i]*INFN, Sezione di Roma, Piazzale A. Moro 2, I-00185 Roma, Italy*

[j]*Institute of Theoretical Physics, Faculty of Physics, University of Warsaw, ul. Pasteura 5, PL–02–093 Warsaw, Poland*

[k]*Theory Center, IPNS, KEK, Tsukuba 305-0801, Japan*

[l]*DESY, Notkestraße 85, D-22607 Hamburg, Germany*

[m]*Institut für Physik, Humboldt-Universität zu Berlin, D-12489 Berlin, Germany*

[n]*CERN, Geneva, Switzerland*

[o]*Physics Department, Florida State University, Tallahassee, FL 32306-4350, USA*

[p]*Theoretical Physics Department, CERN, Geneva, Switzerland*

[q]*Department of Physics and Astronomy, University of California, Irvine, CA 92697 USA*

[r]*Department of Physics, Tohoku University, Sendai, Miyagi 980-8578, Japan*

E-mail: hepfit-support@roma1.infn.it

ABSTRACT: `HEPfit` is a flexible open-source tool which, given the Standard Model or any of its extensions, allows to *i)* fit the model parameters to a given set of experimental observables; *ii)* obtain predictions for observables. `HEPfit` can be used either in Monte Carlo mode, to perform a Bayesian Markov Chain Monte Carlo analysis of a given model, or as a library, to obtain predictions of observables for a given point in the parameter space of the model, allowing `HEPfit` to be used in any statistical framework. In the present version, around a thousand observables have been implemented in the Standard Model and in several new physics scenarios. In this paper, we describe the general structure of the code as well as models and observables implemented in the current release.

arXiv:1910.14012v1 [hep-ph] 30 Oct 2019

Contents

1   Introduction

Searching for New Physics (NP) beyond the Standard Model (SM) in the era of the Large Hadron Collider (LHC) requires combining experimental and theoretical information from many sources to optimize the NP sensitivity. NP searches, even in the absence of a positive signal, provide useful information which puts constraints on the viable parameter space of any NP model. Should a NP signal emerge at future LHC runs or elsewhere, the combination of all available information remains a crucial step to pin down the actual NP model. NP searches at the LHC require extensive detector simulations and are usually restricted to a subset of simplified NP models. Given the high computational demand of direct searches, it is crucial to explore only regions of the parameter space compatible with other constraints. In this respect, indirect searches can be helpful and make the study of more general models viable.

HEPfit aims at providing a tool which allows to combine all available information to select allowed regions in the parameter space of any NP model. To this end, it can compute many observables with state-of-the-art theoretical expressions in a set of models which can be extended by the user. It also offers the possibility of sampling the parameter space using a Markov Chain Monte Carlo (MCMC) implemented using the BAT library [1–3]. Alternatively, HEPfit can be used as a library to obtain predictions of the observables in any implemented model. This allows to use HEPfit in any statistical framework.

HEPfit is written in C++ and parallelized with MPI. This is the first public release with a limited set of observables and models, which we plan to enlarge. The code is released under the GNU General Public License, so that contributions from users are possible and welcome. In particular, the present version provides Electroweak Precision Observables (EWPO), Higgs signal strengths, and several flavour observables in the SM, in Two-Higgs-Doublet Models (THDM), and in several parameterizations of NP contributions. Furthermore, it also calculates various Lepton Flavour Violating (LFV) observables in the Minimal Supersymmetric Standard Model (MSSM). In the near future, we plan to add many more flavour observables and to enlarge the spectrum of NP models.

The paper is organized as follows. In Section 2 we give a brief description of HEPfit including the statistical framework used, the MPI parallelization and some other details. In Section 3 we discuss the models implemented in HEPfit. In Section 4 we go on discussing some of the observables implemented in HEPfit. In Section 5 we present some physics results obtained using HEPfit in previous publications. Indeed several physics analyses [4–29] have been completed using HEPfit and serves as a validation of the code and its use as an open-source computational framework. A detailed description of the installation procedure can be found in Section 6 followed by examples of how to use HEPfit in Section 7. Updated information and detailed online documentation can be found on the HEPfit website [30].

2   The HEPfit code

HEPfit is a computational tool for the combination of indirect and direct constraints on High Energy Physics models. The code is built in a modular structure so that one can pick and choose which observables to use and what model to analyze. It also provides an interface that can be used to build customized models along with customized observables. This flexible framework allows defining a model of choice and observables that depend on the parameters of this model,

thus opening up the possibility of using `HEPfit` to perform phenomenological analyses in such a model.

The tool comes with a built-in statistical framework based on a Bayesian MCMC analysis. However, any statistical framework can be used along with this tool since a library is made available. `HEPfit` also allows for the incorporation of parametric and experimental correlations and can read likelihood distributions directly from `ROOT` histograms. This removes the necessity for setting experimental constraints through parameterized distributions which might require making approximations.

Since the statistical core of `HEPfit` is based on a MCMC, speed of computation is of utmost importance. `HEPfit` is already massively parallelized to run over large number of CPUs using `OpenMPI` and scales well to hundreds of processing units. The framework further brings forth the flexibility of defining a model of choice and observables that depend on the parameters of this model, thus opening up the possibility of performing various analyses using `HEPfit`.

The package comes with several examples of how `HEPfit` can be used and detailed documentation of the code and the physics can be found online on the `HEPfit` web site. Throughout its development, emphasis has been placed on speed and streamlining error handling. `HEPfit` has been tested through several analyses on various hardware architecture and displays reliable scaling to large systems.

2.1 Statistical framework

`HEPfit` can be used both as a library to compute the values of chosen observables and also as a MCMC based Bayesian analysis framework. While the former approach allows for choosing the statistical framework one wants to use, the latter uses a robust Bayesian MCMC framework implemented in the public code `BAT` [1–3]. In this section we give a brief overview of the Bayesian statistical framework implemented in `HEPfit` using `BAT`.

2.1.1 Bayesian framework

Once the model parameters, $\vec{\mathbf{x}}$, and the data, D, are defined one can define the posterior distribution according to Bayes theorem as:

$$P(\vec{\mathbf{x}}|D) = \frac{P(D|\vec{\mathbf{x}})P_0(\vec{\mathbf{x}})}{\int P(D|\vec{\mathbf{x}})P_0(\vec{\mathbf{x}})d\vec{\mathbf{x}}} \ , \tag{2.1}$$

where $P_0(\vec{\mathbf{x}})$ is the prior distribution of the parameters which represents the prior knowledge of the parameters which can come from experiments or theory computations or can be uninformative. The denominator is called the *normalization* or the *evidence*, the computation of which can allow for model comparison through the Bayes factor. The likelihood is denoted as $P(D|\vec{\mathbf{x}})$. Once the (unnormalized) posterior distribution[1] is mapped out using sampling methods (in our case a MCMC routine), one can obtain the marginalized posterior distributions of the individual parameters from which the credibility regions can be computed. The 1D marginalized distribution is given by

$$P(x_i|D) = \int P(\vec{\mathbf{x}}|D) \prod_{j \neq i} dx_j \ , \tag{2.2}$$

---

[1]While for a simple parameter space it is possible to compute the normalization factor, a MCMC analysis provides an unnormalized posterior distribution. This is the relevant ingredient for the purpose of studying any credibility interval. From now on, we implicitly assume we are dealing with an unnormalized posterior density.

where all the variables but the one for which the marginalized posterior distribution is being computed are integrated over, and similarly for marginalized 2D distributions.

2.1.2 Markov Chain Monte Carlo

In general, the posterior distribution specified in Eq. (2.1) cannot be computed easily, especially when there is a proliferation of model parameters. Using a naive Monte Carlo sampling algorithm can lead to unacceptable execution times because of their inherent inefficiency in sampling the parameter space. However, MCMC procedures overcome this hurdle and make the application of Bayes theorem quite tractable.

The implementation of MCMC in BAT uses a Metropolis-Hastings algorithm to sample the parameter space from the posterior. The steps of a Metropolis-Hastings algorithm for sampling from a (unnormalized) probability density $f(\vec{x})$ are as follows:

1. Start at a random point in the parameter space $\vec{x}$.

2. Generate a proposal point $\vec{y}$ according to a symmetric probability distribution $g(\vec{x}, \vec{y})$.

3. Compare the value of the function f at proposal point $\vec{y}$ with the value at the current point $\vec{x}$. The proposal point is accepted if:

   - $f(\vec{y}) \geq f(\vec{x})$,
   - otherwise, generate a random number r from a uniform distribution in the range $[0, 1]$ and accept the proposal if $f(\vec{y})/f(\vec{x}) > r$.

   If neither conditions are satisfied the proposal is rejected.

4. Continue from step 1.

In our case, the function $f(\vec{y})$ is the unnormalized posterior, namely the numerator of Eq. (2.1). The MCMC implementation consists of two parts. The first part is called the pre-run or the burn-in phase where the chains start from arbitrary random points in the parameter space and reach a stationary state after a certain number of iterations, through the tuning of the proposal function. The stationary state is reached once the targeted efficiency of the proposal and R-values close to one are obtained. The R-value for a parameter is essentially the distance of its mean values in the various chains in units of the standard deviation of the parameter in each chain [31, 32]. Samples of the parameter space are not collected during the pre-run. Once the pre-run is over, the samples of the parameter space are collected in the main run to get the marginalized distributions of all the parameters and the corresponding posterior distributions of the observables and of any other derived quantity that may have been defined. The details of the implementation of the MCMC framework can be found in Refs. [1–3].

2.1.3 Integration of BAT with HEPfit

The MCMC framework implemented in BAT is integrated in HEPfit using the library that is provided by BAT on compilation. The MonteCarloEngine class in HEPfit inherits from the BCModel class in BAT and overloads the LogLikelihood function. This method generates the numerical likelihood for one point in the parameter space with the values of the observables computed by HEPfit and the experimental and theoretical constraints provided to HEPfit. The parameters and

Physics Problem	Hardware	Run Configuration	Time
Unitarity Triangle Fit	3 nodes, 120 CPUs	120 chains, 1.4M iterations	00:02:10
	1 nodes, 40 CPUs	40 chains, 600K iterations	00:00:21
$b \to s$ decays in SMEFT[†] [24]	6 nodes, 240 CPUs	240 chains, 12.5K iterations	02:05:00
	6 nodes, 240 CPUs	240 chains, 39K iterations	05:20:00
combination of Higgs signal strengths and EWPO[28]	1 node, 16 CPUs	16 chains, 5M iterations	00:14:15
	1 node, 16 CPUs	16 chains, 24M iterations	02:08:00
$D \to PP$ decays and CP asymmetry[23]	3 nodes, 240 CPUs	240 chains, 4M iterations	00:18:30
	1 node, 8 CPUs	8 chains, 200K iterations	00:00:10

Table 1. *Some representative runs with* `HEPfit` *to show the advantages of the MPI implementation. Times are given in DD:HH:MM. The number of iterations refer to the sum total of pre-run and main-run iterations. The number of chains are equal to the number of CPUs by choice. [†] The $b \to s$ analysis is done with factorized priors, hence the number of iterations should be multiplied by the number of parameters ($\sim$50) to get a comparative estimate with the other cases. All runs performed in the BIRD or Maxwell clusters at DESY, Hamburg.*

their distributions are passed by `HEPfit` to `BAT` through the `MonteCarloEngine` class. `HEPfit` takes care of correlated parameter priors by rotating them to the eigenvector basis in order to increase the efficiency of sampling.

The output of a run, as detailed in Section 7.1, is produced by both `BAT` and `HEPfit`. All 1D and 2D marginalized distributions and posterior distributions are produced using the `BCH1D` and the `BCH2D` classes of `BAT` and stored in a `ROOT` file. One can choose to store the chains in the `ROOT` file as well using `HEPfit`. While this is useful for post-processing, since it makes the full sample available point by point, it entails a dramatic increase in size of the output `ROOT` file.

It should be noted that `BAT` is necessary only when running in the MCMC mode. If one chooses to run `HEPfit` as an event generator or to only compute values of observables for custom statistical analyses, the interface with `BAT` is not used at all.

2.2 Parallelization with MPI

One of the most important advantages of `HEPfit` over several other similar publicly available codes is that it is completely parallelized using `OpenMPI`, allowing it to be run on both single CPUs with multiple cores and on several nodes on large clusters. The MCMC algorithm is very apt for this kind of parallelization since an integer number of chains can be run on each core. Ideally allocating one core per chain minimizes the run time.

The official version of `BAT` is parallelized using `OpenMP`. However, `OpenMP` relies on shared memory and cannot be distributed over several nodes in a cluster. To overcome this limitation we used `OpenMPI` to parallelize both `BAT` and `HEPfit`. The parallelization is at the level of the computation of likelihood and observables. This means that the MCMC at both the pre-run and main run stages can take advantage of this parallelization. Once the likelihood computation (which requires the computation of the observables) is done, the flow is returned to the master, which performs the generation of the next set of proposal points. The computation of efficiencies and convergence, as well as the optimization of the proposal function, are currently not distributed since they require full information on the chain states. This is the only bottleneck in the parallelization, since the time the master takes to process these steps might be comparable to the time required

to compute all the observables by each chain, if the number of chains is very large. However, this begins to be a matter of concern only when the number of chains is in the range of several hundreds, a situation that a normal user is unlikely to encounter.

To demonstrate the advantages that one can get from the parallelization built into `HEPfit` and to give an estimate of the scaling of the run-times with the number of cores, we give some examples of analyses that can be done both on personal computers and on large clusters in Table 1. These should not be taken as benchmarks since we do not go into the details of the hardware, compiler optimization, etc. Rather, these should be taken as an indication of how MPI parallelization greatly enhances the performance of the `HEPfit` code.

2.3 Custom models and observables

Another unique feature that `HEPfit` offers is the possibility of creating custom models and custom observables. All the features of `HEPfit` are made available along with all the observables and parameters predefined in `HEPfit`. An example of such a use of `HEPfit` can be found in Ref. [23]. Detailed instructions for implementation are given in Section 7.4.

The user can define a custom model using a template provided with the package, by adding a set of parameters to any model defined in `HEPfit`. Generally, in addition to defining the new parameters, the user should also specify model-specific additional contributions to any observables predefined in `HEPfit` that he wants to use. Furthermore, new observables can be defined in terms of these new parameters.

New observables can also be defined in the context of the predefined `HEPfit` models. In this case, the user just needs to specify the observable in terms of the model parameters, without the need to create a custom model. The parameters already used in `HEPfit` can also be accessed. For example, one does not need to redefine the Cabibbo-Kobayashi-Maskawa (CKM) mixing matrix, $V_{\rm CKM}$, if one needs to use it in the computation of a custom observable. One can simply call the SM object available to all observables and then use the implementation of $V_{\rm CKM}$ already provided either in terms of the Wolfenstein parameters or in terms of the elements of the matrix. It should be noted that one does not need to define a custom model to define custom observables. A custom model should be defined only if the user requires parameters not already present in `HEPfit`. More details can be found in Section 7.4.

3 Models defined in `HEPfit`

The basic building blocks of `HEPfit` are the classes `Model` and `Observable`. Actual models extend the base class `Model` sequentially (e.g. `QCD` ← `StandardModel` ← `THDM` ← . . .). Inheritance allows a given model to use all the methods of the parent ones and to redefine those which have to include additional contributions specific to the extended model. For example, the method computing the strong coupling constant (α_s) includes strong corrections in `QCD`, adds electromagnetic corrections in `StandardModel`, and any additional contributions in classes extending the `StandardModel`. Models contain model parameters (both fundamental model parameters and auxiliary ones) and model flags which control specific options.

An instance of the `Observable` class contains the experimental information relative to a given physical observable as well as an instance of the class `ThObservable`, responsible for the computation of that observable in the given model. This is the class where both the experimental or theoretical constraints and the theory computation in the model are accessible, allowing for the

likelihood calculation. We now briefly review the models implemented in the current release of `HEPfit`.

3.1 The Standard Model

In `HEPfit`, the minimal model to be defined in order to compute any observable is the `StandardModel`, which for convenience extends a class `QCD`, which in turn, extends the abstract class `Model`.

