2HDM Meeting.

Flavour Physics tools:

Part I: Flavio

Víctor Martín Lozano
victor.lozano@desy.de

Flavio.

https://flav-io.github.io/

Python based package for flavour physics and other precision tests of the SM developed by David Straub.

- Observables calculator
- Likelihoods
- Plots
- Easy to customize

Flavio.

https://flav-io.github.io/

Installation:

As it is a tool written totally in python one can install it easily doing:

```
python3 -m pip install flavio
```

And install also other dependencies:

```
pip3 install flavio[plotting]
```

Flavio.

https://flav-io.github.io/

Flavio is a tool that uses flavour physics and other precision tests in the SM to compute predictions for observables that can be sensitive to New Physics (NP).

In order to do so, it parameterised the NP in terms of Wilson coefficients of dimension six operators.

Flavio.

https://flav-io.github.io/

Observables used by Flavio:

Meson-antimeson mixing, W, Z, tau and muon lepton decays, b,c-quark decays, e+e- scattering, dipole moments, Higgs production and decay...

Flavio.

https://flav-io.github.io/

Category	Process	Observables
Meson mixing	$B^0 \leftrightarrow \bar{B}^0$	ΔM_d, a_{fs}^d
	$B_s \leftrightarrow \bar{B}_s$	ΔM_s, a_{fs}^s
	$K^0 \leftrightarrow \bar{K}^0$	$\lvert\epsilon_K\rvert$
	$D^0 \leftrightarrow \bar{D}^0$	x, y, ϕ, q/p,
		x_{12}, y_{12}, ϕ_{12}, x_{12}^{Im}
Non-lept. B decays	$B^0 \to \psi K_S$	$S_{\psi K_S}$
	$B^0 \to \psi\phi$	$S_{\psi\phi}$
Radiative B decays	$B \to X_{s,d}\gamma$	BR, A_{CP}
	$B^+ \to K^{*+}\gamma$	BR, A_{CP}
	$B^0 \to K^{*0}\gamma$	BR, $S_{K^*\gamma}$, A_{CP}
	$B_s \to \phi\gamma$	BR, $S_{\phi\gamma}$, $A_{\Delta\Gamma}$
Rare lept. B decays	$B_{s,d} \to \ell^+\ell^-$	BR, $A_{\Delta\Gamma}$
Rare SL B decays	$B^+ \to \ell^+\nu$	BR
	$B^{+,0} \to K^{*+,0}\nu\bar{\nu}$	BR, F_L
	$B^{+,0} \to K^{+,0}\nu\bar{\nu}$	BR
	$B^{+,0} \to \rho^{+,0}\nu\bar{\nu}$	BR
	$B^{+,0} \to \pi^{+,0}\nu\bar{\nu}$	BR
	$B^{+,0} \to K^{*+,0}\ell^+\ell^-$	BR, S_i, A_i, $P_i^{(\prime)}$
	$B^{+,0} \to K^{+,0}\ell^+\ell^-$	BR, F_H, A_{FB}
	$B_s \to \phi\ell^+\ell^-$	BR, F_L, S_i
	$\Lambda_b \to \Lambda\ell^+\ell^-$	BR, F_L, $A_{\text{FB}}^{\ell,h,\ell h}$

Flavio.

https://flav-io.github.io/

Category	Process	Observables
SL tree-level B dec.	$B^{0,+} \to \pi^{+,0}\ell\nu$	BR
	$B^{0,+} \to \rho^{+,0}\ell\nu$	BR
	$B^{0,+} \to D^{*+,0}\ell\nu$	BR, $\text{BR}_{L,T}$, F_L, $d\text{BR}/d\chi_i$
	$B^{0,+} \to D^{+,0}\ell\nu$	BR
	$B^- \to \omega\ell^-\bar{\nu}$	BR
	$B_s \to K^{*+}\ell^-\bar{\nu}$	BR
Lep. tree-level B dec.	$B^+ \to \ell\nu$	BR
	$B_c \to \ell\nu$	BR
Rare K decays	$K^{+,}{}_L \to \pi^{+,0}\nu\bar{\nu}$	BR
Non-leptonic K decay	$K \to \pi\pi$	ε'/ε
SL tree-level K dec.	$K^+ \to \ell^+\nu$	BR
	$K^{+,}{}_L \to \pi^{0,+}\ell^-\bar{\nu}$	BR
π decays	$\pi^+ \to e^+\nu$	BR

