A global fit at 1-loop the easy way - VLQs for BSM

Matthew Kirk
ICCUB, Barcelona

Institut de Ciències del Cosmos

UNIVERSITAT DE BARCELONA

EFT2023, MITP – 1 Sep 2023

(based on 2204.05962, 2212.06862 with Crivellin, Kitahara, Mescia)

Outline

- Some motivation for VLQs
  - And why 1-loop is important
- How we did 1-loop calculations
  - First the hard way, and then the easy way
- Putting everything in the fit
- The physics results

Motivations for vector-like fermions

- Appear in many BSM theories – GUTs, extra dimensions, composite Higgs
- Can explain $(g-2)_\mu$, $b \to s\ell\ell$, CAA, ...
- Not currently ruled out by experiment (unlike heavy chiral fermions)

Vector-like fermions (VLFs)

- Left and right components have same gauge charges
- Allows to directly write a mass term in the Lagrangian
  - Not limited to electroweak scale

VLQs

- But after EW symmetry breaking, can mix with the SM quarks
  - So all VLQs cause shifts in many processes, already tree level!
  - And of course even more @ 1-loop!

Vector-like quarks (VLQs)

Name	U	D	Q_1	Q_5	Q_7	T_1	T_2
Irrep	$(3,1)_{\frac{2}{3}}$	$(3,1)_{-\frac{1}{3}}$	$(3,2)_{\frac{1}{6}}$	$(3,2)_{-\frac{5}{6}}$	$(3,2)_{\frac{7}{6}}$	$(3,3)_{-\frac{1}{3}}$	$(3,3)_{\frac{2}{3}}$

- Lots of different representations, so can mix (and therefore affect) lots of quark processes

$$
\begin{aligned}
-\mathcal{L}_{\rm VLQ} =& \xi^U_{fi}\bar{U}_f\tilde{H}^\dagger q_i + \xi^D_{fi}\bar{D}_f H^\dagger q_i + \xi^u_{fi}\bar{Q}_f\tilde{H}u_i + \xi^d_{fi}\bar{Q}_f H d_i \\
&+\xi^{Q_5}_{fi}\bar{Q}_{5,f}\tilde{H}d_i + \xi^{Q_7}_{fi}\bar{Q}_{7,f}Hu_i + \frac{1}{2}\xi^{T_1}_{fi}H^\dagger\tau\cdot\bar{T}_{1,f}q_i + \frac{1}{2}\xi^{T_2}_{fi}\tilde{H}^\dagger\tau\cdot\bar{T}_{2,f}q_i + \text{h.c.}\,,
\end{aligned}
\qquad (3.5)
$$

Vector-like quarks (VLQs)