The model implemented in the `QCD` class defines the following model parameters: the value of $\alpha_s(M)$ at a provided scale M, the $\overline{\text{MS}}$ quark masses $\bar{m}_q$ (except for the top quark mass where for convenience the pole mass is taken as input parameter and then converted to $\overline{\text{MS}}$). With this information, the class initializes instances of the `Particle` class for each quark. In addition, objects of type `Meson`, containing information on masses, lifetimes, decay constants and other hadronic parameters (these are taken as model parameters although in principle they are derived quantities), are instantiated for several mesons. Furthermore, bag parameters for meson mixings and decays are instantiated. This class also defines methods to implement the running of α_s and quark masses.

The `StandardModel` class extends `QCD` by adding the remaining SM parameters, namely the Fermi constant G_F, the fine-structure constant α, the Z boson mass M_Z, the Higgs boson mass m_h and the CKM mixing matrix (instantiating the corresponding object `CKM`).[2] The Pontecorvo-Maki-Nakagawa-Sakata (PMNS) mixing matrix is defined but currently not activated. It also fixes the `QCD` parameter M introduced above to M_Z and the `QCD` parameter $\alpha_s(M)$ to $\alpha_s(M_Z)$. Furthermore, it contains `Particle` objects for leptons. Several additional model parameters describe the hadronic vacuum polarization contribution to the running of α, and the theoretical uncertainties in the W mass and other EWPO, for which is convenient to use available numerical estimates. Moreover, the running of α_s is extended to include electromagnetic corrections.

The `StandardModel` class also provides matching conditions for weak effective Hamiltonians through the class `StandardModelMatching`. Low-energy weak effective Hamiltonians, both $\Delta F = 1$ and $\Delta F = 2$, are provided on demand by the class `Flavour` instantiated by `StandardModel`.

Although extending `Model` and `QCD`, `StandardModel` is the actual base class for any further definition of NP models (e.g. THDM, SUSY, etc.). Details on the implementation of `StandardModel` and `QCD` can be found in the online documentation.

3.2 Two-Higgs-Doublet models

One of the most straightforward extensions of the SM is the Two-Higgs-Doublet model (THDM) [33–35]. No fundamental theorem forbids to add a second scalar doublet to the SM particle content. The THDM can offer a solution to problems as the stability of the scalar potential up to very large scales (see e.g. ref. [36]) or electroweak baryogenesis (see e.g. refs. [37–39]), which cannot be solved in the SM. Furthermore, it could emerge as an effective description of more complicated models like SUSY models, which necessarily contain two Higgs doublets.

There are several THDM variants with different phenomenological implications. At the moment `HEPfit` contains the versions which exclude flavour-changing neutral currents at tree-level as well as CP violation in the Higgs sector. In order to fulfil the first demand, an additional softly broken Z_2 symmetry is assumed, which can be chosen in four different ways; thus these versions

---

[2]Two CKM parameterizations (the Wolfenstein one and a parameterization using $|V_{us}|$, $|V_{cb}|$, $|V_{ub}|$, and the angle γ of the unitarity triangle as inputs) can be selected using the model flag `FlagWolfenstein`.

Type I	Type II	Type X ("lepton specific")	Type Y ("flipped")
$Y_1^d \equiv 0$, $Y_1^\ell \equiv 0$	$Y_2^d \equiv 0$, $Y_2^\ell \equiv 0$	$Y_1^d \equiv 0$, $Y_2^\ell \equiv 0$	$Y_2^d \equiv 0$, $Y_1^\ell \equiv 0$

Table 2. Yukawa couplings in the four possible Z_2 symmetric THDM types.

are called type I, type II, type X and type Y.[3] The four types only differ in the Yukawa couplings of the Higgs fields. The corresponding assignments can be found in Table 2, where Y_j^f denotes the coupling of one of the two Higgs doublets Φ_j ($j = 1, 2$) to the fermion field f. By definition, $Y_1^u \equiv 0$ for all four types. In the configuration file `THDM.conf`, one has to choose the THDM type by setting the flag `modelTypeflag` to `type1`, `type2`, `typeX` or `typeY`.

We write the Higgs potential for Φ_1 and Φ_2 as

$$V_H^{\text{THDM}} = m_{11}^2 \Phi_1^\dagger \Phi_1 + m_{22}^2 \Phi_2^\dagger \Phi_2 - m_{12}^2 \left[\Phi_1^\dagger \Phi_2 + \Phi_2^\dagger \Phi_1 \right] + \tfrac{1}{2}\lambda_1 (\Phi_1^\dagger \Phi_1)^2 + \tfrac{1}{2}\lambda_2 (\Phi_2^\dagger \Phi_2)^2$$
$$+ \lambda_3 (\Phi_1^\dagger \Phi_1)(\Phi_2^\dagger \Phi_2) + \lambda_4 (\Phi_1^\dagger \Phi_2)(\Phi_2^\dagger \Phi_1) + \tfrac{1}{2}\lambda_5 \left[(\Phi_1^\dagger \Phi_2)^2 + (\Phi_2^\dagger \Phi_1)^2 \right], \quad (3.1)$$

and the Yukawa part of the Lagrangian as

$$\mathcal{L}_Y^{\text{THDM}} = -Y_2^u \bar{Q}_L \tilde{\Phi}_2 u_R - \sum_{j=1}^{2} \left[Y_j^d \bar{Q}_L \Phi_j d_R + Y_j^\ell \bar{L}_L \Phi_j \ell_R \right] + \text{h.c.},$$

where one of the choices from Table 2 has to be applied.

The THDM contains five physical Higgs bosons, two of which are neutral and even under CP transformations, one is neutral and CP-odd, and the remaining two carry the electric charge ± 1 and are degenerate in mass. We assume that the 125 GeV resonance measured at the LHC is the lighter CP-even Higgs h, while the other particles are labelled H, A and $H^\pm$, respectively. The eight parameters from the Higgs potential (3.1) can be transformed into physical parameters:

- the vacuum expectation value v,

- the lighter CP-even Higgs-boson mass m_h,

- the heavier CP-even Higgs-boson mass m_H,

- the CP-odd Higgs-boson mass m_A,

- the charged Higgs-boson mass m_{H^+},

- the mixing angle α,

- the mixing angle β and

- the soft Z_2 breaking parameter m_{12}^2 from (3.1).

The Fermi constant G_F and m_h are defined in the SM configuration file. For practical reasons, the `HEPfit` implementation uses $\beta - \alpha$ and $\log_{10} \tan\beta$, instead of α and β, and squared H, A and H^+ masses.

---

[3]The THDM of type II contains the scalar Higgs part of the MSSM.

HEPfit name	Parameter
delEps_1	$\delta\epsilon_1$
delEps_2	$\delta\epsilon_2$
delEps_3	$\delta\epsilon_3$
delEps_b	$\delta\epsilon_b$

Table 3. Model parameters in the NPEpsilons_pureNP class.

3.3 The Georgi-Machacek model

In the Georgi-Machacek model [40, 41], the SM is extended by two $SU(2)$ triplets. This construction can simultaneously explain the smallness of neutrino masses (via the seesaw mechanism) and the electroweak ρ parameter. In HEPfit, we implemented the custodial Georgi-Machacek model, in which the additional heavy scalars can be combined into a quintet, a triplet and a singlet under the custodial $SU(2)$ with masses m_5, m_3, and m_1, respectively. Further model parameters in HEPfit are the triplet vev v_Δ, the singlet mixing angle α and the two trilinear couplings μ_1 and μ_2. For details of the HEPfit implementation of this model we refer to reference [18].

3.4 Oblique corrections in electroweak precision observables

Assuming the physics modifying the on-shell properties of the W and Z bosons is universal, such effects can be encoded in three quantities: the relative normalization of neutral and charged currents, and the two relative differences between the three possible definitions of the weak mixing angle. These effects are captured by the so-called ϵ_i parameters introduced in [42–44]. The model class NPEpsilons_pureNP describes the NP contributions to these quantities. It also allows contributions in the additional, non-universal, ϵ_b parameter, also introduced in [44] to describe modifications of the $Zb\bar{b}$ interactions. The model parameters in this class are defined in Table 3.

The $\delta\epsilon_i$, $i = 1, 2, 3$ can be readily mapped into the oblique parameters describing NP modifying the propagator of the electroweak gauge bosons:

$$\delta\varepsilon_1 = \alpha T - W + 2X\frac{\sin\theta_w}{\cos\theta_w} - Y\frac{\sin^2\theta_w}{\cos^2\theta_w}, \tag{3.2}$$

$$\delta\varepsilon_2 = -\frac{\alpha}{4\sin^2\theta_w}U - W + 2X\frac{\sin\theta_w}{\cos\theta_w} - V, \tag{3.3}$$

$$\delta\varepsilon_3 = \frac{\alpha}{4\sin^2\theta_w}S - W + \frac{X}{\sin\theta_w\cos\theta_w} - Y, \tag{3.4}$$

where θ_w is the weak mixing angle, the S, T, U parameters were originally introduced in Ref. [45] and V, W, X, Y in Ref. [46]. All these parameterize the different coefficients in the expansion of the gauge boson self-energies for $q^2 \ll \Lambda^2$ with Λ the typical scale of the NP. Traditionally, the literature of electroweak precision tests has focused on the first three parameters (which also match the number of different universal effects that can appear in the EWPO). Because of that, we include the model class NPSTU, which describes this type of NP. The relevant parameters are collected in Table 4. It is important to note, however, that the U parameter is typically expected to be suppressed with respect to S, T by M_W^2/Λ^2. Indeed, at the leading order in M_W^2/Λ^2 the four parameters describing universal NP effects in electroweak observables are S, T, W and Y [46].

HEPfit name	Parameter
obliqueS	S
obliqueT	T
obliqueU	U

Table 4. Model parameters in the NPSTU class.

3.5   The dimension-six Standard Model Effective Field Theory

When the typical mass scale of NP is significantly larger than the energies tested by the experimental observables, the new effects can be described in a general way by means of an effective Lagrangian

$$\mathcal{L}_{\text{eff}} = \mathcal{L}_{\text{SM}} + \sum_{d>4} \frac{1}{\Lambda^{d-4}} \mathcal{L}_d. \tag{3.5}$$

In Eq. (3.5) $\mathcal{L}_{\text{SM}}$ is the SM Lagrangian, Λ is the cut-off scale where the effective theory ceases to be valid, and

$$\mathcal{L}_d = \sum_i C_i^{(d)} \mathcal{O}_i^{(d)} \tag{3.6}$$

contains only (Lorentz and) gauge-invariant local operators, $\mathcal{O}_i^{(d)}$, of mass dimension d. In the so-called SM effective field theory (SMEFT), these operators are built using exclusively the SM symmetries and fields, assuming the Higgs belongs to an $SU(2)_L$ doublet. The Wilson coefficients, $C_i^{(d)}$, encode the dependence on the details of the NP model. They can be obtained by matching with a particular ultraviolet (UV) completion of the SM [47–59], allowing to project the EFT results into constraints on definite scenarios.

At any order in the effective Lagrangian expansion a complete basis of physically independent operators contains only a finite number of higher-dimensional interactions. In particular, for NP in the multi-TeV region, the precision of current EW measurements only allows to be sensitive to the leading terms in the $1/\Lambda$ expansion in Eq. (3.5), i.e., the *dimension-six effective Lagrangian* (at dimension five there is only the Weinberg operator giving Majorana masses to the SM neutrinos, which plays a negligible role in EW processes). The first complete basis of independent dimension-six operators was introduced by Grzadkowski, Iskrzynski, Misiak, and Rosiek and contains a total of 59 independent operators, barring flavour indices and Hermitian conjugates [60]. This is what is commonly known in the literature as the *Warsaw* basis.

The main implementation of the dimension-six Standard Model effective Lagrangian in HEPfit is based in the Warsaw basis, though other operators outside this basis are also available for some calculations. Currently, all the dimension-six interactions entering in the EWPO as well as Higgs signal strengths have been included in the NPSMEFTd6 model class. Two options are available, depending on whether lepton and quark flavour universality is assumed (NPSMEFTd6_LFU_QFU) or not (NPSMEFTd6). These implementations assume that we use the $\{M_Z, \alpha, G_F\}$ scheme for the SM EW input parameters. The complete list of operators as well the corresponding names for the HEPfit model parameters can be found in the online documentation along with a complete description of the model flags.[4]

---

[4]The free parameters in the model also include several nuisance parameters to control theory uncertainties in certain Higgs processes.

Name	Parameter	Name	Parameter	Name	Parameter	Name	Parameter
Kw	κ_W	Kz	κ_Z	Kg	κ_g	Kga	κ_γ
Kzga	$\kappa_{Z\gamma}$	Ku	κ_u	Kc	κ_c	Kt	κ_t
Kd	κ_d	Ks	κ_s	Kb	κ_b	Ke	κ_e
Kmu	κ_μ	Ktau	κ_τ	BrHinv	$\mathrm{BR}_{\mathrm{inv}}$	BrHexo	$\mathrm{BR}_{\mathrm{exo}}$

Table 5. Model parameters in the `HiggsKigen` class. "Name" refers to the name of the parameter in `HEPfit` that can be used in the configuration files.

By default, the theoretical predictions for the experimental observables including the NP contributions coming from the effective Lagrangian are computed consistently with the assumption of only dimension-six effects. In other words, for a given observable, O, only effects of order $1/\Lambda^2$ are considered, and all NP contributions are linear in the NP parameters:

$$O = O_{\mathrm{SM}} + \sum_i F_i \frac{C_i}{\Lambda^2}. \tag{3.7}$$

Note that these linear contributions always come from the interference with the SM amplitudes. The default behaviour described above can be altered by using some of the model flags. The flag `QuadraticTerms` will allow to test the quadratic effects of the dimension-six interactions, to be implemented in future releases of the code.

3.6 Modified Higgs couplings in the κ-framework

In many scenarios of NP one of the main predictions are deviations in the Higgs boson couplings with respect to the SM ones. Such a scenario can be described in general by considering the following effective Lagrangian for a light Higgs-like scalar field h [61, 62]:

$$\begin{aligned}
\mathcal{L} = &\frac{1}{2}\partial_\mu h \partial^\mu h - V(h) + \frac{v^2}{4}\mathrm{Tr}(D_\mu \Sigma^\dagger D^\mu \Sigma)\left(1 + 2\kappa_V \frac{h}{v} + \dots\right) \\
&- m_{u^i}\left(\overline{u_L^i}\ \overline{d_L^i}\right)\Sigma \begin{pmatrix} \overline{u_R^i} \\ 0 \end{pmatrix}\left(1 + \kappa_u \frac{h}{v} + \dots\right) + \mathrm{h.c.} \\
&- m_{d^i}\left(\overline{u_L^i}\ \overline{d_L^i}\right)\Sigma \begin{pmatrix} 0 \\ d_R^i \end{pmatrix}\left(1 + \kappa_d \frac{h}{v} + \dots\right) + \mathrm{h.c.} \\
&- m_{\ell^i}\left(\overline{\nu_L^i}\ \overline{\ell_L^i}\right)\Sigma \begin{pmatrix} 0 \\ \ell_R^i \end{pmatrix}\left(1 + \kappa_\ell \frac{h}{v} + \dots\right) + \mathrm{h.c.}.
\end{aligned} \tag{3.8}$$

This Lagrangian assumes an approximate custodial symmetry and the absence of other light degrees of freedom below the given cut-off scale. In the previous Lagrangian the longitudinal components of the W and Z gauge bosons, $\chi^a(x)$, are described by the 2×2 matrix $\Sigma(x) = \exp(i\sigma_a \chi^a(x)/v)$, with σ_a the Pauli matrices, and $V(h)$ is the scalar potential of the Higgs field, whose details are not relevant for the discussion here. The SM is recovered for $\kappa_V = \kappa_u = \kappa_d = \kappa_\ell = 1$. Deviations in such a class of scenarios (and beyond) are conveniently encoded in the so-called κ framework [63]. In this parameterization, deviations from the SM in the Higgs properties are described by coupling modifier, κ_i, defined from the different Higgs production cross sections and decay widths. Schematically,

$$(\sigma \cdot \mathrm{BR})(i \to H \to f) = \kappa_i^2 \sigma^{\mathrm{SM}}(i \to H)\frac{\kappa_f^2 \Gamma^{\mathrm{SM}}(H \to f)}{\Gamma_H}, \tag{3.9}$$

where the total Higgs width, allowing the possibility of non-SM invisible or exotic decays, parameterized by $\mathrm{BR}_{\mathrm{inv}}$ and $\mathrm{BR}_{\mathrm{exo}}$, can be written as

$$\Gamma_H = \Gamma_H^{\mathrm{SM}} \frac{\sum_i \kappa_i^2 \mathrm{BR}_i^{\mathrm{SM}}}{1 - \mathrm{BR}_{\mathrm{inv}} - \mathrm{BR}_{\mathrm{exo}}}. \tag{3.10}$$

The model class `HiggsKigen` contains a general implementation of the parameterization described in the κ framework, offering also several flags to adjust some of the different types of assumptions that are commonly used in the literature. The most general set of coupling modifiers allowed in this class is described in Table 5, including also the possibility for non-SM contributions to invisible or exotic (non-invisible) Higgs decays.[5] Note that, even though the coupling modifiers are defined for all SM fermions, the current implementation of the code neglects modifications of the Higgs couplings to strange, up and down quarks, and to the electron. Furthermore, the parameters associated to $\kappa_{g,\gamma,Z\gamma}$, which are typically used in an attempt to interpret data allowing non-SM particles in the SM loops, are only meaningful if the model flag `KiLoop` is active.