Flavio.

https://flav-io.github.io/

Category	Process	Observables
LFV τ decays	$\tau \to \ell\gamma$	BR
	$\tau \to 3\mu$	BR
	$\tau \to \mu ee$	BR
	$\tau \to \rho\ell$	BR
	$\tau \to \phi\ell$	BR
Tree-level τ decays	$\tau \to \ell\nu\nu$	BR
	$\tau \to K\nu$	BR
	$\tau \to \pi\nu$	BR
LFV μ decays	$\mu \to e\gamma$	BR
	$\mu \to 3e$	BR
Z prod. & decay	$Z \to f\bar{f}$	$\Gamma_Z, \Gamma_f, A, A_{\rm FB}, R_f$
	$e^+e^- \to Z \to q\bar{q}$	$\sigma_{\rm had}$
W prod. & decay	$W \to \ell\nu$	BR
	$W \to ff'$	Γ_W
	m_W	
EDMs	d_n	
MDMs	a_e, a_μ, a_τ	
other	$\nu_\mu N \to \nu_\ell \mu^+\mu^- N$	$R_{\rm trident}$

Flavio.

https://flav-io.github.io/

New Physics in Flavio:

- No NP model implemented
- NP in the form of dim-6 operators, Wilson coefficients
- Weak Effective field Theory (WET) - flavio
- Standard Model Effective Field Theory (SMEFT) - Warsaw
- RGEs, matching using Wilson Coefficient exchange format (Wcxf)
- Other basis are allowed up to a correct translation

Flavio.

https://flav-io.github.io/

New Physics in Flavio:

Basis flavio (EFT WET)

Basis used by the flavio package. Neutrinos are in the flavour basis.

Sectors

The effective Lagrangian is defined as

$$\mathcal{L}_{\text{eff}} = -\mathcal{H}_{\text{eff}} = \sum_{O_i = O_i^\dagger} C_i O_i + \sum_{O_i \neq O_i^\dagger} \left(C_i O_i + C_i^* O_i^\dagger \right).$$

sbsb

WC name	Operator	Type
CVLL_bsbs	$(\bar{s}_L \gamma^\mu b_L)(\bar{s}_L \gamma_\mu b_L)$	C
CVRR_bsbs	$(\bar{s}_R \gamma^\mu b_R)(\bar{s}_R \gamma_\mu b_R)$	C
CSLL_bsbs	$(\bar{s}_R b_L)(\bar{s}_R b_L)$	C
CSRR_bsbs	$(\bar{s}_L b_R)(\bar{s}_L b_R)$	C
CTLL_bsbs	$(\bar{s}_R \sigma^{\mu\nu} b_L)(\bar{s}_R \sigma_{\mu\nu} b_L)$	C
CTRR_bsbs	$(\bar{s}_L \sigma^{\mu\nu} b_R)(\bar{s}_L \sigma_{\mu\nu} b_R)$	C
CVLR_bsbs	$(\bar{s}_L \gamma^\mu b_L)(\bar{s}_R \gamma_\mu b_R)$	C
CSLR_bsbs	$(\bar{s}_R b_L)(\bar{s}_L b_R)$	C

Basis Warsaw (EFT SMEFT)

Basis suggested by Grzadkowski, Iskrzyński, Misiak, and Rosiek (arXiv:1008.4884v3). At variance with their definition, the Wilson coefficients are defined to be dimensionful, such that $\mathcal{L} = \sum_i C_i O_i$. The set of redundant operators coincides with the choice of DSixTools (arXiv:1704.04504). The weak basis for the fermion fields is chosen such that the running dimension-6 mass matrices of charged leptons and down-type quarks are diagonal at the scale where the coefficient values are specified, while up-type quark singlet field is rotated to diagonalise the running dimension-6 up-type quark mass matrix “from the right”.