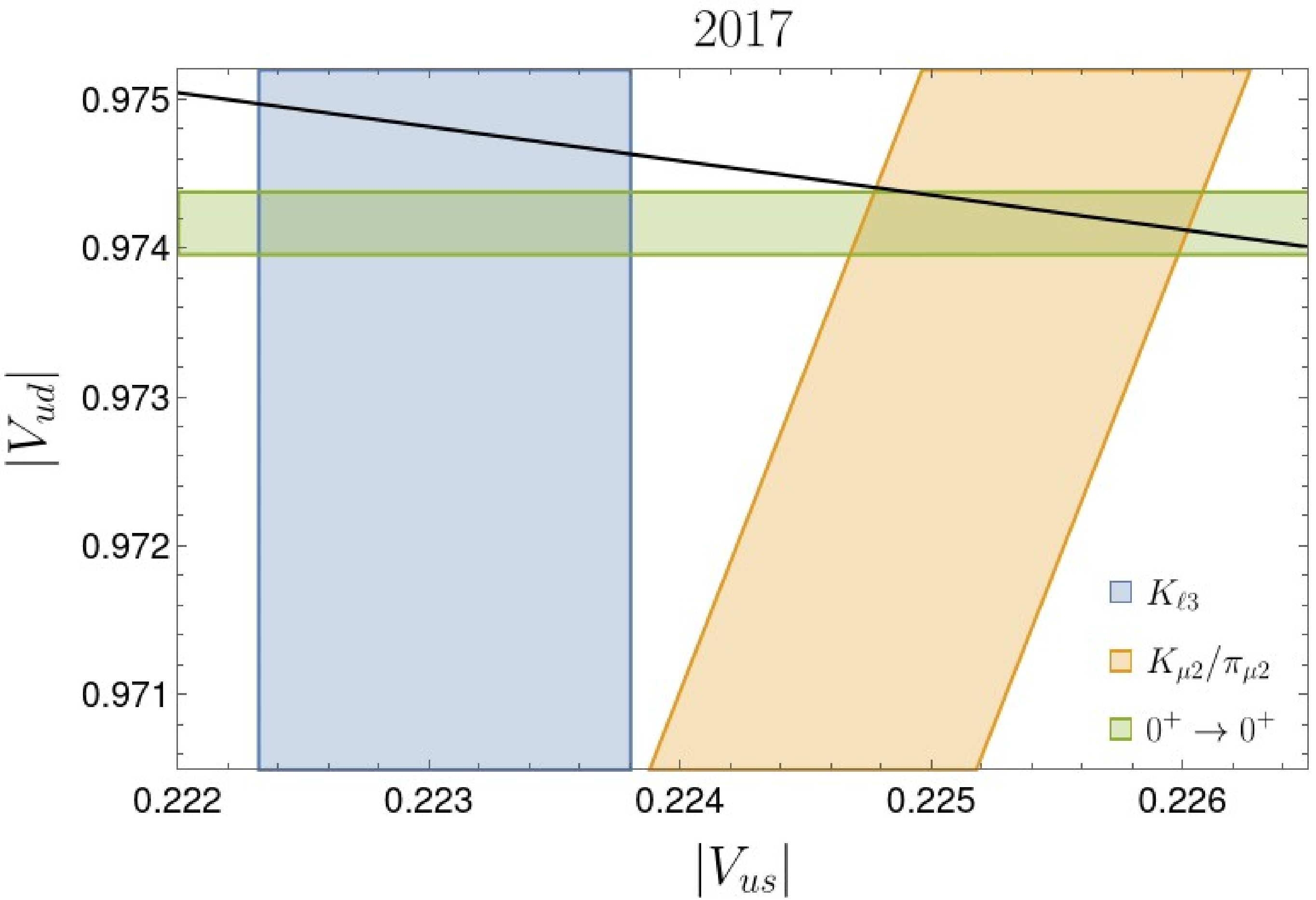
- Lots of different representations, so can mix (and therefore affect) lots of quark processes
  - Mix with 2nd/3rd gen up-type => enhanced $t \to cZ$ plus $b \to s\ell\ell$ (2204.05962)
  - Mix with 1st/2nd gen up- or down-type => CAA (2212.06862)

$b \to s\ell\ell$

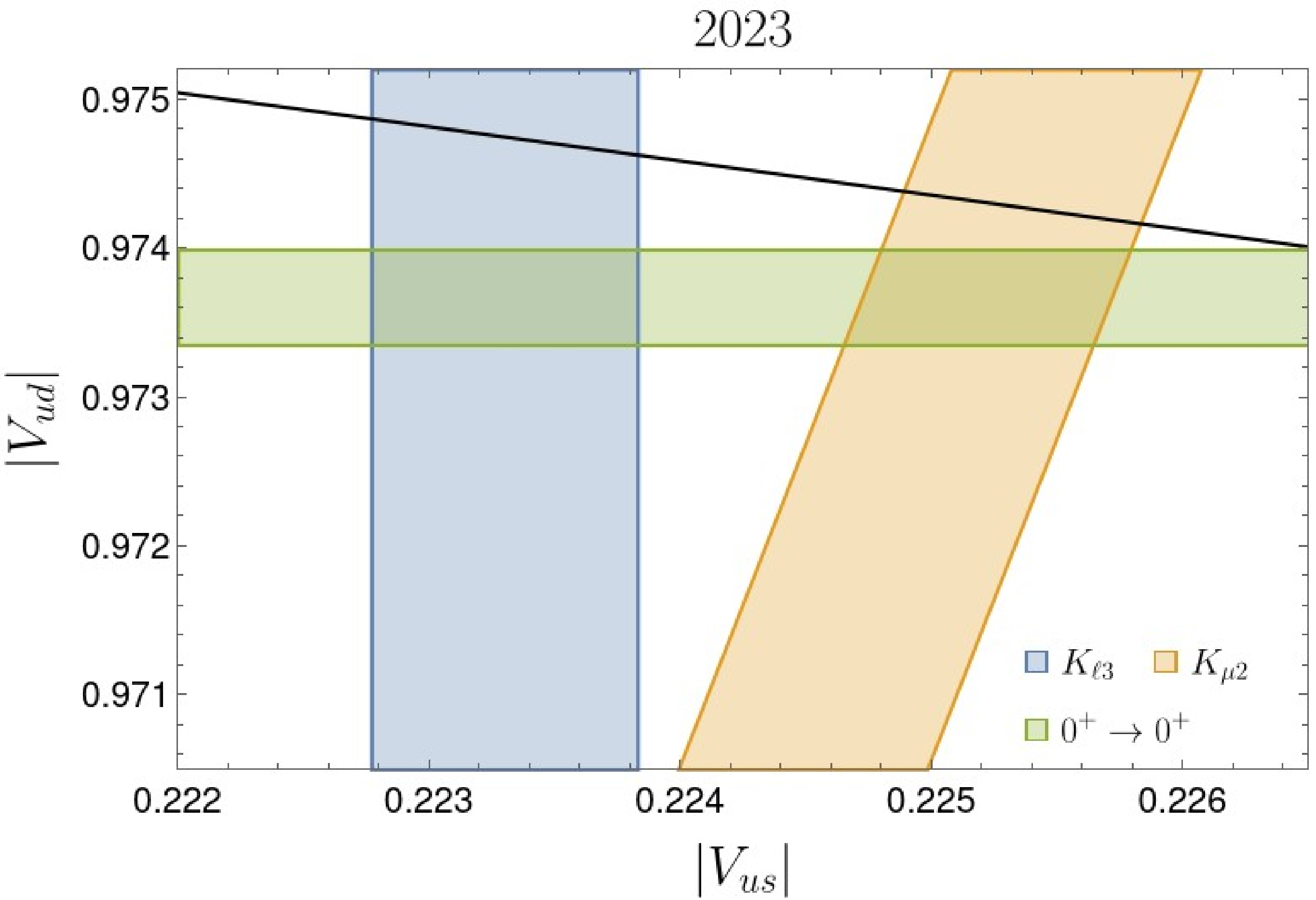
- Even now R_K seems SM like, still plenty of tension in $b \to s\mu\mu$ measurements
- Even better for VLQs – no need to do anything fancy on the lepton side