Finally, in scenarios like the one in Eq. (3.8), while both κ_V and κ_f can modify the different Higgs production cross sections and decay widths, the leading corrections to EWPO come only from κ_V. These are given by the following 1-loop contributions to the oblique S and T parameters:

$$S = \frac{1}{12\pi}(1 - \kappa_V^2)\ln\left(\frac{\Lambda^2}{m_H^2}\right), \qquad T = -\frac{3}{16\pi c_W^2}(1 - \kappa_V^2)\ln\left(\frac{\Lambda^2}{m_H^2}\right), \tag{3.11}$$

where Λ is the cutoff of the effective Lagrangian in Eq. (3.8). We set $\Lambda = 4\pi v/\sqrt{|1 - \kappa_V^2|}$, as given by the scale of violation of perturbative unitarity in WW scattering.

The above contributions from κ_V to EWPO are also implemented in the `HiggsKigen` class, where κ_V is taken from the model parameter associated to the W coupling, κ_W. Note however that, for $\kappa_W \neq \kappa_Z$ power divergences appear in the contributions to oblique corrections, and the detailed information of the UV theory is necessary for calculating the contributions to EWPO. Therefore, in `HiggsKigen` the use and interpretation of EWPO is subject to the use of the flag `Custodial`, which enables $\kappa_W = \kappa_Z$.

Other flags in the model allow to use a global scaling for all fermion couplings (flag `UniversalKf`), a global scaling for all SM couplings (flag `UniversalK`), and to trade the exotic branching ratio parameter by a scaling of the total Higgs width, according to Eq. (3.10) (flag `UseKH`).

4 Some important observables implemented in `HEPfit`

A large selection of observables has been implemented in `HEPfit`. Broadly speaking, these observables can be classified into those pertaining to electroweak physics, Higgs physics, and flavour physics. Observables should not necessarily be identified with experimentally accessible quantities, but can also be used to impose theoretical constraints, such as unitarity bounds, that can constrain the parameter space of theoretical models, particularly beyond the SM. In what follows we give a brief overview of the main observables that are available in `HEPfit` along with some details about their implementation when necessary.

---

[5]As in the `NPSMEFTd6` class, there are several nuisance parameters in the model to control theory uncertainties in certain Higgs processes. We refer to the documentation for a extensive list of the model parameters.

4.1 Electroweak physics

The main EWPO have been implemented in `HEPfit`, including Z-pole observables as well as properties of the W boson (e.g. W mass and decay width). The SM predictions for these observables are implemented including the state-of-the-art of radiative corrections, following the work in references [64–94]. In the current version of `HEPfit`, all these observables are computed as a function of the following SM input parameters: the Z, Higgs and top-quark masses, M_Z, m_h and m_t, respectively; the strong coupling constant at the Z-pole, $\alpha_s(M_Z^2)$, and the 5-flavour contribution to the running of the electromagnetic constant at the Z-pole, $\Delta\alpha_{\text{had}}^{(5)}(M_Z^2)$.

The predictions including modifications due to NP effects are also implemented for different models/scenarios, e.g. oblique parameters [42–45, 95–97], modified Z couplings [49, 98–110], the SMEFT [60, 111], etc.

4.2 Higgs physics

In the Higgs sector, most of the observables currently included in `HEPfit` are the Higgs-boson production cross sections or branching ratios, always normalized to the corresponding SM prediction. Modifications with respect to the SM are implemented for several models, e.g. `NPSMEFTd6` or `HiggsKigen`. This set of observables allows to construct the different signal strengths for each production×decay measured at the LHC experiments and to test different NP hypotheses.

Apart from the observables needed for LHC studies, the corresponding observables for the production at future lepton colliders are also implemented in `HEPfit`. These are available for different values of centre-of-mass energies and/or polarization fractions, covering most of the options present in current proposals for such future facilities. Observables for studies at ep colliders or at 100 TeV pp colliders are also available in some cases.

4.3 Flavour physics

The list of observables already implemented in `HEPfit` includes several leptonic and semileptonic weak decays of flavoured mesons, meson-antimeson oscillations, and lepton flavour and universality violations. All these observables have been also implemented in models beyond the SM.

`HEPfit` has a dedicated flavour program in which several $\Delta F = 2$, $\Delta F = 1$ [6, 8, 9, 11, 12, 14, 21, 112–114] observables have been implemented to state-of-the-art precision in the SM and beyond. `HEPfit` also includes observables that require lepton flavour violation. In Table 6 we list some of the processes that have either been fully implemented (✓) or are currently under development (○). We also list out the models in which they have been implemented. H_{eff} refers here both to NP in the weak effective Hamiltonian as well as to NP in the SMEFT. `HEPfit` is continuously under development and the list of available observables keeps increasing. The complete list can be found in the online documentation.

4.4 Model-specific observables

Explicit NP models usually enlarge the particle spectrum, leading to model-specific observables connected to (limits on) properties of new particles (masses, production cross sections, etc.). Furthermore, theoretical constraints such as vacuum stability, perturbativity, etc. might be applicable to NP models. Both kinds of observables have been implemented for several models extending the SM Higgs sector.

Processes	SM	THDM	MSSM	H_{eff}
$\Delta F = 2$	✓	✓	○	✓
$B \to \tau\nu$	✓	✓	○	○
$B \to D^{(*)}\ell\nu_\ell$	✓	✓		✓
$B_q \to \mu\mu$	✓	○	○	○
rare K decays	○			○
$B \to X_s\gamma$	✓	✓	○	✓
$B \to V\gamma$	✓			✓
$B \to P/V\ell^+\ell^-$	✓			✓
$B \to X_s\ell^+\ell^-$	○			○
$\ell_i \to \ell_j\gamma$			✓	
$\ell_i \to 3\ell_j$			✓	
$(g-2)_\mu$			✓	

Table 6. Some processes that have been implemented (✓) or are under development (○) in `HEPfit` for flavour physics.

For example, in the THDM with a softly broken Z_2 symmetry we implemented the conditions that the Higgs potential is bounded from below at LO [115] and that the unitarity of two-to-two scalar scattering processes is perturbative at NLO [8, 116, 117]. These requirements can also be imposed at higher scales; the renormalization group running is performed at NLO [36]. Also the possibility that the Higgs potential features a second minimum deeper than the electroweak vacuum [118] can be checked in `HEPfit`. Similar constraints can be imposed on the Georgi-Machacek model. In this case, both boundedness from below [119] and unitarity [120] are available at LO.

5 Selected results using `HEPfit`

`HEPfit` has so far been used to perform several analyses of electroweak, Higgs and flavour physics in the SM and beyond. In this section we highlight some of the results that have been obtained, accompanied by a brief summary. The details of these analyses can be found in the original publications.

5.1 Electroweak and Higgs physics

The first paper published using the nascent `HEPfit` code featured a full-fledged analysis of EWPO in the SM and beyond [4], later generalized to include more NP models and Higgs signal strengths [7, 17, 28]. In the top left plot of Figure 1, taken from ref. [4], we show the two-dimensional probability distribution for the NP parameters ε_1 and ε_3 [42, 43], obtained assuming $\varepsilon_2 = \varepsilon_2^{\text{SM}}$ and $\varepsilon_b = \varepsilon_b^{\text{SM}}$. The impact of different constraints is also shown in the plot. The top right plot, from ref. [7], presents the results for modified Higgs couplings, multiplying SM Higgs couplings to vector bosons by a universal scaling factor κ_V and similarly for fermions by a universal factor κ_f. The figure shows the interplay between Higgs observables and EWPO in constraining the modified couplings. The bottom plot, taken from ref. [28], gives a pictorial representation of constraints on several effective couplings, including correlations, for different future lepton colliders. The `HEPfit` code was also used to obtain most of the results presented in the future collider comparison study
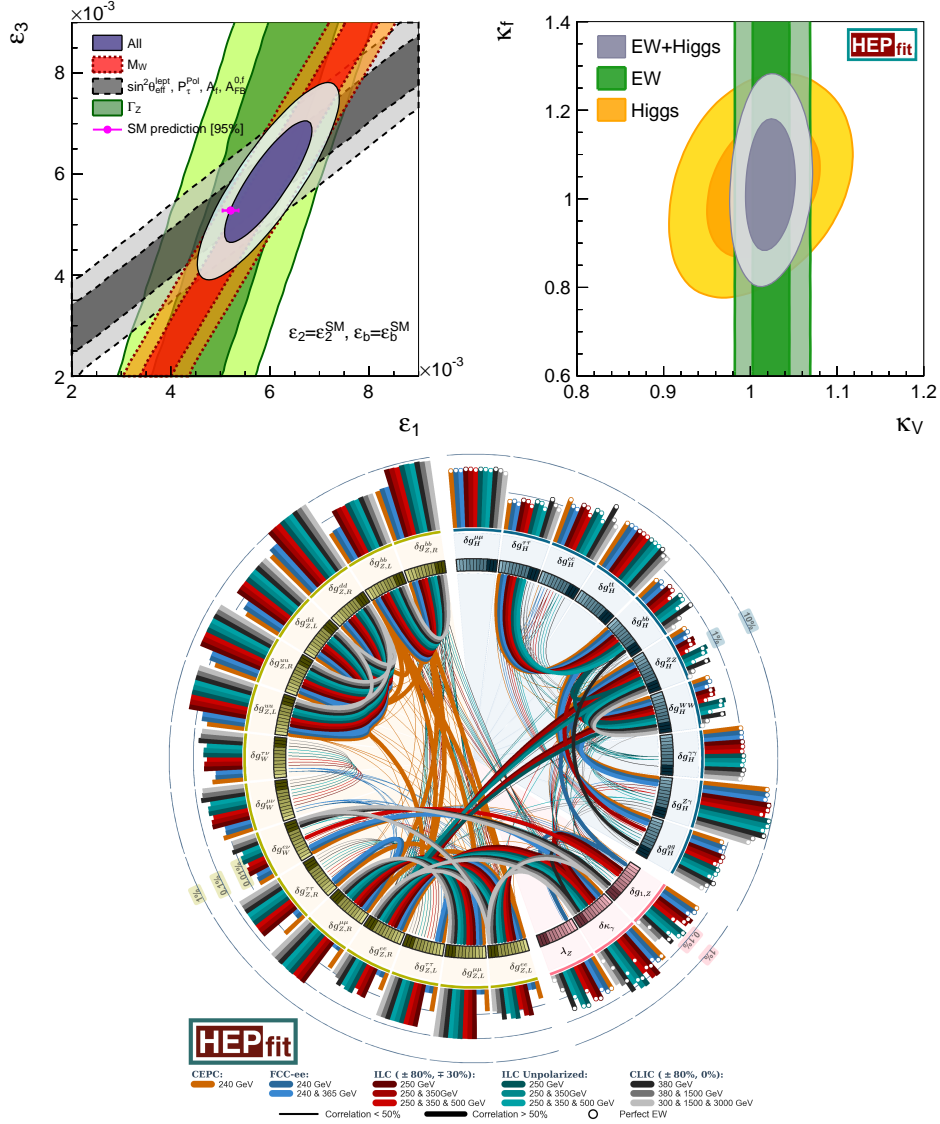
Figure 1. *Selected results from the papers presented in Section 5.1. Top left (from ref. [4]): two-dimensional probability distribution for ε_1 and ε_3 in the fit, assuming $\varepsilon_2 = \varepsilon_2^{SM}$ and $\varepsilon_b = \varepsilon_b^{SM}$, showing the impact of different constraints. The SM prediction at 95% is denoted by a point with an error bar. Top right (from ref. [7]): two-dimensional 68% (dark) and 95% (light) probability contours for κ_V and κ_f (from darker to lighter), obtained from the fit to the Higgs-boson signal strengths and the EWPO. Bottom (from ref. [28]): a scheme-ball illustration of the correlations between Higgs and EW sector couplings. The Z-pole runs are included for FCC-ee and CEPC. Projections from HL-LHC and measurements from LEP and SLD are included in all scenarios. The outer bars give the 1σ precision on the individual coupling.*