Sectors

The effective Lagrangian is defined as

$$\mathcal{L}_{\text{eff}} = -\mathcal{H}_{\text{eff}} = \sum_{O_i = O_i^\dagger} C_i O_i + \sum_{O_i \neq O_i^\dagger} \left(C_i O_i + C_i^* O_i^\dagger \right).$$

dB=dL=0

WC name	Operator	Type
G	$f^{ABC} G_\mu^{A\nu} G_\nu^{B\rho} G_\rho^{C\mu}$	R
Gtilde	$f^{ABC} \widetilde{G}_\mu^{A\nu} G_\nu^{B\rho} G_\rho^{C\mu}$	R
W	$\epsilon^{IJK} W_\mu^{I\nu} W_\nu^{J\rho} W_\rho^{K\mu}$	R
Wtilde	$\epsilon^{IJK} \widetilde{W}_\mu^{I\nu} W_\nu^{J\rho} W_\rho^{K\mu}$	R
phi	$(\varphi^\dagger \varphi)^3$	R
phiBox	$(\varphi^\dagger \varphi) \Box (\varphi^\dagger \varphi)$	R
phiD	$(\varphi^\dagger D^\mu \varphi)^* (\varphi^\dagger D_\mu \varphi)$	R
phiG	$\varphi^\dagger \varphi G_{\mu\nu}^A G^{A\mu\nu}$	R
phiB	$\varphi^\dagger \varphi B_{\mu\nu} B^{\mu\nu}$	R
phiW	$\varphi^\dagger \varphi W_{\mu\nu}^I W^{I\mu\nu}$	R
phiWB	$\varphi^\dagger \tau^I \varphi W_{\mu\nu}^I B^{\mu\nu}$	R
phiGtilde	$\varphi^\dagger \varphi \widetilde{G}_{\mu\nu}^A G^{A\mu\nu}$	R
phiBtilde	$\varphi^\dagger \varphi \widetilde{B}_{\mu\nu} B^{\mu\nu}$	R

Flavio.

https://flav-io.github.io/

New Physics in Flavio:

NP contribution to B^0 and B_s mixing via the operator $(\bar{q}_R \gamma^\mu b_R)^2$

with $q = d, s$

```
from wilson import Wilson
w = Wilson({ 'CVRR_bdbd': 1e-10, 'CVRR_bsbs': 1e-10 }, scale=160, eft='WET', basis='flavio')
```

```
flavio.np_prediction('DeltaM_s', w)
```

Flavio.

https://flav-io.github.io/

New Physics in Flavio:

NP contribution to B^0 and B_s mixing via the operator $(\bar{q}_R \gamma^\mu b_R)^2$ with $q = d, s$

```
from wilson import Wilson
w = Wilson({ 'CVRR_bdbd': 1e-10, 'CVRR_bsbs': 1e-10 }, scale=160, eft='WET', basis='flavio')
```

Value of the Wilson coefficient at a scale

```
flavio.np_prediction('DeltaM_s', w)
```

Flavio.

https://flav-io.github.io/

New Physics in Flavio:

NP contribution to B^0 and B_s mixing via the operator $(\bar{q}_R \gamma^\mu b_R)^2$

with $q = d, s$

```
from wilson import Wilson
w = Wilson({ 'CVRR_bdbd': 1e-10, 'CVRR_bsbs': 1e-10 }, scale=160, eft='WET', basis='flavio')
```

Effective Field Theory   Basis

```
flavio.np_prediction('DeltaM_s', w)
```

Flavio.

https://flav-io.github.io/

New Physics in Flavio:

NP contribution to B^0 and B_s mixing via the operator $(\bar{q}_R \gamma^\mu b_R)^2$

with $q = d, s$

```
from wilson import Wilson
w = Wilson({ 'CVRR_bdbd': 1e-10, 'CVRR_bsbs': 1e-10 }, scale=160, eft='WET', basis='flavio')
```

Effective Field Theory

Basis

```
flavio.np_prediction('DeltaM_s', w)
```

Flavio.

https://flav-io.github.io/

New Physics in Flavio:

NP contribution to B^0 and B_s mixing via the operator $(\bar{q}_R \gamma^\mu b_R)^2$

with $q = d, s$

```
from wilson import Wilson
w = Wilson({ 'CVRR_bdbd': 1e-10, 'CVRR_bsbs': 1e-10 }, scale=160, eft='WET', basis='flavio')
```

Once the NP scenario is defined we can type np_prediction to obtain the central value of the observable

```
flavio.np_prediction('DeltaM_s', w)
```

`DeltaM_d` `DeltaM_d/DeltaM_s`

Flavio.

https://flav-io.github.io/

Default values:

Flavio allows a change in the value of the parameters.