CAA?

- Cabibbo Angle Anomaly
- Recent (since 2018ish) changes to V_{ud} and V_{us} determinations mean there is now a roughly $3\,\sigma$ discrepancy between experiments and the relationship predicted by the SM => $V_{ud}^2 + V_{us}^2 = 1$



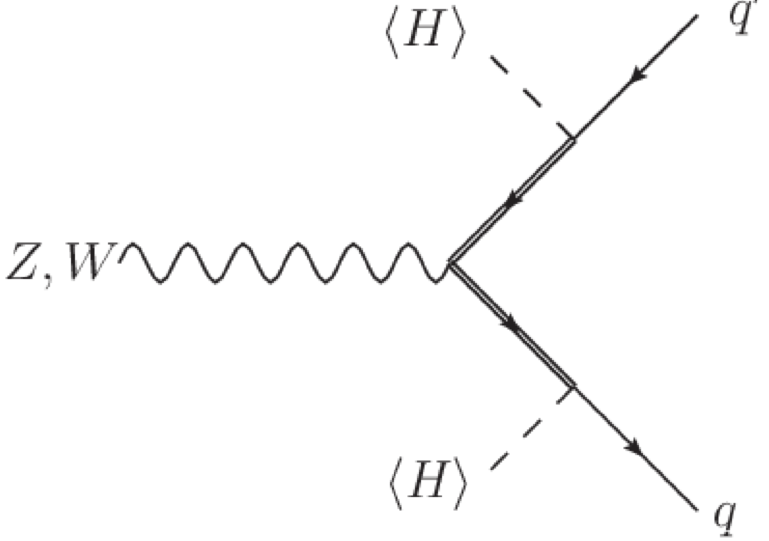
2017
0.975
0.974
0.973
0.972
0.971
$|V_{ud}|$
0.222
0.223
0.224
0.225
0.226
$|V_{us}|$
$K_{\ell 3}$
$K_{\mu 2}/\pi_{\mu 2}$
$0^+ \to 0^+$



2023
0.975
0.974
0.973
0.972
0.971
$|V_{ud}|$
0.222
0.223
0.224
0.225
0.226
$|V_{us}|$
$K_{\ell 3}$
$K_{\mu 2}$
$0^+ \to 0^+$