in Ref. [27]. These are summarized in the *Electroweak Physics* chapter of the *Physics Briefing Book* [121], prepared as input for the Update of the European Strategy for Particle Physics 2020.[6]
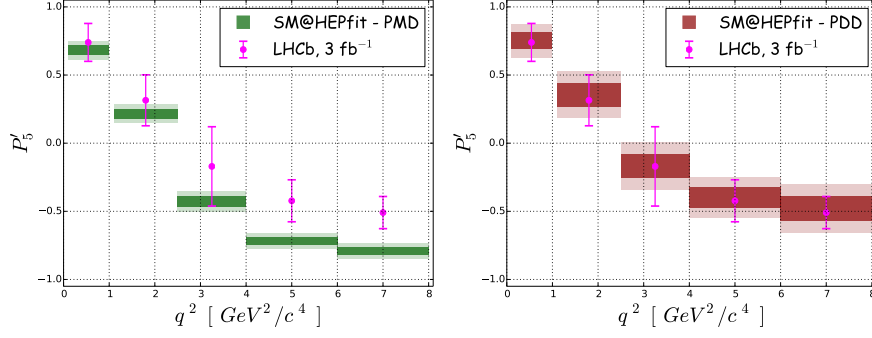
Figure 2. *Results of a fit for the LHCb results on the angular variable P_5' in two different theoretical scenarios: assuming the validity of an extrapolation of the QCD sum rules calculation of ref. [126] at maximum hadronic recoil to the full kinematic range (left), or allowing for sizable long-distance contributions to be present for q^2 closer to $4m_c^2$ (right). For more details see ref. [21]*
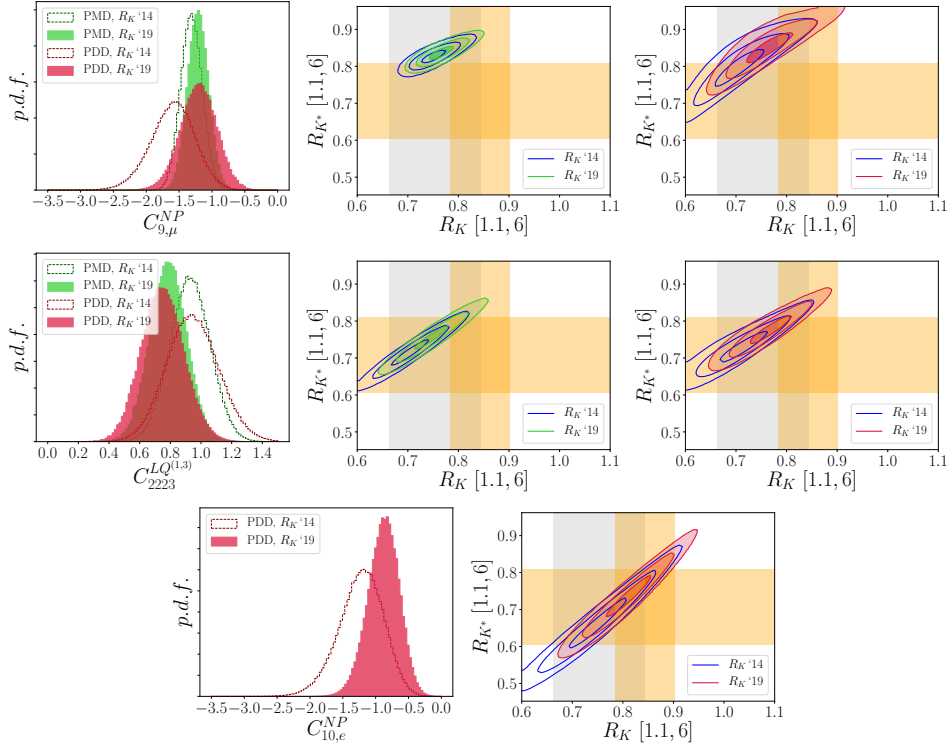
Figure 3. *First row: probability density function (p.d.f.) for the NP contribution to the Wilson coefficient $C_{9,\mu}^{\mathrm{NP}}$. The green-filled p.d.f. shows the posterior obtained in the optimistic approach to hadronic contributions after the inclusion of the updated measurement for R_K, while the red-filled p.d.f. is the analogous posterior obtained allowing for sizable hadronic contributions (the dashed posteriors are the ones obtained employing the 2014 R_K measurement); the following panels report the combined 2D p.d.f. of the corresponding results for R_K and R_{K^*}, where the colour scheme follows the one employed in the first panel. The horizontal band corresponds to the 1σ experimental region for R_{K^*} from [127], while the two vertical bands corresponds to the previous and the current 1σ experimental regions for R_K. Second row: analogous to the first row, but relative to the SMEFT Wilson coefficient C_{2223}^{LQ}. Third row: analogous to the first row, but relative to the NP contribution to the Wilson coefficient $C_{10,e}^{\mathrm{NP}}$. More details can be found in [24].*
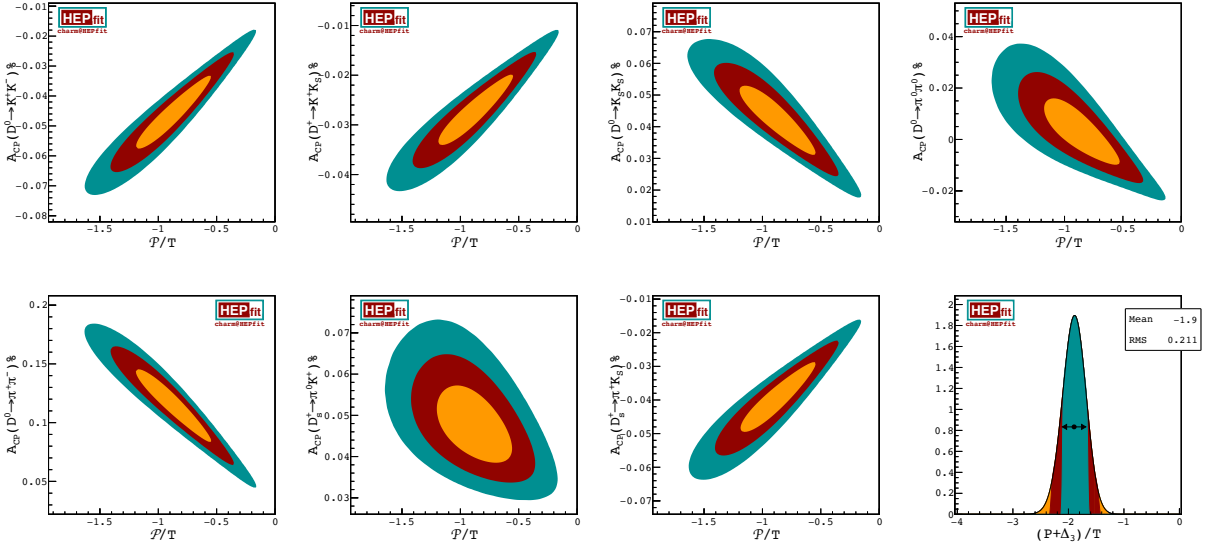
Figure 4. *The correlations between the ratio of the penguin and tree contributions, $\mathcal{P}/T$, and the CP asymmetries (given in %). HFLAV world average of $\Delta A_{\rm CP}$ has been used for the fit and these CP asymmetries correspond to the negative solution for the phases. The orange, red and green regions are the 68%, 95% and 99% probability regions respectively. The bottom right-most panel shows the fit to $(P + \Delta_3)/T = \mathcal{P}/T - 1$. The orange, red and green regions are the 68%, 95% and 99% probability regions respectively for the 2D histograms and the contrary for the 1D histogram. More details can be found in Ref. [23].*

5.2 Flavour physics

Analyses in flavour physics using `HEPfit` has produced several results following the claimed anomalies in B physics. We started off by reexamining the SM theoretical uncertainties and the possibility of explaining the anomalies claimed in the angular distribution of $B \to K^* \ell^+ \ell^-$ decays through these uncertainties [6, 9, 12, 21, 22]. We showed that the anomalies in the angular coefficients P_5' could be explained by allowing for a conservative estimate of the theoretical uncertainties, see Figure 2.

Having shown that the claimed deviations in the angular observables from the SM predictions could be explained by making a more conservative assumption about the non-perturbative contributions, we addressed the cases for the deviations from unity of the measured values of the lepton non-universal observables $R_{K^{(*)}}$, fitting simultaneously for the NP Wilson coefficients and the non-perturbative hadronic contributions [11, 24]. As before, we studied the impact of hadronic contributions on the global fit. The conclusions from our study were quite clear: on one hand the flavour universal effects could be explained by enlarged hadronic effects, reducing the significance of flavour universal NP effects. On the other hand, flavour non-universal effects could only be explained by the presence of NP contributions. In Figure 3 we present some of our results. More details can be found in Ref. [24].

Besides B physics, `HEPfit` has also been used for the analysis of final state interactions (FSI) and CP asymmetries in $D \to PP$ ($P = K, \pi$) decays. These have recently come to the forefront of measurements with the pioneering 5σ observation of $\Delta A_{\rm CP} = A_{\rm CP}(D \to K^+ K^-) - A_{\rm CP}(D \to$
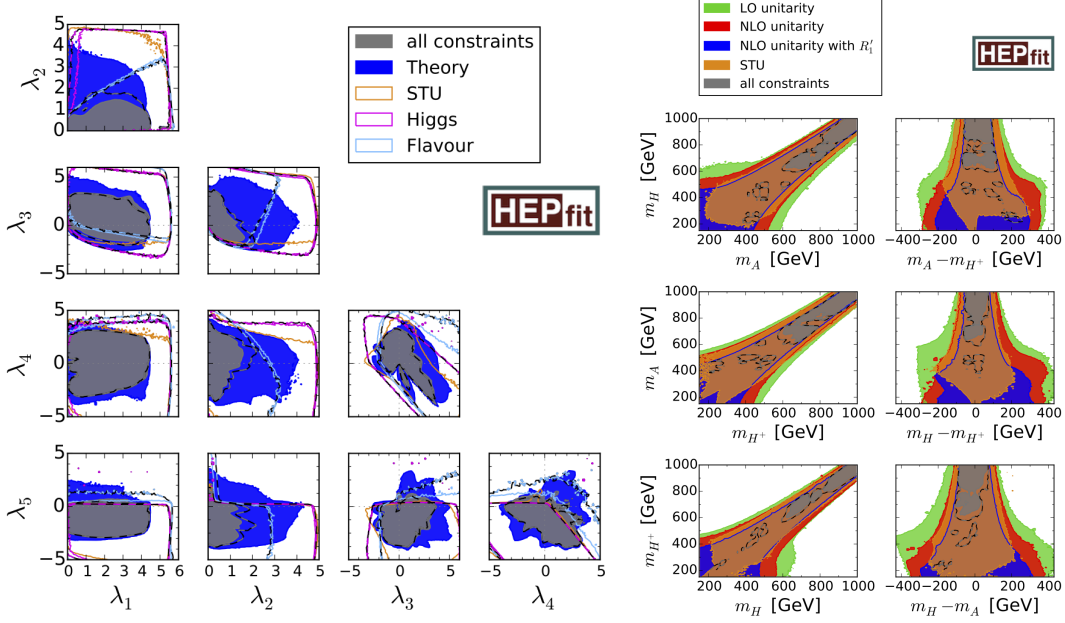
Figure 5. *Left panel: λ_i vs λ_j planes. The blue shaded regions are 99.7% probability areas taking into account theoretical constraints described in Section 4.4. Orange, pink and light blue lines mark the 95.4% boundaries of fits using only the oblique parameters (STU), all Higgs observables (strengths and direct searches) and flavour observables, respectively. The grey contours are compatible with all theoretical and experimental bounds at a probability of 95.4%. The solid lines are understood as the type II contours, the coloured dashed lines represent the corresponding type I fits. Right panel: Allowed regions in the heavy Higgs boson masses and their mass differences planes in the THDM of type I (dashed lines) and type II (solid lines). The unitarity bounds to the green, red and blue regions are meant at a probability of 99.7%, and the orange and grey lines mark the 95.4% boundaries. More details can be found in [8].*

$\pi^+\pi^-$) made by the LHCb collaboration [128–130]. This work takes advantage of the high precision reached by the measurements of the branching ratios in two particle final states consisting of kaons and/or pions of the pseudoscalar charmed particles to deduce the predictions of the SM for the CP violating asymmetries in their decays. The amplitudes are constructed in agreement with the measured branching ratios, where the $SU(3)_F$ violations come mainly from the FSI and from the non-conservation of the strangeness changing vector currents. A fit is performed of the parameters to the branching fractions and ΔA_{CP} using HEPfit and predict several CP asymmetries using our parameterization. In Figure 4 the fit to the penguin amplitude and the predictions for the CP asymmetries are shown. More details can be found in Ref. [23].

5.3 Constraints on specific new physics models: the case of extended scalar sectors

Several scalar extensions of the SM have been analysed using HEPfit. The THDM with a softly broken Z_2 symmetry has been widely studied taking into account most of the relevant constraints available at the moment. Theoretical constraints described in Section 4.4 are very useful to restrict the NP parameter space. In this case approximate expressions for the NLO perturbative unitarity conditions were obtained following the method described in [131]. These expressions are valid in the large center-of-mass limit and therefore they are only considered above a certain energy scale, default value of which is set to 750 GeV. As shown in the left panel of Figure 5, constraints on the $\lambda_i - \lambda_j$ (see Eq. (3.1)) planes can be obtained. These can be translated into constraints on

physical observables such as the mass splitting of the scalar particles, $m_H - m_A, m_H - m_{H^\pm}$ and $m_A - m_{H^\pm}$, as shown in the right panel of Figure 5. Theoretical constraints are independent of the specific model (type I, II, X, Y), which makes them especially useful.

Constraints on the mass planes coming from theoretical observables are complementary to the oblique STU parameters described in Section 4.1 (see left panel of Figure 5). Results coming from the Higgs observables (Section 4.2) are of special interest since they can provide us with direct bounds on the alignment angle $\beta - \alpha$. Lastly, flavour observables described in Section 4.3 provide bounds on the Yukawa couplings (see Table 2), which depend on the quantity $\tan\beta$ when written in the physical basis.

6 Installation

The installation of HEPfit requires the availability of CMake in the system. A description of CMake and the details of how to install it can be found in the CMake website. Most package managers for Linux distributions should have a CMake package available for installation. For Mac users, it can be either installed from source or from a Unix port like Darwin Ports or Fink, or the installation package can be downloaded from the CMake website. We list below the dependencies that need to be satisfied to successfully install HEPfit:

- **GSL:** The GNU Scientific Library (GSL) is a C library for numerical computations. It can be found on the GSL website. Most Linux package managers will have a stable version as will any ports for Mac. HEPfit is compatible with GSL v1.16 or greater.

- **ROOT v5 or greater:** ROOT is an object oriented data analysis framework. You can obtain it from the ROOT website. BAT requires ROOT v5.34.19 or greater. Both HEPfit and BAT are compatible with ROOT v6. **NOTE:** If GSL is installed before compiling ROOT from source, then ROOT builds by default the MathMore library, which depends on GSL. Hence it is recommended to install GSL before installing ROOT.

- **BOOST:** BOOST is a C++ library which can be obtained from the BOOST website or from Linux package managers or Mac ports. HEPfit only requires the BOOST headers, not the full libraries, so a header-only installation is sufficient. HEPfit has been tested to work with BOOST v1.53 and greater.

- **MPI:** Optionally, HEPfit can be compiled with MPI for usage in parallelized clusters and processors supporting multi-threading. In this case, the HEPfit installer will patch and compile BAT with MPI support as described below. To this purpose one needs OpenMPI which is also available through package managers in Linux and ports on Mac.

- **BAT v1.0 (not required for the Library mode):** The BAT website offers the source code for BAT but it should *not* be used with HEPfit since a patch is required to integrate BAT with HEPfit. With the compilation option -DBAT_INSTALL=ON explained below, the HEPfit installation package will download, patch and install BAT. The parallelized version of BAT compatible with the parallelized version of HEPfit can be installed with the additional option -DMPIBAT=ON for which MPI must be installed (see "MPI Support" below).

6.1 Installation procedure

Quick Installation Instructions:

In a nutshell, if all dependencies are satisfied, for a fully MPI compatible MCMC capable `HEPfit` version `x.y` installation from the tarball downloaded from the HEPfit website:

```
$ tar xvzf HEPfit-x.y.tar.gz
$ mkdir HEPfit-x.y/build
$ cd HEPfit-x.y/build
$ cmake .. -DLOCAL_INSTALL_ALL=ON -DMPIBAT=ON
$ make
$ make install
```

To run your first example:

```
$ cd examples/MonteCarloMode/
$ make
$ mpiexec -n 5 ./analysis ../config/StandardModel.conf MonteCarlo.conf
```

This is all you need for running a MCMC simulation on 5 cores with the model, parameters and observables specified in the configuration files in `examples/config` directory with `HEPfit`. For variations please read what follows.

Detailed Installation Instructions

Unpack the tarball containing the `HEPfit` version `x.y` source which you can obtain from the HEPfit website. A directory called `HEPfit-x.y` will be created containing the source code. To generate Makefiles, enter the source directory and run `CMake`:

```
$ cd HEPfit-x.y
$ cmake . <options>
```

(RECOMMENDED:) Alternatively, a directory separate from the source directory can be made for building `HEPfit` (recommended as it allows for easy deletion of the build):

```
$ mkdir HEPfit-x.y/build
$ cd HEPfit-x.y/build
$ cmake .. <options>
```

where the available options are:

- `-DLOCAL_INSTALL_ALL=ON`: to install `BAT` and `HEPfit` in the current directory (default: OFF). This is equivalent to setting the combination of the options:

  ```
  -DCMAKE_INSTALL_PREFIX=./HEPfit -DBAT_INSTALL_DIR=./BAT -DBAT_INSTALL=ON
  ```

  These variables cannot be modified individually when `-DLOCAL_INSTALL_ALL=ON` is set.

- `-DCMAKE_INSTALL_PREFIX=<HEPfit installation directory>`: the directory in which `HEPfit` will be installed (default: `/usr/local`).

- `-DNOMCMC=ON`: to enable the mode without MCMC (default: OFF).

- `-DDEBUG_MODE=ON`: to enable the debug mode (default: OFF).