Masses and widths are taken from PDG

One can change easily change the values but there are two kind of parameters: correlated and uncorrelated

Uncorrelated parameters: Uncertainties are not correlated (G_F, α_S, etc)

Correlated parameters: Uncertainties are correlated

```
values:                correlation:
  - m_c: 1.273(10)       [[1., 0.957, 0.933, 0.613, 0.136],
  - m_b: 4.195(14)       [0.957, 1., 0.943, 0.625, 0.181],
  - m_s: 0.09247(69)     [0.933, 0.943, 1., 0.703, 0.292],
  - m_d: 0.004675(56)    [0.613, 0.625, 0.703, 1., -0.16],
  - m_u: 0.002130(41)    [0.136, 0.181, 0.292, -0.16, 1.]]
```

Flavio.

https://flav-io.github.io/

Likelihoods:

Flavio provides the likelihood in the parameter or the Wilson coefficient space.

$$L(\mathbf{C},\boldsymbol{\theta}) = \prod_i L^i_{\rm exp}\left(\mathbf{O}_i^{\rm exp}, \mathbf{O}_i^{\rm th}(\mathbf{C},\boldsymbol{\theta})\right) \times L_\theta(\boldsymbol{\theta})$$

```
from flavio.statistics.likelihood import Likelihood
llh = Likelihood(observables=['DeltaM_d', 'DeltaM_s'])
```

However, this could be quite computationally demanding, so there is an option that is called "fast likelihood" approach

$$\chi^2(\vec{\xi}) = \vec{\Delta}^T C^{-1}(\vec{\xi} = \hat{\vec{\xi}})\vec{\Delta}, \qquad \Delta_i = (x_i^{\rm exp} - x_i^{\rm th}(\vec{\theta}))$$

$$C(\vec{\xi}) = C_{\rm exp} + C_{\rm th}(\vec{\xi})$$

Flavio.

https://flav-io.github.io/

Fits:

Flavio provides a way of fitting the results using the likelihood in two different approaches:

Bayesian

```
import flavio.statistics.fits

ckmfit = flavio.statistics.fits.BayesianFit(
  name = 'Bayesian SM CKM fit',
  fit_parameters = ['Vus', 'Vub', 'Vcb', 'gamma'],
  nuisance_parameters = ['bag_B0_1', ...],
  observables  = ['eps_K', 'DeltaM_s', ...],
  )
```

Frequentist

```
import flavio.statistics.fits

vubvcbfit = flavio.statistics.fits.FrequentistFit(
  name = 'Frequentist SM Vub-Vcb fit',
  fit_parameters = ['Vub', 'Vcb'],
  nuisance_parameters = ['bag_B0_1', ...],
  observables  = ['eps_K', 'DeltaM_s', ...],
  )
```

Then we have our fit on the parameters/Wilson coefficients and Flavio can print the results in plots.

Flavio.

https://flav-io.github.io/

Adding new observables and parameters.

Observables:

```
from flavio.classes import Observable, Prediction

def my_fct(wc_obj, par):
    return par['m_B0']

Observable('m_B0-obs')
Prediction('m_B0-obs', my_fct)
```

Parameters:

```
from flavio.classes import Parameter

par = flavio.default_parameters # or any other instance of flavio.classes.ParameterConstraint
my_p = Parameter('my_parameter')
my_p.tex = r"$m_\text{par}$"
my_p.description = r"Toy parameter with the TeX representation $m_\text{obs}$"

par.set_constraint('my_parameter', '0.123(45)')
```

Thank you for your attention!