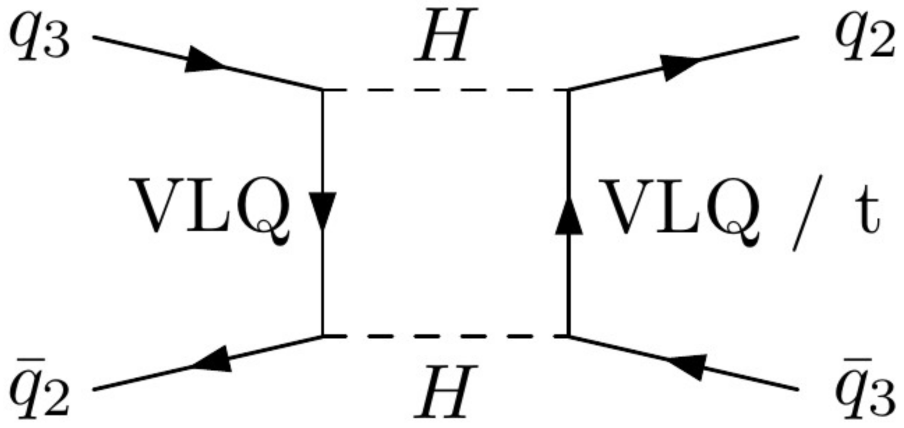
VLQs at tree level

- Affect Z and W decays => lots of effects
- E.g.
  - Flavour changing Z vertex
  - Modified W vertex