- -DBAT_INSTALL_DIR=<BAT installation directory>: (default: /usr/local). This option is overridden by -DLOCAL_INSTALL_ALL=ON .

- -DBAT_INSTALL=ON to download and install BAT. This is relevant only if -DNOMCMC=ON is not set. Use -DBAT_INSTALL=OFF only if you know your BAT installation is already patched by HEPfit and is with or without MPI support as needed. (default: ON).

- -DMPIBAT=ON: to enable support for MPI for both BAT and HEPfit (requires an implementation of MPI, default: OFF).

- -DMPI_CXX_COMPILER=<path to mpi>/mpicxx: You can specify the MPI compiler with this option.

- -DBOOST_INCLUDE_DIR=<boost custom include path>/boost/: if BOOST is not installed in the search path then you can specify where it is with this option. The path must end with the boost/ directory which contains the headers.

- -DGSL_CONFIG_DIR=<path to gsl-config>: HEPfit used gsl-config to get the GSL parameters. If this is not in the search path, you can specify it with this option.

- -DROOT_CONFIG_DIR=<path to root-config>: HEPfit used root-config to get the ROOT parameters. If this is not in the search path, you can specify it with this option.

- -DINTEL_FORTRAN=ON: If you are compiling with INTEL compilers then this flag turns on support for the compilers (default: OFF).

Setting the option -DBAT_INSTALL=ON, the HEPfit installer will download, compile and install the BAT libraries.

NOTE: If BAT libraries and headers are present in target directory for BAT they will be overwritten unless -DBAT_INSTALL=OFF is set. This is done so that the correct patched version of BAT compatible with HEPfit gets installed. **No MCMC mode:** The generated Makefiles are used for building a HEPfit library. If you do not perform a Bayesian statistical analysis with the MCMC, you can use the option -DNOMCMC=ON. In this case, BAT is not required.

MPI Support: If you want to perform an MCMC run with MPI support, you can specify the option -DMPIBAT=ON. This option must not be accompanied with -DBAT_INSTALL=OFF in order to enable the HEPfit installer to download, patch and compile BAT and build HEPfit with MPI support:

```
$ cmake . -DMPIBAT=ON <other options>
```

ROOT: CMake checks for ROOT availability in the system and fails if ROOT is not installed. You can specify the path to root-config using the option -DROOT_CONFIG_DIR=<path to root-config>.

BOOST: CMake also checks for BOOST headers availability in the system and fails if BOOST headers are not installed. You can specify the path to the BOOST include files with -DBOOST_INCLUDE_DIR=<boost custom include path>/boost/.

The **recommended installation** flags for a locally installed `HEPfit` with full MPI and MCMC support is:

```
$ cmake . -DLOCAL_INSTALL_ALL=ON -DMPIBAT=ON
```

This will enable easy portability of all codes and easy upgrading to future version as nothing will be installed system wide. Also, this is useful if you do not have root access and cannot install software in system folders. After successful `CMake` run, execute the build commands:

```
$ make
$ make install
```

to compile and install `HEPfit`, where the command `make VERBOSE=1` enables verbose output and `make -j` allows for parallel compilation. Note that depending on the setting of installation prefix you might need root privileges to be able to install `HEPfit` with `sudo make install` instead of just `make install`.

6.2   Post installation

After the completion fo the installation with `make install` the following three files can be found in the installation location. The file `libHEPfit.h` is a combined header file corresponding to the library `libHEPfit.a`.

Executable: `<CMAKE_INSTALL_PREFIX>/bin/hepfit-config`

Library: `<CMAKE_INSTALL_PREFIX>/lib/libHEPfit.a`

Combined Header: `<CMAKE_INSTALL_PREFIX>/include/HEPfit/HEPfit.h`

Using hepfit-config: A hepfit-config script can be found in the `<CMAKE_INSTALL_PREFIX>/bin/` directory, which can be invoked with the following options:

- `--cflags` to obtain the include path needed for compilation against the `HEPfit` library.

- `--libs` to obtain the flags needed for linking against the `HEPfit` library.

Examples: The example programs can be found in the `HEPfit` build directory:

- `examples/LibMode_config/`

- `examples/LibMode_header/`

- `examples/MonteCarloMode/`

- `examples/EventGeneration/`

- `examples/myModel/`

The first two demonstrate the usage of the `HEPfit` library, while the third one can be used for testing a Monte Carlo run with the `HEPfit` executable. The fourth example can be used to generate values of observables with a sample of parameters drawn from the parameter space. The fifth one is an example implementation of a custom model and custom observables. To make an executable to run these examples:

```
$ cd examples/MonteCarloMode/
$ make
```

This will produce an executable called `analysis` in the current directory that can be used to run `HEPfit`. The details are elaborated on in the next section.

7   Usage and examples

After the `HEPfit` installer generates the library `libHEPFit.a` along with header files included in a combined header file, `HEPfit.h`, the given example implementation can be used to perform a MCMC based Bayesian statistical analysis. Alternatively, the library can be used to obtain predictions of observables for a given point in the parameter space of a model, allowing `HEPfit` to be called from the user's own program. We explain both methods below. In addition `HEPfit` provides the ability to the user to define custom models and observables as explained in 2.3. We give a brief description on how to get started with custom models and observables.

7.1   Monte Carlo mode

The Monte Carlo analysis is performed with the `BAT` library. First, a text configuration file (or a set of files) containing a list of model parameters, model flags and observables to be analyzed has to be prepared. Another configuration file for the Monte Carlo run has to be prepared, too.

Step 1: Model configuration file

The configuration files are the primary way to control the behaviour of the code and to detail its input and output. While a lot of checks have been implemented in `HEPfit` to make sure the configuration files are of the right format, it is not possible to make it error-proof. Hence, care should be taken in preparing these files. A configuration file for model parameters, model flags, and observables is written as follows:

```
1  StandardModel
2  # Model parameters:
3  ModelParameter   mtop          173.2       0.9         0.
4  ModelParameter   mHl           125.6       0.3         0.
5  ...
6  CorrelatedGaussianParameters    V1_lattice 2
7  ModelParameter   a_0V    0.496    0.067    0.
8  ModelParameter   a_1V    -2.03    0.92     0.
9  1.00     0.86
10 0.86     1.00
11
12 <All the model parameters have to be listed here>
13
14 # Observables:
15 Observable   Mw        Mw        M_{W}      80.3290 80.4064 MCMC  weight 80.385
       0.015 0.
16 Observable   GammaW    GammaW    #Gamma_{W} 2.08569 2.09249 MCMC  weight 2.085
       0.042 0.
17 #
```

```
18  # Correlated observables:
19  CorrelatedGaussianObservables Zpole2 7
20  Observable   Alepton    Alepton    A_{l}        0.143568 0.151850 MCMC weight 0.1513
           0.0021  0.
21  Observable   Rbottom    Rbottom    R_{b}        0.215602 0.215958 MCMC weight 0.21629
           0.00066 0.
22  Observable   Rcharm     Rcharm     R_{c}        0.172143 0.172334 MCMC weight 0.1721
           0.0030  0.
23  Observable   AFBbottom AFBbottom A_{FB}^{b} 0.100604 0.106484 MCMC weight 0.0992
           0.0016  0.
24  Observable   AFBcharm   AFBcharm   A_{FB}^{c} 0.071750 0.076305 MCMC weight 0.0707
           0.0035  0.
25  Observable   Abottom    Abottom    A_{b}        0.934320 0.935007 MCMC weight 0.923
           0.020   0.
26  Observable   Acharm     Acharm     A_{c}        0.666374 0.670015 MCMC weight 0.670
           0.027   0.
27  1.00    0.00    0.00    0.00    0.00    0.09    0.05
28  0.00    1.00   -0.18   -0.10    0.07   -0.08    0.04
29  0.00   -0.18    1.00    0.04   -0.06    0.04   -0.06
30  0.00   -0.10    0.04    1.00    0.15    0.06    0.01
31  0.00    0.07   -0.06    0.15    1.00   -0.02    0.04
32  0.09   -0.08    0.04    0.06   -0.02    1.00    0.11
33  0.05    0.04   -0.06    0.01    0.04    0.11    1.00
34  #
35  # Output correlations:
36  Observable2D  MwvsGammaW Mw M_{W} 80.3290 80.4064 noMCMC noweight GammaW #Gamma_{W
           } 2.08569 2.09249
37  ...
38  Observable2D Bd_Bsbar_mumu noMCMC noweight
39  Observable   BR_Bdmumu      BR(B_{d}#rightarrow#mu#mu)    1. -1.  1.05e-10    0.
           0.
40  Observable   BRbar_Bsmumu   BR(B_{s}#rightarrow#mu#mu)    1. -1.  3.65e-9     0.
           0.
41  ...
42  Observable2D S5_P5 noMCMC noweight
43  BinnedObservable    S_5        S_5     1. -1.  0.   0.   0.   4.   6.
44  BinnedObservable    P_5        P_5     1. -1.  0.   0.   0.   4.   6.
45  #
46  # Including other configuration files
47  IncludeFile Flavour.conf
```

where the lines beginning with the '#' are commented out. Each line has to be written as follows:

1. The first line must be **the name of the model** to be analyzed, where the available models are listed in the HEPfit online documentation.

2. **Model flags**, if necessary, should be specified right after the model because some of them can control the way the input parameters are read.

   ```
   ModelFlag <name> <value>
   ```

3. **A model parameter** is given in the format:

   ```
   ModelParameter <name> <central value> <Gaussian error> <flat error>
   ```

where all the parameters in a given model (see the online documentation) have to be listed in the configuration file.

4. **A set of correlated model parameters** is specified with

```
CorrelatedGaussianParameters name Npar
```

which initializes a set of `Npar` correlated parameters. It must be followed by exactly `Npar` ModelParameter lines and then by `Npar` rows of `Npar` numbers for the correlation matrix. See the above example.

5. **An `Observable`** to be computed is specified in one of the following formats:

```
Observable <name> <obs label> <histolabel> <min> <max> (no)MCMC (no)weight
    <central value> <Gaussian error> <flat error>
#
Observable <name> <obs label> <histolabel> <min> <max> (no)MCMC file
    <filename> <histoname>
#
Observable <name> <obs label> <histolabel> <min> <max> noMCMC noweight
#
Observable <name> <obs label> <histolabel> <min> <max> noMCMC writeChain
```

- `<name>` is a user given name for different observables which must be unique for each observable.
- `<obs label>` is the theory label of the observable (see the online documentation).
- `<histolabel>` is used for the label of the output `ROOT` histogram, while `<min>` and `<max>` represent the range of the histogram (if `<min>` ≥ `<max>` the range of the histogram is set automatically).
- `(no)MCMC` is the flag specifying whether the observable should be included in likelihood used for the MCMC sampling.
- `(no)weight` specifies if the observable weight will be computed or not. If `weight` is specified with `noMCMC` then a chain containing the weights for the observable will be stored in the `MCout*.root` file.
- `noMCMC noweight` is the combination to be used to get a prediction for an observable.
- When the `weight` option is specified, at least one of the `<Gaussian error>` or the `<flat error>` must be nonvanishing, and the `<central value>` must of course be specified.
- When using the `file` option, a histogram in a `ROOT` file must be specified by the name of the `ROOT` file (`filename`) and then the name of the histogram (`histoname`) in the file (including, if needed, the directory).
- The `writeChain` option allows one to write all the values of the observable generated during the main run of the MCMC into the `ROOT` file.

6. **A `BinnedObservable`** is similar in construction to an `Observable` but with two extra arguments specifying the upper and lower limit of the bin:

```
1  BinnedObservable <name> <obs label> <histolabel> <min> <max> (no)MCMC
       (no)weight <central value> <Gaussian error> <flat error> <bin_min>
       <bin_max>
2  #
3  BinnedObservable <name> <obs label> <histolabel> <min> <max> (no)MCMC
       (no)weight <filename> <histoname> <bin_min> <bin_max>
4  #
5  BinnedObservable <name> <obs label> <histolabel> <min> <max> noMCMC
       writeChain 0. 0. 0. <bin_min> <bin_max>
```

Because of the order of parsing the `<central value> <Gaussian error> <flat error>` cannot be dropped out even in the `noMCMC noweight` case for a `BinnedObservable`.

7. **A `FunctionObservable`** is the same as a `BinnedObservable` but with only one extra argument that points to the value at which the function is computed:

```
1  FunctionObservable <name> <obs label> <histolabel> <min> <max> (no)MCMC
       (no)weight <central value> <Gaussian error> <flat error> <x_value>
2  #
3  FunctionObservable <name> <obs label> <histolabel> <min> <max> (no)MCMC
       (no)weight <filename> <histoname> <x_value>
4  #
5  FunctionObservable <name> <obs label> <histolabel> <min> <max> noMCMC
       writeChain 0. 0. 0. <x_value>
```

8. **An asymmetric Gaussian** constraint can be set using `AsyGausObservable`:

```
1  AsyGausObservable <name> <obs label> <histolabel> <min> <max> (no)MCMC
          (no)weight <central value> <left_error> <right_error>
```

9. **Correlations among observables** can be taken into account with the line `CorrelatedGaussianObservables name Nobs`, which initializes a set of `Nobs` correlated observables. It must be followed by exactly `Nobs` `Observable` lines and then by `Nobs` rows of `Nobs` numbers for the correlation matrix (see the above example). One can use the keywords `noMCMC` and `noweight`, instead of `MCMC` and `weight`.

```
1  CorrelatedGaussianObservables <name> Nobs
2  Observable <name> <obs label> <histolabel> <min> <max> (no)MCMC (no)weight
       <central value> <Gaussian error> <flat error>
3  Observable <name> <obs label> <histolabel> <min> <max> (no)MCMC (no)weight
       <central value> <Gaussian error> <flat error>
4  ...
5  ...
6  <Total of Nobs lines of Observables>
7  <Nobs×Nobs correlation matrix>
```

Any construction for `Observable` mentioned in item 5 of this list above can be used in a `CorrelatedGaussianObservables` set. Also, `BinnedObservables` or `FunctionObservables` can be used instead of and alongside `Observable`. If `noweight` is specified for any `Observable` then that particular `Observable` along with the corresponding row and column of the correlation matrix is excluded from the set of `CorrelatedGaussianObservables`.

In addition, the inverse covariance matrix of a set of `Nobs Observables` can be specified with the following:

```
ObservablesWithCovarianceInverse <name> Nobs
Observable <name> <obs label> <histolabel> <min> <max> MCMC weight    <
    central value> 0. 0.
Observable <name> <obs label> <histolabel> <min> <max> MCMC weight    <
    central value> 0. 0.
...
...
<Total of Nobs lines of Observables>
<Nobs×Nobs inverse covariance matrix>
```

10. **A correlation between two observables** can be obtained with any of the four following specifications:

```
Observable2D <name> <obs1 label> <histolabel1> <min1> <max1> (no)MCMC (no)
    weight <obs2 label> <histolabel2> <min2> <max2>
#
Observable2D <name> <obs1 label> <histolabel1> <min1> <max1> MCMC file <
    filename> <histoname> <obs2 label> <histolabel2> <min2> <max2>
#
Observable2D <name> (no)MCMC (no)weight
(Binned)Observable <obs label 1> <histolabel 1> <min> <max> <central value> <
    Gaussian error> <flat error> (<bin_min> <bin_max>)
(Binned)Observable <obs label 2> <histolabel 2> <min> <max> <central value> <
    Gaussian error> <flat error> (<bin_min> <bin_max>)
#
Observable2D <name> MCMC file filename histoname
(Binned)Observable <obs label 1> <histolabel 1> <min> <max> (<bin_min> <
    bin_max>)
(Binned)Observable <obs label 2> <histolabel 2> <min> <max> (<bin_min> <
    bin_max>)
```

11. **Include configuration files** with the `IncludeFile` directive. This is useful if one wants to separate the input configurations for better organization and flexibility.

Step 2: Monte Carlo configuration file:

The parameters and options of the Monte Carlo run are specified in a configuration file, separate from the one(s) for the model. Each line in the file has a pair of a label and its value, separated by space(s) or tab(s). The available parameters and options are:

`NChains`: The number of chains in the Monte Carlo run. A minimum of 5 is suggested (default). If the theory space is complicated and/or the number of parameters is large then more chains are necessary. The amount of statistics collected in the main run is proportional to the number of chains.

`PrerunMaxIter` : The maximum number of iterations that the prerun will go through (Default: 1000000). The prerun ends automatically when the chains converge (by default R<1.1, see below) and all efficiencies are adjusted. While it is not necessary for the prerun to converge for a run to

be completed, one should exercise caution if convergence is not attained.

NIterationsUpdateMax: The maximum number of iterations after which the proposal functions are updated in the pre-run and convergence is checked. (Default: 1000)

Seed: The seed can be fixed for deterministic runs. (Default: 0, corresponding to a random seed initialization)

Iterations: The number of iterations in the main run. This run is for the purpose of collecting statistics and is at the users discretion. (Default: 100000)

MinimumEfficiency: This allows setting the minimum efficiency of all the parameters to be attained in the prerun. (Default: 0.15)

MaximumEfficiency: This allows setting the maximum efficiency of all the parameters to be attained in the prerun. (Default: 0.35)

RValueForConvergence: The R-value for which convergence is considered to be attained in the prerun can be set with this flag. (Default: 1.1)

WriteParametersChains: The chains will be written in the ROOT file MCout*.root. This can be used for analyzing the performance of the chains and/or to use the sampled pdf for postprocessing. (Default: false)

FindModeWithMinuit: To find the global mode with MINUIT starting from the best fit parameters in the MCMC run. (Default: false)

RunMinuitOnly: To run a MINUIT minimization only, without running the MCMC. (Default: false)

CalculateNormalization: Whether the normalization of the posterior pdf will be calculated at the end of the Monte Carlo run. This is useful for model comparison. (Default: false)

NIterationNormalizationMC: The maximum number of iterations used to compute the normalization. (Default: 1000000)

PrintAllMarginalized: Whether all marginalized distributions will be printed in a pdf file (MonteCarlo_plots_*.pdf). (Default: true)

PrintCorrelationMatrix: Whether the parametric correlation will be printed in ParamCorrelations*.pdf and ParamCorrelations*.tex. (Default: false)

PrintKnowledgeUpdatePlots: Whether comparison between prior and posterior knowledge will be printed in a plot stored in ParamUpdate*.pdf. (Default: false)

PrintParameterPlot: Whether a summary of the parameters will be printed in ParamSum-

mary*.pdf. (Default: false)

`PrintTrianglePlot`: Whether a triangle plot of the parameters will be printed. (Default: false)

`WritePreRunData`: Whether the prerun data is written to a file. Useful to exploit a successful prerun for multiple runs. (Default: false)

`ReadPreRunData`: Whether the prerun data will be read from a previously stored prerun file. (Name of the file, default: empty)

`MultivariateProposal`: Whether the proposal function will be multivariate or uncorrelated. (Default: true)

`Histogram1DSmooth`: Sets the number of iterative smoothing of 1D histograms. (Default: 0)

`Histogram2DType`: Sets the type of 2D histograms: $1001 \rightarrow$ Lego (default), $101 \rightarrow$ Filled, $1 \rightarrow$ Contour.

`MCMCInitialPosition`: The initial distribution of chains over the parameter space. (Options: Center, RandomPrior, RandomUniform (default))

`PrintLogo`: Toggle the printing of the `HEPfit` logo on the histograms. (Default: true)

`NoHistogramLegend`: Toggle the printing of the histogram legend. (Default: false)

`PrintLoglikelihoodPlots`: Whether to print the 2D histograms for the parameters vs. loglikelihood. (Default: false)

`WriteLogLikelihoodChain`: Whether to write the value of log likelihood in a chain. (Default: false)

`Histogram2DAlpha`: Control the transparency of the 2D histograms. This does not work with all 2D histogram types. (Default: 1)

`NBinsHistogram1D`: The number of bins in the 1D histograms. (Default: 100, 0 sets default)

`NBinsHistogram2D`: The number of bins in the 2D histograms. (Default: 100, 0 sets default)

`InitialPositionAttemptLimit`: The maximum number of attempts made at getting a valid logarithm of the likelihood for all chains before the pre-run starts. (Default: 10000, 0 sets default)

`SignificantDigits` The number of significant digits appearing in the Statistics file. (Default: computed based on individual results, 0 sets default)

`HistogramBufferSize`: The memory allocated to each histogram. Also determines the num-

ber of events collected before setting automatically the histogram range. (Default: 100000)

For example, a **Monte Carlo configuration file** is written as:

```
1 NChains                  10
2 PrerunMaxIter            50000
3 Iterations               10000
4 #Seed                     1
5 PrintAllMarginalized     true
6 PrintCorrelationMatrix   true
7 PrintKnowledgeUpdatePlots false
8 PrintParameterPlot       false
9 MultivariateProposal     true
```

where a '#' can be placed at the beginning of each line to comment it out.

Step 3: Run

Library mode with MCMC: An example can be found in `examples/MonteCarloMode`

```
$ cd examples/MonteCarloMode
$ make
```

After creating the configuration files, run with the command:

```
$ ./analysis <model conf> <Monte Carlo conf>
```

Alternative: Run with MPI
`HEPfit` allows for parallel processing of the MCMC run and the observable computations. To allow for this `HEPfit`, and `BAT` have to be compiled with MPI support as explained in Section 6. The command

```
$ mpiexec -n N ./analysis <model conf> <Monte Carlo conf>
```

will launch analysis on `N` thread/cores/processors depending on the smallest processing unit of the hardware used. Our MPI implementation allows for runs on multi-threaded single processors as well as clusters with MPI support. **NOTE:** Our MPI implementation of `HEPfit` cannot be used with `BAT` compiled with the `--enable-parallel` option. It is mandatory to use the MPI patched version of `BAT` as explained in the online documentation.

Output Files:

- `log.txt`: the log file containing information on the prerun and on the run.

- `MCout.root`: the `ROOT` file containing all the information of the run and the histograms (and possibly the corresponding chains).

- `MonteCarlo_results.txt`: a text file containing some information on the fitted parameters.

- `MonteCarlo_plots.pdf`: a file containing the histograms for the parameters.

- `Observables`: a directory containing the histograms for all the observables specified in the config file, as well as some text files.

- `Observables/HistoLog.txt`: a file containing the information on over-run and under-run during the filling of histograms.

- `Observables/Statistics.txt`: a file containing a compilation of the statistics extracted from the histograms.

Other files might be generated depending on the options specified in the Monte Carlo configuration file.

7.2 Event generation mode

Using the model configuration file used in the Monte Carlo mode, one can obtain predictions of observables. An example can be found in `examples/EventGeneration` folder:

```
$ cd examples/EventGeneration
$ make
```

After making the configuration files, run with the command:

```
$ ./analysis <model conf> <number of iterations> [output folder]
```

The `<number of iterations>` defines the number of random points in the parameter space that will be evaluated. Setting this to 0 gives the value of the observables at the central value of all the parameters. If the `[output folder]` is not specified everything is printed on the screen and no data is saved. Alternately, one can specify the output folder and the run will be saved if `<number of iterations>` > 0. The output folder can be found in `./GeneratedEvents`. The structure of the output folder is as follows.
Output folder structure:

- `CGO`: Contains any correlated Gaussian observables that might have been listed in the model configuration files.

- `Observables`: Contains any observables that might have been listed in the model configuration files.

- `Parameters`: Contains all the parameters that were varied in the model configuration files.

- `Summary.txt`: Contains a list of the model used, the parameters varied, the observables computed and the number of events generated. This can be used, for example, to access all the files from a third party program.

The parameters and the observables are stored in the respective directories in files that are named after the same. For example, the parameter `lambda` will be saved in the file `lambda.txt` in the `Parameters` folder.

7.3 Library mode without MCMC

The library mode allows for access to all the observables implemented in HEPfit without a Monte Carlo run. The users can specify a `Model` and vary `ModelParameters` according to their own algorithm and get the corresponding predictions for the observables. This is made possible through:

- a combined library: `libHEPfit.a` (installed in `HEPFIT_INSTALL_DIR/lib`).

- a combined header file: HEPfit.h (installed in HEPFIT_INSTALL_DIR/include/HEPfit).

The HEPfit library allows for two different implementations of the access algorithm.

Non-Minimal Mode:

In the non-minimal mode the user can use the `Model conf` file to pass the default value of the model parameters. The following elements must be present in the user code to define the parameters and access the observable. (For details of model parameters, observables, etc. please look up the online documentation.)

```cpp
// Include the necessary header file.
#include <HEPfit.h>

// Define the model configuration file.
std::string ModelConf = "SomeModel.conf";

// Define a map for the observables.
std::map<std::string, double> DObs;

// Define a map for the parameters to be varied.
std::map<std::string, double> DPars;

// Initialize the observables to be returned.
DObs["Mw"] = 0;
DObs["GammaZ"] = 0.;
DObs["AFBbottom"] = 0.;

// Create and object of the class ComputeObservables.
ComputeObservables CO(ModelConf, DObs);

// Vary the parameters that need to be varied in the analysis.
DPars["Mz"] = 91.1875;
DPars["AlsMz"] = 0.1184;

// Get the map of observables with the parameter values defined above.
DObs = CO.compute(DPars);
```

Minimal Mode:

In the minimal mode the user can use the default values in the `InputParameters` header file to define the default values of the model parameters, therefore not requiring any additional input files to be parsed. (For details of model name, flags, parameters, observables, etc. please look up the online documentation.)

```cpp
// Include the necessary header file.
#include <HEPfit.h>

// Define a map for the observables.
std::map<std::string, double> DObs;

// Define a map for the mandatory model parameters used during initializing a
    model.
std::map<std::string, double> DPars_IN;

// Define a map for the parameters to be varied.
std::map<std::string, double> DPars;
```

```
12
13 // Define a map for the model flags.
14 std::map<std::string, std::string> DFlags;
15
16 // Define the name of the model to be used.
17 std::string ModelName = "NPZbbbar";
18
19 // Create and object of the class InputParameters.
20 InputParameters IP;
21
22 // Read a map for the mandatory model parameters. (Default values in
       InputParameters.h)
23 DPars_IN = IP.getInputParameters(ModelName);
24
25 // Change the default values of the mandatory model parameters if necessary.
26 // This can also be done with Dpars after creating an object of ComputeObservables
27 DPars_IN["mcharm"] = 1.3;
28 DPars_IN["mub"] = 4.2;
29
30 // Initialize the observables to be returned.
31 DObs["Mw"] = 0;
32 DObs["GammaZ"] = 0.;
33 DObs["AFBbottom"] = 0.;
34
35 // Initialize the model flags to be set.
36 DFlags["NPZbbbarLR"] = "TRUE";
37
38 // Create and object of the class ComputeObservables.
39 ComputeObservables CO(ModelName, DPars_IN, DObs);
40
41 // Set the flags for the model being used, if necessary.
42 CO.setFlags(DFlags);
43
44 // Vary the parameters that need to be varied in the analysis.
45 DPars["mtop"] = 170.0;
46 DPars["mHl"] = 126.0;
47
48 // Get the map of observables with the parameter values defined above.
49 DObs = CO.compute(DPars);
```

Use of hepfit-config: A hepfit-config script can be found in the `HEPFIT_INSTALL_DIR/bin` directory, which can be invoked with the following options:

```
Library and Library Path: hepfit-config --libs

Include Path: hepfit-config --cflags
```

7.4   Custom models and observables

A very useful feature of `HEPfit` is that it allows the user to create custom models and observables. We have already provided a template that can be found in the `examples/myModel` directory which can be used as a starting point. Below we describe how to implement both custom models and custom observables.

Custom Models: The idea of a custom model is to define an additional set of parameters over and above what is defined in any model in `HEPfit`. Typically the starting point is the `StandardModel`, as in the template present in the `HEPfit` package. Going by this template in the `examples/myModel` directory, to create a model one has to define the following:

- In the `myModel.h` header file:

    1. Define the number of additional parameters:
       ```
       static const int NmyModelvars = 4;
       ```

    2. Define the variables corresponding to the parameters:
       ```
       double c1, c2, c3, c4;
       ```

    3. Define getters for all the parameters:
       ```
       double getc1() const { return c1; }
       double getc2() const { return c2; }
       double getc3() const { return c3; }
       double getc4() const { return c4; }
       ```

- In the `myModel.cpp` file:

    1. Define the names of the parameters (they can be different from the variable names):
       ```
       const std::string myModel::myModelvars[NmyModelvars] = {"c1", "c2", "c3",
           "c4"};
       ```

    2. Link the parameter name to the variable containing it for all the parameters:
       ```
       ModelParamMap.insert(std::make_pair("c1", std::cref(c1)));
       ModelParamMap.insert(std::make_pair("c2", std::cref(c2)));
       ModelParamMap.insert(std::make_pair("c3", std::cref(c3)));
       ModelParamMap.insert(std::make_pair("c4", std::cref(c4)));
       ```

    3. Link the names of the parameters to the corresponding variables in the `setParameter` method:
       ```
       if(name.compare("c1") == 0)
               c1 = value;
           else if(name.compare("c2") == 0)
               c2 = value;
           else if(name.compare("c3") == 0)
               c3 = value;
           else if(name.compare("c4") == 0)
               c4 = value;
           else
               StandardModel::setParameter(name,value);
       ```

This completes the definition of the model. One can also define flags that will control certain aspects of the model, but since this is an advanced and not so commonly used feature we will not describe it here. There is an implementation in the template for the user to follow should it be needed. Finally, the custom model needs to be added with a name to the `ModelFactory` in the main function as is done in `examples/myModel/myModel_MCMC.cpp`.

```
ModelF.addModelToFactory("myModel", boost::factory<myModel*>() );
```

Custom Observables

The definition of custom observables does not depend on having defined a custom model or not. A custom observable can be any observable that has not been defined in `HEPfit`. It can be a function of parameters already defined in a `HEPfit` model or in a custom model or a combination of the two. However, a custom observable has to be explicitly added to the `ThObsFactory` in the main function as is done in `examples/myModel/myModel_MCMC.cpp`.

```
ThObsF.addObsToFactory("BIN1", boost::bind(boost::factory<yield*>(), _1, 1) );
ThObsF.addObsToFactory("BIN2", boost::bind(boost::factory<yield*>(), _1, 2) );
ThObsF.addObsToFactory("BIN3", boost::bind(boost::factory<yield*>(), _1, 3) );
ThObsF.addObsToFactory("BIN4", boost::bind(boost::factory<yield*>(), _1, 4) );
ThObsF.addObsToFactory("BIN5", boost::bind(boost::factory<yield*>(), _1, 5) );
ThObsF.addObsToFactory("BIN6", boost::bind(boost::factory<yield*>(), _1, 6) );
ThObsF.addObsToFactory("C_3", boost::factory<C_3*>() );
ThObsF.addObsToFactory("C_4", boost::factory<C_4*>() );
```

The first 6 observables require an argument and hence needed `boost::bind`. The last two do not need an argument. The implementation of these observables can be found in `examples/myModel /src/myObservables.cpp` and the corresponding header file. In this template the `myObservables` class inherits from the `THObservable` class and the observables called `yield`, `C_3` and `C_4` inherit from the former. Passing an object of the `StandardModel` class as a reference is mandatory as is the overloading of the `computeThValue` method by the custom observables, which is used to compute the value of the observable at each iteration.

8   Summary

`HEPfit` is a multipurpose and flexible analysis framework that can be used for fitting models to experimental and theoretical constraints. It comes with the ability to use the Bayesian MCMC framework implemented in `BAT`, which is highly efficient and allows for both factorized and non-factorized priors and is integrated with `ROOT`. The key features of the `HEPfit` framework are:

- It allows for Bayesian analyses using an efficiently parallelized MCMC and for any other custom statistical analysis that the user might want to implement. This is made possible by allowing for the computation of the observables using the `HEPfit` library.