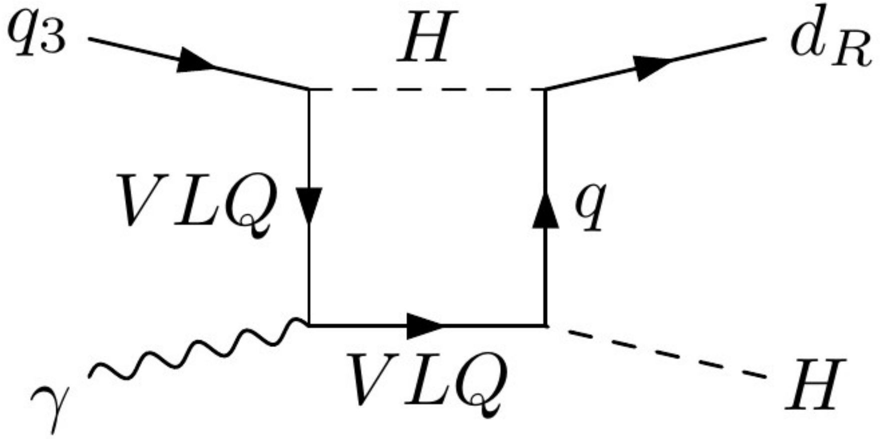
VLQs at 1-loop

- B_s mixing (or meson mixing in general)
- Radiative decays
- W mass



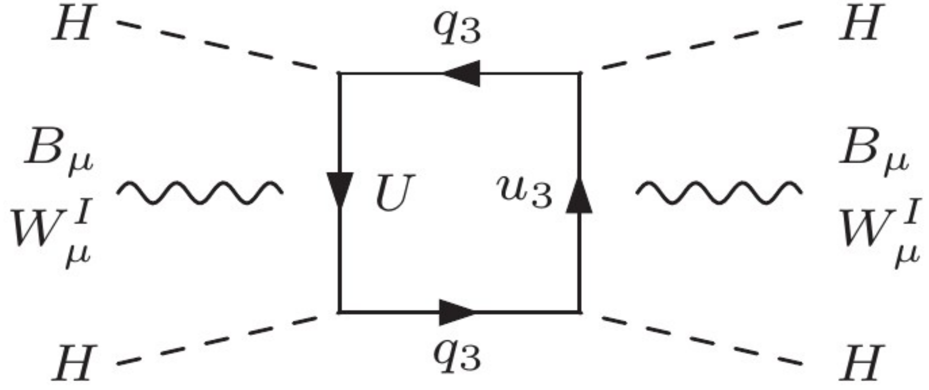
VLQs at 1-loop

- B_s mixing (or meson mixing in general)
- **Radiative decays**
- W mass



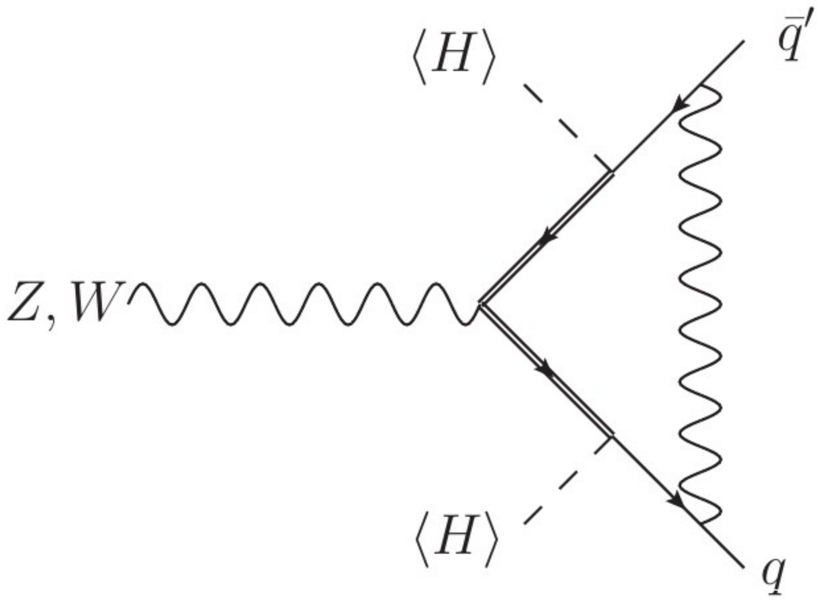
VLQs at 1-loop

- B_s mixing (or meson mixing in general)
- Radiative decays
- **W mass**



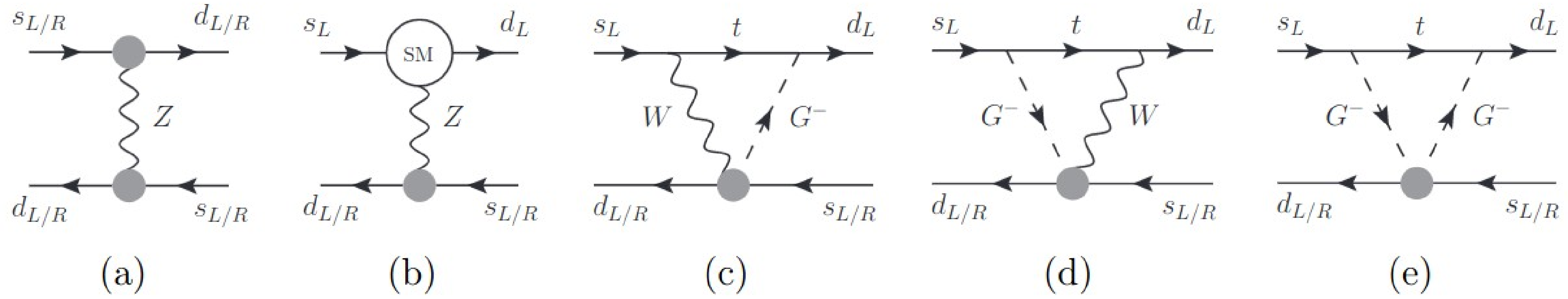
VLQs at 1-loop

- Also further modified gauge couplings
  - E.g. at tree level the U VLQ only modifies Zuu vertex, but @ 1-loop also modifies Zdd
  - So can give effects in Zbb or Zbs for example



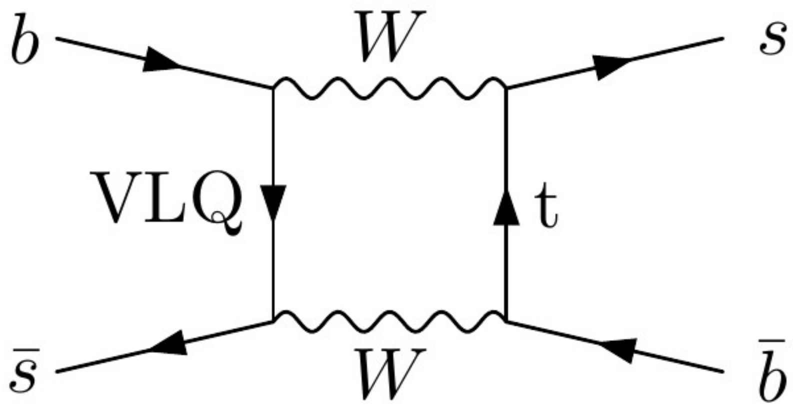
VLQs at 1-loop

- ϵ_K from modified Zsd
  - With real BSM couplings, no imaginary contribution to $\Delta F = 2$ SMEFT coefficients
  - But 1-loop matching from SMEFT to WET picks up phase from SM penguin (see 1612.08839, 1703.04753)



Calculating 1-loop effects

- Fixed order way
  - Directly calculated every observable
  - Large logs common e.g. B mixing: $\log(M/m_t)$



Calculating 1-loop effects

- EFT way
  - VLQs have mass far above SM scale
    - Exp limit is 1.3 TeV for 3rd gen quark couplings 1808.02343
    - For 1st or 2nd gen, limit is similar 2006.07172
  - So integrate them out and use the SMEFT

SMEFT

- "Factorises" calculations
  - Match UV to SMEFT → RG in SMEFT (→ match SMEFT to LEFT → RG in LEFT) → observables in terms of WCs

SMEFT

- "Factorises" calculations
  - Match UV to SMEFT → RG in SMEFT (→ match SMEFT to LEFT → RG in LEFT) → observables in terms of WCs
- Each step is independent