- The Bayesian analysis framework in `HEPfit` using `BAT` is parallelized with MPI and can be run on a large number of processors without a substantial increase in the overhead. This makes the use of `HEPfit` extremely scalable from desktop computers to large clusters.

- Over and above the models and observables defined in `HEPfit`, it also allows for users to define their own models and observables. This gives users the flexibility to use `HEPfit` for any model and set of observables of their choice. User-defined models can add new parameters and the observables can be functions of these parameters and/or of the parameters already defined in `HEPfit`.

HEPfit has been thoroughly tested over the years with several physics results already published. This has allowed us not only to gain confidence in the implementation of HEPfit but also to gather configuration files that are publicly available in the HEPfit repository and can be used by anyone wishing to start using HEPfit with minimal initial effort. In this article we give a summary of the structure of the code, the statistical framework used in BAT, the parallelization of the code, a brief overview of the models and observables implemented in HEPfit and an overview of the physics results that has been produced.

Eventually, this article should be also retained as an optimal starting point for installing and running HEPfit. For further technical details on the usage of HEPfit and on the structure of the code, a comprehensive online documentation is available on the HEPfit website.

Acknowledgments

References

[1] A. Caldwell, D. Kollár, and K. Kröninger, *BAT - The Bayesian analysis toolkit*, *Computer Physics Communications* **180** (Nov., 2009) 2197–2209, [arXiv:0808.2552].

[2] A. C. Caldwell, D. Kollár, and K. Kröninger, *BAT - the bayesian analysis toolkit*, *Journal of Physics: Conference Series* **219** (apr, 2010) 032013.

[3] F. Beaujean, A. Caldwell, D. Kollár, and K. Kröninger, *BAT – the bayesian analysis toolkit*, *Journal of Physics: Conference Series* **331** (dec, 2011) 072040.

[4] M. Ciuchini, E. Franco, S. Mishima, and L. Silvestrini, *Electroweak Precision Observables, New Physics and the Nature of a 126 GeV Higgs Boson*, *JHEP* **08** (2013) 106, [arXiv:1306.4644].

[5] L. Reina, J. de Blas, M. Ciuchini, E. Franco, D. Ghosh, S. Mishima, M. Pierini, and L. Silvestrini, *Precision constraints on non-standard Higgs-boson couplings with HEPfit*, *PoS* **EPS-HEP2015** (2015) 187.

[6] M. Ciuchini, M. Fedele, E. Franco, S. Mishima, A. Paul, L. Silvestrini, and M. Valli, $B \to K^* \ell^+ \ell^-$ *decays at large recoil in the Standard Model: a theoretical reappraisal*, *JHEP* **06** (2016) 116, [arXiv:1512.07157].

[7] J. de Blas, M. Ciuchini, E. Franco, S. Mishima, M. Pierini, L. Reina, and L. Silvestrini, *Electroweak precision observables and Higgs-boson signal strengths in the Standard Model and beyond: present and future*, *JHEP* **12** (2016) 135, [arXiv:1608.01509].

[8] V. Cacchio, D. Chowdhury, O. Eberhardt, and C. W. Murphy, *Next-to-leading order unitarity fits in Two-Higgs-Doublet models with soft $\mathbb{Z}_2$ breaking*, *JHEP* **11** (2016) 026, [arXiv:1609.01290].

[9] M. Ciuchini, M. Fedele, E. Franco, S. Mishima, A. Paul, L. Silvestrini, and M. Valli, $B \to K^* \ell^+ \ell^-$ *in the Standard Model: Elaborations and Interpretations*, *PoS* **ICHEP2016** (2016) 584, [arXiv:1611.04338].

[10] J. de Blas, M. Ciuchini, E. Franco, S. Mishima, M. Pierini, L. Reina, and L. Silvestrini, *Electroweak precision constraints at present and future colliders*, *PoS* **ICHEP2016** (2017) 690, [arXiv:1611.05354].

[11] M. Ciuchini, A. M. Coutinho, M. Fedele, E. Franco, A. Paul, L. Silvestrini, and M. Valli, *On Flavourful Easter eggs for New Physics hunger and Lepton Flavour Universality violation*, *Eur. Phys. J.* **C77** (2017), no. 10 688, [arXiv:1704.05447].

[12] M. Ciuchini, M. Fedele, E. Franco, S. Mishima, A. Paul, L. Silvestrini, and M. Valli, *Knowns and Unknowns in the Predictions for $B \to K^* \mu^+ \mu^-$*, *Nucl. Part. Phys. Proc.* **285-286** (2017) 45–49.

[13] O. Eberhardt, *Two-Higgs-doublet model fits with HEPfit*, in *Proceedings, 2017 European Physical Society Conference on High Energy Physics (EPS-HEP 2017): Venice, Italy, July 5-12, 2017*, 2017. arXiv:1709.09414.

[14] S. Gori, C. Grojean, A. Juste, and A. Paul, *Heavy Higgs Searches: Flavour Matters*, *JHEP* **01** (2018) 108, [arXiv:1710.03752].

[15] J. de Blas, M. Ciuchini, E. Franco, S. Mishima, M. Pierini, L. Reina, and L. Silvestrini, *The Global Electroweak and Higgs Fits in the LHC era*, *PoS* **EPS-HEP2017** (2017) 467, [arXiv:1710.05402].

[16] D. Chowdhury and O. Eberhardt, *Update of Global Two-Higgs-Doublet Model Fits*, *JHEP* **05** (2018) 161, [arXiv:1711.02095].

[17] J. de Blas, O. Eberhardt, and C. Krause, *Current and Future Constraints on Higgs Couplings in the Nonlinear Effective Theory*, *JHEP* **07** (2018) 048, [arXiv:1803.00939].

[18] C.-W. Chiang, G. Cottin, and O. Eberhardt, *Global fits in the Georgi-Machacek model*, *Phys. Rev.* **D99** (2019), no. 1 015001, [arXiv:1807.10660].

[19] L. Cheng, O. Eberhardt, and C. W. Murphy, *Novel theoretical constraints for color-octet scalar models*, *Chin. Phys.* **C43** (2019), no. 9 093101, [arXiv:1808.05824].

[20] O. Eberhardt, *Current status of Two-Higgs-Doublet models with a softly broken $\mathbb{Z}_2$ symmetry*, in *39th International Conference on High Energy Physics (ICHEP 2018) Seoul, Gangnam-Gu, Korea, Republic of, July 4-11, 2018*, 2018. arXiv:1809.04851.

[21] M. Ciuchini, A. M. Coutinho, M. Fedele, E. Franco, A. Paul, L. Silvestrini, and M. Valli, *Hadronic uncertainties in the $B \to K^* \mu^+ \mu^-$ decay*, PoS **BEAUTY2018** (2018) 044, [arXiv:1809.03789].

[22] M. Ciuchini, A. M. Coutinho, M. Fedele, E. Franco, A. Paul, L. Silvestrini, and M. Valli, *On hadronic uncertainties polluting the New Physics hunt in $b \to s$ transitions*, Nucl. Part. Phys. Proc. **303-305** (2018) 8–13.

[23] F. Buccella, A. Paul, and P. Santorelli, *$SU(3)_F$ breaking through final state interactions and CP asymmetries in $D \to PP$ decays*, Phys. Rev. **D99** (2019), no. 11 113001, [arXiv:1902.05564].

[24] M. Ciuchini, A. M. Coutinho, M. Fedele, E. Franco, A. Paul, L. Silvestrini, and M. Valli, *New Physics in $b \to s \ell^+ \ell^-$ confronts new data on Lepton Universality*, Eur. Phys. J. **C79** (2019), no. 8 719, [arXiv:1903.09632].

[25] P. Arnan, A. Crivellin, M. Fedele, and F. Mescia, *Generic loop effects of new scalars and fermions in $b \to s \ell^+ \ell^-$ and a vector-like 4th generation*, JHEP **06** (2019) 118, [arXiv:1904.05890].

[26] D. Bečirević, M. Fedele, I. Nišandžić, and A. Tayduganov, *Lepton Flavor Universality tests through angular observables of $\overline{B} \to D^{(*)} \ell \overline{\nu}$ decay modes*, arXiv:1907.02257.

[27] J. de Blas et al., *Higgs Boson Studies at Future Particle Colliders*, arXiv:1905.03764.

[28] J. de Blas, G. Durieux, C. Grojean, J. Gu, and A. Paul, *On the future of Higgs, electroweak and diboson measurements at lepton colliders*, arXiv:1907.04311.

[29] G. Durieux, A. Irles, V. Miralles, A. Peñuelas, R. Pöschl, M. Perelló, and M. Vos, *The electro-weak couplings of the top and bottom quarks - global fit and future prospects*, arXiv:1907.10619.

[30] **HEPfit** Collaboration. http://hepfit.roma1.infn.it.

[31] A. Gelman and D. B. Rubin, *Inference from iterative simulation using multiple sequences*, Statistical Science **7** (1992), no. 4 457–472.

[32] S. P. Brooks and A. Gelman, *General methods for monitoring convergence of iterative simulations*, Journal of Computational and Graphical Statistics **7** (1998), no. 4 434–455, [https://www.tandfonline.com/doi/pdf/10.1080/10618600.1998.10474787].

[33] T. Lee, *A Theory of Spontaneous T Violation*, Phys.Rev. **D8** (1973) 1226–1239.

[34] J. F. Gunion and H. E. Haber, *The CP conserving two Higgs doublet model: The Approach to the decoupling limit*, Phys. Rev. **D67** (2003) 075019, [hep-ph/0207010].

[35] 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].

[36] D. Chowdhury and O. Eberhardt, *Global fits of the two-loop renormalized Two-Higgs-Doublet model with soft Z_2 breaking*, JHEP **11** (2015) 052, [arXiv:1503.08216].

[37] A. I. Bochkarev, S. V. Kuzmin, and M. E. Shaposhnikov, *Electroweak baryogenesis and the Higgs boson mass problem*, Phys. Lett. **B244** (1990) 275–278.

[38] A. E. Nelson, D. B. Kaplan, and A. G. Cohen, *Why there is something rather than nothing: Matter from weak interactions*, Nucl. Phys. **B373** (1992) 453–478.

[39] G. C. Dorsch, S. J. Huber, and J. M. No, *A strong electroweak phase transition in the 2HDM after LHC8*, JHEP **10** (2013) 029, [arXiv:1305.6610].

[40] H. Georgi and M. Machacek, *Doubly Charged Higgs Bosons*, Nucl. Phys. **B262** (1985) 463–477.

[41] M. S. Chanowitz and M. Golden, *Higgs boson triplets with $M_W = M_Z \cos\theta_W$*, *Phys. Lett.* **165B** (1985) 105–108.

[42] G. Altarelli and R. Barbieri, *Vacuum polarization effects of new physics on electroweak processes*, *Phys.Lett.* **B253** (1991) 161–167.

[43] G. Altarelli, R. Barbieri, and S. Jadach, *Toward a model independent analysis of electroweak data*, *Nucl.Phys.* **B369** (1992) 3–32 ; ibid. **B376** (1992) 444.

[44] G. Altarelli, R. Barbieri, and F. Caravaglios, *Nonstandard analysis of electroweak precision data*, *Nucl.Phys.* **B405** (1993) 3–23.

[45] M. E. Peskin and T. Takeuchi, *Estimation of oblique electroweak corrections*, *Phys.Rev.* **D46** (1992) 381–409.

[46] R. Barbieri, A. Pomarol, R. Rattazzi, and A. Strumia, *Electroweak symmetry breaking after LEP1 and LEP2*, *Nucl.Phys.* **B703** (2004) 127–146, [`hep-ph/0405040`].

[47] F. del Aguila, M. Pérez-Victoria, and J. Santiago, *Observable contributions of new exotic quarks to quark mixing*, *JHEP* **09** (2000) 011, [`hep-ph/0007316`].

[48] F. del Aguila, J. de Blas, and M. Pérez-Victoria, *Effects of new leptons in Electroweak Precision Data*, *Phys. Rev.* **D78** (2008) 013010, [`arXiv:0803.4008`].

[49] F. del Aguila, J. de Blas, and M. Pérez-Victoria, *Electroweak Limits on General New Vector Bosons*, *JHEP* **09** (2010) 033, [`arXiv:1005.3998`].

[50] J. de Blas, M. Chala, M. Pérez-Victoria, and J. Santiago, *Observable Effects of General New Scalar Particles*, *JHEP* **04** (2015) 078, [`arXiv:1412.8480`].

[51] A. Drozd, J. Ellis, J. Quevillon, and T. You, *The Universal One-Loop Effective Action*, *JHEP* **03** (2016) 180, [`arXiv:1512.03003`].

[52] F. del Aguila, Z. Kunszt, and J. Santiago, *One-loop effective lagrangians after matching*, *Eur. Phys. J.* **C76** (2016), no. 5 244, [`arXiv:1602.00126`].

[53] B. Henning, X. Lu, and H. Murayama, *One-loop Matching and Running with Covariant Derivative Expansion*, *JHEP* **01** (2018) 123, [`arXiv:1604.01019`].

[54] S. A. R. Ellis, J. Quevillon, T. You, and Z. Zhang, *Mixed heavy-light matching in the Universal One-Loop Effective Action*, *Phys. Lett.* **B762** (2016) 166–176, [`arXiv:1604.02445`].

[55] J. Fuentes-Martin, J. Portoles, and P. Ruiz-Femenia, *Integrating out heavy particles with functional methods: a simplified framework*, *JHEP* **09** (2016) 156, [`arXiv:1607.02142`].

[56] Z. Zhang, *Covariant diagrams for one-loop matching*, *JHEP* **05** (2017) 152, [`arXiv:1610.00710`].

[57] S. A. R. Ellis, J. Quevillon, T. You, and Z. Zhang, *Extending the Universal One-Loop Effective Action: Heavy-Light Coefficients*, *JHEP* **08** (2017) 054, [`arXiv:1706.07765`].

[58] J. D. Wells and Z. Zhang, *Effective field theory approach to trans-TeV supersymmetry: covariant matching, Yukawa unification and Higgs couplings*, *JHEP* **05** (2018) 182, [`arXiv:1711.04774`].

[59] J. de Blas, J. C. Criado, M. Perez-Victoria, and J. Santiago, *Effective description of general extensions of the Standard Model: the complete tree-level dictionary*, *JHEP* **03** (2018) 109, [`arXiv:1711.10391`].

[60] B. Grzadkowski, M. Iskrzynski, M. Misiak, and J. Rosiek, *Dimension-Six Terms in the Standard Model Lagrangian*, *JHEP* **10** (2010) 085, [`arXiv:1008.4884`].

[61] R. Contino, C. Grojean, M. Moretti, F. Piccinini, and R. Rattazzi, *Strong double Higgs production at the LHC*, *JHEP* **1005** (2010) 089, [`arXiv:1002.1011`].

[62] A. Azatov, R. Contino, and J. Galloway, *Model-independent bounds on a light Higgs*, *JHEP* **1204** (2012) 127, [arXiv:1202.3415].

[63] **LHC Higgs Cross Section Working Group** Collaboration, A. David, A. Denner, M. Duehrssen, M. Grazzini, C. Grojean, G. Passarino, M. Schumacher, M. Spira, G. Weiglein, and M. Zanetti, *LHC HXSWG interim recommendations to explore the coupling structure of a Higgs-like particle*, arXiv:1209.0040.

[64] A. Sirlin, *Radiative corrections in the $SU(2)_L \times U(1)$ theory: A simple renormalization framework*, *Phys.Rev.* **D22** (1980) 971–981.

[65] W. Marciano and A. Sirlin, *Radiative corrections to neutrino induced neutral current phenomena in the $SU(2)_L \times U(1)$ theory*, *Phys.Rev.* **D22** (1980) 2695; ibid. **D31** (1985) 213.

[66] A. Djouadi and C. Verzegnassi, *Virtual very heavy top effects in LEP/SLC precision measurements*, *Phys.Lett.* **B195** (1987) 265.

[67] A. Djouadi, $\mathcal{O}(\alpha\alpha_s)$ *vacuum polarization functions of the standard model gauge bosons*, *Nuovo Cim.* **A100** (1988) 357.

[68] B. A. Kniehl, *Two-loop corrections to the vacuum polarizations in perturbative QCD*, *Nucl.Phys.* **B347** (1990) 86–104.

[69] F. Halzen and B. A. Kniehl, Δr *beyond one loop*, *Nucl.Phys.* **B353** (1991) 567–590.

[70] B. A. Kniehl and A. Sirlin, *Dispersion relations for vacuum polarization functions in electroweak physics*, *Nucl.Phys.* **B371** (1992) 141–148.

[71] B. A. Kniehl and A. Sirlin, *On the effect of the $t\bar{t}$ threshold on electroweak parameters*, *Phys.Rev.* **D47** (1993) 883–893.

[72] R. Barbieri, M. Beccaria, P. Ciafaloni, G. Curci, and A. Vicere, *Radiative correction effects of a very heavy top*, *Phys.Lett.* **B288** (1992) 95–98, [hep-ph/9205238].

[73] R. Barbieri, M. Beccaria, P. Ciafaloni, G. Curci, and A. Vicere, *Two-loop heavy top effects in the standard model*, *Nucl.Phys.* **B409** (1993) 105–127.

[74] A. Djouadi and P. Gambino, *Electroweak gauge bosons selfenergies: Complete QCD corrections*, *Phys.Rev.* **D49** (1994) 3499–3511; ibid. **D53** (1996) 4111, [hep-ph/9309298].

[75] J. Fleischer, O. Tarasov, and F. Jegerlehner, *Two-loop heavy top corrections to the ρ parameter: A simple formula valid for arbitrary Higgs mass*, *Phys.Lett.* **B319** (1993) 249–256.

[76] J. Fleischer, O. Tarasov, and F. Jegerlehner, *Two-loop large top mass corrections to electroweak parameters: Analytic results valid for arbitrary Higgs mass*, *Phys.Rev.* **D51** (1995) 3820–3837.

[77] L. Avdeev, J. Fleischer, S. Mikhailov, and O. Tarasov, $\mathcal{O}(\alpha\alpha_s^2)$ *correction to the electroweak ρ parameter*, *Phys.Lett.* **B336** (1994) 560–566; ibid. **B349** (1995) 597-598, [hep-ph/9406363].

[78] K. Chetyrkin, J. H. Kuhn, and M. Steinhauser, *Corrections of order $\mathcal{O}(G_F M_t^2 \alpha_s^2)$ to the ρ parameter*, *Phys.Lett.* **B351** (1995) 331–338, [hep-ph/9502291].

[79] K. Chetyrkin, J. H. Kuhn, and M. Steinhauser, *QCD corrections from top quark to relations between electroweak parameters to order α_s^2*, *Phys.Rev.Lett.* **75** (1995) 3394–3397, [hep-ph/9504413].

[80] G. Degrassi, P. Gambino, and A. Vicini, *Two-loop heavy top effects on the m_Z-m_W interdependence*, *Phys.Lett.* **B383** (1996) 219–226, [hep-ph/9603374].

[81] G. Degrassi, P. Gambino, and A. Sirlin, *Precise calculation of M_W, $\sin^2 \hat{\theta}_W(M_Z)$, and $\sin^2 \theta_{\text{eff}}^{\text{lept}}$*, *Phys.Lett.* **B394** (1997) 188–194, [hep-ph/9611363].

[82] G. Degrassi and P. Gambino, *Two-loop heavy top corrections to the Z^0 boson partial widths*, *Nucl.Phys.* **B567** (2000) 3–31, [hep-ph/9905472].

[83] A. Freitas, W. Hollik, W. Walter, and G. Weiglein, *Complete fermionic two-loop results for the M_W-M_Z interdependence, Phys.Lett.* **B495** (2000) 338–346; ibid. **B570** (2003) 260-264, [`hep-ph/0007091`].

[84] J. van der Bij, K. Chetyrkin, M. Faisst, G. Jikia, and T. Seidensticker, *Three-loop leading top mass contributions to the ρ parameter, Phys.Lett.* **B498** (2001) 156–162, [`hep-ph/0011373`].

[85] A. Freitas, W. Hollik, W. Walter, and G. Weiglein, *Electroweak two-loop corrections to the M_W-M_Z mass correlation in the standard model, Nucl.Phys.* **B632** (2002) 189–218; ibid. **B666** (2003) 305-307, [`hep-ph/0202131`].

[86] M. Awramik and M. Czakon, *Complete two loop bosonic contributions to the muon lifetime in the standard model, Phys.Rev.Lett.* **89** (2002) 241801, [`hep-ph/0208113`].

[87] A. Onishchenko and O. Veretin, *Two-loop bosonic electroweak corrections to the muon lifetime and M_Z-M_W interdependence, Phys.Lett.* **B551** (2003) 111–114, [`hep-ph/0209010`].

[88] M. Awramik, M. Czakon, A. Onishchenko, and O. Veretin, *Bosonic corrections to Δr at the two loop level, Phys.Rev.* **D68** (2003) 053004, [`hep-ph/0209084`].

[89] M. Awramik and M. Czakon, *Two loop electroweak bosonic corrections to the muon decay lifetime, Nucl.Phys.Proc.Suppl.* **116** (2003) 238–242, [`hep-ph/0211041`].

[90] M. Awramik and M. Czakon, *Complete two loop electroweak contributions to the muon lifetime in the standard model, Phys.Lett.* **B568** (2003) 48–54, [`hep-ph/0305248`].

[91] M. Awramik, M. Czakon, A. Freitas, and G. Weiglein, *Precise prediction for the W boson mass in the standard model, Phys.Rev.* **D69** (2004) 053006, [`hep-ph/0311148`].

[92] M. Faisst, J. H. Kuhn, T. Seidensticker, and O. Veretin, *Three loop top quark contributions to the rho parameter, Nucl. Phys.* **B665** (2003) 649–662, [`hep-ph/0302275`].

[93] I. Dubovyk, A. Freitas, J. Gluza, T. Riemann, and J. Usovitsch, *The two-loop electroweak bosonic corrections to $\sin^2 \theta_{\mathrm{eff}}^{\mathrm{b}}$, Phys. Lett.* **B762** (2016) 184–189, [`arXiv:1607.08375`].

[94] I. Dubovyk, A. Freitas, J. Gluza, T. Riemann, and J. Usovitsch, *Complete electroweak two-loop corrections to Z boson production and decay, Phys. Lett.* **B783** (2018) 86–94, [`arXiv:1804.10236`].

[95] D. Kennedy and B. Lynn, *Electroweak radiative corrections with an effective Lagrangian: Four fermion processes, Nucl.Phys.* **B322** (1989) 1–54.

[96] D. Kennedy, B. Lynn, C. Im, and R. Stuart, *Electroweak cross-sections and asymmetries at the Z^0, Nucl.Phys.* **B321** (1989) 83.

[97] M. E. Peskin and T. Takeuchi, *A new constraint on a strongly interacting Higgs sector, Phys.Rev.Lett.* **65** (1990) 964–967.

[98] P. Bamert, C. Burgess, J. M. Cline, D. London, and E. Nardi, *R_b and new physics: A comprehensive analysis, Phys.Rev.* **D54** (1996) 4275–4300, [`hep-ph/9602438`].

[99] H. E. Haber and H. E. Logan, *Radiative corrections to the $Zb\bar{b}$ vertex and constraints on extended Higgs sectors, Phys.Rev.* **D62** (2000) 015011, [`hep-ph/9909335`].

[100] D. Choudhury, T. M. Tait, and C. Wagner, *Beautiful mirrors and precision electroweak data, Phys.Rev.* **D65** (2002) 053002, [`hep-ph/0109097`].

[101] K. Agashe, R. Contino, L. Da Rold, and A. Pomarol, *A custodial symmetry for $Zb\bar{b}$, Phys.Lett.* **B641** (2006) 62–66, [`hep-ph/0605341`].

[102] A. Djouadi, G. Moreau, and F. Richard, *Resolving the $A_{\mathrm{FB}}^{\mathrm{b}}$ puzzle in an extra dimensional model with an extended gauge structure, Nucl.Phys.* **B773** (2007) 43–64, [`hep-ph/0610173`].

[103] K. Kumar, W. Shepherd, T. M. Tait, and R. Vega-Morales, *Beautiful mirrors at the LHC*, *JHEP* **1008** (2010) 052, [arXiv:1004.4895].

[104] L. Da Rold, *Solving the A_{FB}^b anomaly in natural composite models*, *JHEP* **1102** (2011) 034, [arXiv:1009.2392].

[105] E. Alvarez, L. Da Rold, and A. Szynkman, *A composite Higgs model analysis of forward-backward asymmetries in the production of tops at Tevatron and bottoms at LEP and SLC*, *JHEP* **1105** (2011) 070, [arXiv:1011.6557].

[106] R. Dermisek, S.-G. Kim, and A. Raval, *New vector boson near the Z-pole and the puzzle in precision electroweak data*, *Phys.Rev.* **D84** (2011) 035006, [arXiv:1105.0773].

[107] A. Djouadi, G. Moreau, and F. Richard, *Forward-backward asymmetries of the bottom and top quarks in warped extra-dimensional models: LHC predictions from the LEP and Tevatron anomalies*, *Phys.Lett.* **B701** (2011) 458–464, [arXiv:1105.3158].

[108] R. Dermisek, S.-G. Kim, and A. Raval, *Z' near the Z-pole*, *Phys.Rev.* **D85** (2012) 075022, [arXiv:1201.0315].

[109] B. Batell, S. Gori, and L.-T. Wang, *Higgs couplings and precision electroweak data*, *JHEP* **1301** (2013) 139, [arXiv:1209.6382].

[110] D. Guadagnoli and G. Isidori, *$B(B_s \to \mu^+\mu^-)$ as an electroweak precision test*, *Phys. Lett.* **B724** (2013) 63–67, [arXiv:1302.3909].

[111] W. Buchmuller and D. Wyler, *Effective Lagrangian analysis of new interactions and flavor conservation*, *Nucl.Phys.* **B268** (1986) 621.

[112] **Belle II** Collaboration, E. Kou et al., *The Belle II Physics Book* , arXiv:1808.10567.

[113] S. Descotes-Genon, A. Falkowski, M. Fedele, M. González-Alonso, and J. Virto, *The CKM parameters in the SMEFT*, *JHEP* **05** (2019) 172, [arXiv:1812.08163].

[114] L. Silvestrini and M. Valli, *Model-independent Bounds on the Standard Model Effective Theory from Flavour Physics*, arXiv:1812.10913.

[115] N. G. Deshpande and E. Ma, *Pattern of Symmetry Breaking with Two Higgs Doublets*, *Phys. Rev.* **D18** (1978) 2574.

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

[117] B. Grinstein, C. W. Murphy, and P. Uttayarat, *One-loop corrections to the perturbative unitarity bounds in the CP-conserving two-Higgs doublet model with a softly broken $\mathbb{Z}_2$ symmetry*, *JHEP* **06** (2016) 070, [arXiv:1512.04567].

[118] A. Barroso, P. M. Ferreira, I. P. Ivanov, and R. Santos, *Metastability bounds on the two Higgs doublet model*, *JHEP* **06** (2013) 045, [arXiv:1303.5098].

[119] A. Arhrib, R. Benbrik, M. Chabab, G. Moultaka, M. C. Peyranere, L. Rahili, and J. Ramadan, *The Higgs Potential in the Type II Seesaw Model*, *Phys. Rev.* **D84** (2011) 095005, [arXiv:1105.1925].

[120] M. Aoki and S. Kanemura, *Unitarity bounds in the Higgs model including triplet fields with custodial symmetry*, *Phys. Rev.* **D77** (2008), no. 9 095009, [arXiv:0712.4053]. [Erratum: Phys. Rev.D89,no.5,059902(2014)].

[121] R. K. Ellis et al., *Physics Briefing Book*, arXiv:1910.11775.

[122] **HL-LHC, HE-LHC Working Group** Collaboration, P. Azzi et al., *Standard Model Physics at the HL-LHC and HE-LHC*, arXiv:1902.04070.

[123] **HL/HE WG2 group** Collaboration, M. Cepeda et al., *Higgs Physics at the HL-LHC and HE-LHC*, arXiv:1902.00134.

[124] **FCC** Collaboration, A. Abada et al., *FCC-ee: The Lepton Collider*, *Eur. Phys. J. ST* **228** (2019), no. 2 261–623.

[125] **FCC** Collaboration, A. Abada et al., *FCC Physics Opportunities*, *Eur. Phys. J.* **C79** (2019), no. 6 474.

[126] A. Khodjamirian, T. Mannel, A. A. Pivovarov, and Y. M. Wang, *Charm-loop effect in $B \to K^{(*)}\ell^+\ell^-$ and $B \to K^*\gamma$*, *JHEP* **09** (2010) 089, [arXiv:1006.4945].

[127] **LHCb** Collaboration, R. Aaij et al., *Test of lepton universality with $B^0 \to K^{*0}\ell^+\ell^-$ decays*, *JHEP* **08** (2017) 055, [arXiv:1705.05802].

[128] **LHCb** Collaboration, R. Aaij et al., *Observation of CP Violation in Charm Decays*, *Phys. Rev. Lett.* **122** (2019), no. 21 211803, [arXiv:1903.08726].

[129] **LHCb** Collaboration, F. Betti, *Observation of CP violation in charm decays at LHCb*, in *54th Rencontres de Moriond on Electroweak Interactions and Unified Theories (Moriond EW 2019) La Thuile, Italy, March 16-23, 2019*, 2019. arXiv:1905.05428.

[130] **LHCb** Collaboration, T. Pajero, *Measurements of time-dependent CP violation and mixing in charm at LHCb*, in *17th Conference on Flavor Physics and CP Violation (FPCP 2019) Victoria, BC, Canada, May 6-10, 2019*, 2019. arXiv:1907.01292.

[131] C. W. Murphy, *NLO Perturbativity Bounds on Quartic Couplings in Renormalizable Theories with ϕ^4-like Scalar Sectors*, *Phys. Rev.* **D96** (2017), no. 3 036006, [arXiv:1702.08511].

[132] D. M. Straub, *flavio: a python package for flavour and precision phenomenology in the standard model and beyond*, arXiv:1810.08132.

[133] S. Heinemeyer, W. Hollik, and G. Weiglein, *FeynHiggs: A Program for the calculation of the masses of the neutral CP even Higgs bosons in the MSSM*, *Comput. Phys. Commun.* **124** (2000) 76–89, [[hep-ph/9812320](hep-ph/9812320)].

[134] H. Bahl, T. Hahn, S. Heinemeyer, W. Hollik, S. Paßehr, H. Rzehak, and G. Weiglein, *Precision calculations in the MSSM Higgs-boson sector with FeynHiggs 2.14*, arXiv:1811.09073.

[135] F. Mahmoudi, *SuperIso v2.3: A Program for calculating flavor physics observables in Supersymmetry*, *Comput. Phys. Commun.* **180** (2009) 1579–1613, [arXiv:0808.3144].

[136] D. Chowdhury, R. Garani, and S. K. Vempati, *SUSEFLAV: Program for supersymmetric mass spectra with seesaw mechanism and rare lepton flavor violating decays*, *Comput. Phys. Commun.* **184** (2013) 899–918, [arXiv:1109.3551].

[137] H. Flacher, M. Goebel, J. Haller, A. Hocker, K. Monig, et al., *Revisiting the global electroweak fit of the standard bodel and beyond with Gfitter*, *Eur.Phys.J.* **C60** (2009) 543–583, [arXiv:0811.0009].

[138] F. Herren and M. Steinhauser, *Version 3 of RunDec and CRunDec*, *Comput. Phys. Commun.* **224** (2018) 333–345, [arXiv:1703.03751].

[139] A. Akhundov, A. Arbuzov, S. Riemann, and T. Riemann, *The ZFITTER project*, *Phys. Part. Nucl.* **45** (2014), no. 3 529–549, [arXiv:1302.1395].

[140] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, *JHEP* **07** (2014) 079, [arXiv:1405.0301].