SMEFT

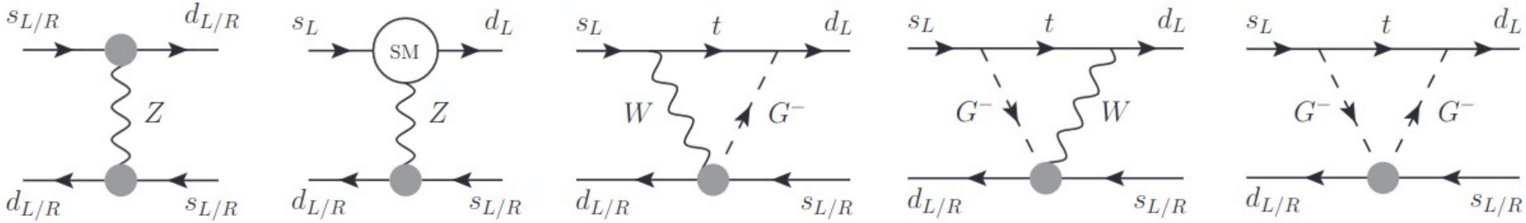
- Match UV to SMEFT
  - Model dependent
- RG in SMEFT
  - Alonso, Jenkins, Manohar, Trott
- Match SMEFT to LEFT
  - Jenkins, Manohar, Stoffer & Dekens, Stoffer
- RG in LEFT
  - Jenkins, Manohar, Stoffer
  - Plus higher orders in QCD
- Observables in terms of WCs
  - Everyone

SMEFT

- Match UV to SMEFT
  - Until recently, by hand
- RG in SMEFT:
  - DsixTools, wilson
- Match SMEFT to LEFT
  - DsixTools, wilson
- RG in LEFT
  - DsixTools, wilson
- Observables in terms of WCs
  - flavio, EOS

1-loop SMEFT → LEFT

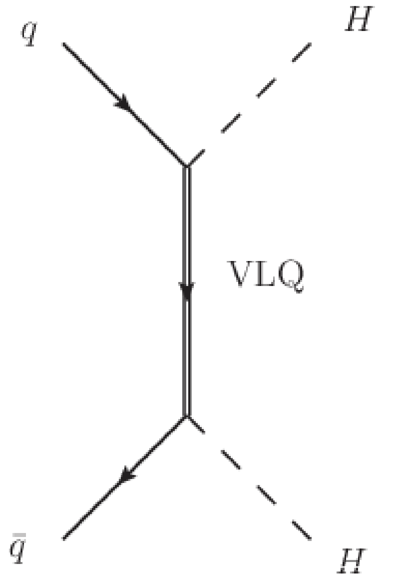
- ϵ_K from modified Zsd
  - But 1-loop matching from SMEFT to WET picks up phase from SM penguin (see 1612.08839, 1703.04753)



  - This automatically included if you remember to turn on 1-loop matching in wilson (which is off by default)

Matching to the SMEFT

- Tree level easy
  - C_{Hq}, C_{uH}
- 1 loop harder
  - See hep-ph/9310302, 2003.12525, 2003.05936, 2107.12133, …



VLQs @ 1-loop

- We spent about 3 months trying to calculate all the coefficients
  - (where by "all" I mean the ones we thought were relevant!)
- Lots learnt along the way

VLQs @ 1-loop

- Finite and log parts comparable!
- E.g. for B_s mixing, divergent box with VLQ and top gives something like $3 + 4\log(M_{\mathrm{VLQ}}/m_t)$
    - Log you can get from RG running
    - But finite part is new from 1-loop matching

VLQs @ 1-loop

- Also unexpected cancellations:
  - B_s mixing:
    - $\frac{\xi^4}{M^2}$ vs $\frac{8\xi^2 y_{\rm top}^2 V_{tb} V_{ts} \log(M/m_t)}{M^2}$
    - Accidental at our considered coupling and masses

VLQs @ 1-loop

- Also unexpected cancellations:
  - $b \to s\gamma$:
    - $$C_{7\gamma}(M_W) \sim C_{d(B,W)} + \frac{1}{16\pi^2} C_{Hq}$$
      $$= 0.2\frac{\xi^2}{M^2} - 0.15\frac{\xi^2}{M^2}$$
    - More robust cancellation

MatchMakerEFT

- Dec 2021 – paper on arXiv 2112.10787
- UV theory specified in terms of FeynRules .fr file
- Matching then proceeds totally automatically

MatchMakerEFT

- Dec 2021 – paper on arXiv 2112.10787
- UV theory specified in terms of FeynRules .fr file
- Matching then proceeds totally automatically

Other matching software is available!

VLQs in MatchMakerEFT

```
M$ClassesDescription = {
F[101] == {
    ClassName        -> VLQQ7,
    Indices          -> {Index[SU2D], Index[Colour]},
    SelfConjugate    -> False,
    QuantumNumbers   -> {Y -> 7/6},
    Mass             -> MQ7,
    FullName        -> "heavy"
}
};

M$Parameters = {

xiQ7 == {
    ParameterType     -> Internal,
    Indices           -> {Index[Generation]},
    ComplexParameter  -> True
  },

MQ7 == {
    ParameterType     -> Internal,
    ComplexParameter  -> False
  }

};
```

```
(* ************************** *)
(* *****   Lagrangian   ***** *)
(* ************************** *)

gotoBFM={G[a__]->G[a]+GQuantum[a],Wi[a__]->Wi[a]+WiQuantum[a],B[a__]->B[a]+BQuantum[a]};

LHeavy := Block[{mu},
 +I*(VLQQ7bar.Ga[mu].DC[VLQQ7, mu])-MQ7*VLQQ7bar.VLQQ7
 ]/.gotoBFM;



LHeavylight := Block[{sp1,ii,jj,kk, aa,cc,ff1,yuk},
  yuk = -xiQ7[ff1] VLQQ7bar[sp1, ii, cc] UR[sp1, ff1, cc] Phi[ii]
        ;
  yuk+HC[yuk]
 ];

LNP := LHeavy + LHeavylight;

Ltot := LSM + LNP;
```

VLQs in MatchMakerEFT

- Quick, no supercomputer needed!
- All algebraic

VLQs in MatchMakerEFT

```
alphaOuG[mif1_, mif2_] → 1/(192 MQ7^2 π^2) onelooporder
  (-3 g3 xiQ7[mif2] × xiQ7bar[fl1] × yu[mif1, fl1] - g3 xiQ7[mif2] × xiQ7bar[mif3] × yu[mif1, mif3]),
alphaOuW[mif1_, mif2_] → 0, alphaOuB[mif1_, mif2_] → 0,
alphaOdG[mif1_, mif2_] → 0,
alphaOdW[mif1_, mif2_] → 0,
alphaOdB[mif1_, mif2_] → 0,
alphaOeW[mif1_, mif2_] → 0,
alphaOeB[mif1_, mif2_] → 0,
alphaOHq1[mif1_, mif2_] → 1/(17280 MQ7^2 π^2)
 onelooporder (135 xiQ7[fl1] × xiQ7bar[fl2] × yu[mif1, fl2] × yubar[mif2, fl1] +
    270 Log[MQ7^2/μ^2] xiQ7[fl1] × xiQ7bar[fl2] × yu[mif1, fl2] × yubar[mif2, fl1] +
    135 xiQ7[fl1] × xiQ7bar[mif3] × yu[mif1, mif3] × yubar[mif2, fl1] +
    135 xiQ7[mif3] × xiQ7bar[fl1] × yu[mif1, fl1] × yubar[mif2, mif3] +
    180 xiQ7[mif4] × xiQ7bar[mif3] × yu[mif1, mif3] × yubar[mif2, mif4]), alphaOHq3[mif1_, mif2_] →
 1/(1920 MQ7^2 π^2) onelooporder (-15 xiQ7[fl1] × xiQ7bar[mif3] × yu[mif1, mif3] × yubar[mif2, fl1] -
    15 xiQ7[mif3] × xiQ7bar[fl1] × yu[mif1, fl1] × yubar[mif2, mif3] -
    20 xiQ7[mif4] × xiQ7bar[mif3] × yu[mif1, mif3] × yubar[mif2, mif4]),
alphaOHu[mif1_, mif2_] → (xiQ7[mif2] × xiQ7bar[mif1])/(2 MQ7^2) + 1/(34560 MQ7^4 π^2)
  onelooporder (-2700 MQ7^2 xiQ7[fl1] × xiQ7[mif2] × xiQ7bar[fl1] × xiQ7bar[mif1] +
    3240 MQ7^2 Log[MQ7^2/μ^2] xiQ7[fl1] × xiQ7[mif2] × xiQ7bar[fl1] × xiQ7bar[mif1] - 1620 MQ7^2 xiQ7[MIF1] ×
```

MatchMakerEFT → smelli

- From MatchMakerEFT we get algebraic expressions for the WCs at 1-loop, in nice simple format (i.e. with generic indices, and repeated indices for summation)
- In smelli (well wilson) need to specify each specific WC, in the non-redundant basis

MatchMakerEFT → smelli

MJKirk commented on Oct 26, 2022 Contributor ···

When matching from a UV theory onto the SMEFT, often one gets pretty simple generic formulas for the SMEFT coefficients. As an example, take this from the wilson paper (bottom of page 9)

$$\left[C_{lq}^{(1)}\right]_{ijkl} = \lambda^{\ell}_{ij}\lambda^{q}_{kl}\,C_1\,, \qquad \left[C_{lq}^{(3)}\right]_{ijkl} = \lambda^{\ell}_{ij}\lambda^{q}_{kl}\,C_3\,.$$

But actually typing out all the coefficients is tedious and error prone. Again from there, you give the example code

```
from wilson import Wilson
ll_33 = ...
...
w = Wilson({'lq3_3333': ll_33 * lq_33 * C3,
            'lq1_3333': ll_33 * lq_33 * C1,
            'lq3_2223': ll_22 * lq_23 * C3, ...},
            scale=Lambda, eft='SMEFT', basis='Warsaw')
```

If I'm correct, there are actually another 17 coefficients hidden in that "..." that you didn't bother to type out, and of course you have to remember which are the non-redundant ones.

https://github.com/wilson-eft/wilson/issues/105

Instead, it's pretty easy to use the following code

```
# Some example values
 ll_33 = 1
lq_33 = 1
ll_23 = 0.2
lq_23 = -0.1
C1 = -0.05
C3 = 0.02

ll = np.array(((0,0,0),(0,ll_23**2, ll_23), (0, ll_23, ll_33)))
lq = np.array(((0,0,0),(0,lq_23**2, lq_23), (0, lq_23, lq_33)))
Clq1 = C1*np.einsum("ij,kl->ijkl", ll, lq)
Clq3 = C1*np.einsum("ij,kl->ijkl", ll, lq)
```

to generate all the wilson coefficients (in what should be the basis where coefficients have the same symmetries as the operators).
Then you can do

```
wilson.util.smeftutil.arrays2wcxf_nonred(wilson.smeftutil.add_missing({"lq3": Clq3, "lq1": Clq1}))
```

to get a dictionary with just the non-redundant coefficients needed to initialise a Wilson instance.

MatchMakerEFT → smelli

- Useful: numpy.einsum
- Einstein summation convention in Python
  - $C_{ijkl} = \xi_i \xi_l (Y^u)_{jk}$
  - np.einsum("i,l,jk→ijkl", xi, xi, Yu)

Real life example

SMEFT modified boson WCs	expression
alphaOHq1[i, j]	$\frac{\text{xiU}[j]\,\text{xiUbar}[i]}{4\,\text{MVLQU}^2}$
alphaOHq3[i, j]	$-\frac{\text{xiU}[j]\,\text{xiUbar}[i]}{4\,\text{MVLQU}^2}$
alphaOHu[i, j]	0
alphaOHd[i, j]	0
alphaOHud[i, j]	0

SMEFT DF=2 WCs	expression
alphaOqq1[i, j, k, l]	$-\frac{\text{xiU}[j]\,\text{xiU}[l]\,\text{xiUbar}[i]\,\text{xiUbar}[k]}{256\,\text{MVLQU}^2\,\pi^2}+\frac{3\,\text{xiU}[l]\,\text{xiUbar}[k]\,\text{yu}[i,\text{fl1}]\,\text{yubar}[j,\text{fl1}]}{512\,\text{MVLQU}^2\,\pi^2}+\frac{3\,\text{xiU}[j]\,\text{xiUbar}[i]\,\text{yu}[k,\text{fl1}]\,\text{yubar}[l,\text{fl1}]}{512\,\text{MVLQU}^2\,\pi^2}$
alphaOqq3[i, j, k, l]	$-\frac{\text{xiU}[j]\,\text{xiU}[l]\,\text{xiUbar}[i]\,\text{xiUbar}[k]}{256\,\text{MVLQU}^2\,\pi^2}+\frac{3\,\text{xiU}[l]\,\text{xiUbar}[k]\,\text{yu}[i,\text{fl1}]\,\text{yubar}[j,\text{fl1}]}{512\,\text{MVLQU}^2\,\pi^2}+\frac{3\,\text{xiU}[j]\,\text{xiUbar}[i]\,\text{yu}[k,\text{fl1}]\,\text{yubar}[l,\text{fl1}]}{512\,\text{MVLQU}^2\,\pi^2}$
alphaOqu1[i, j, k, l]	$-\frac{3\,\text{xiU}[j]\,\text{xiUbar}[i]\,\text{yu}[\text{fl1},l]\,\text{yubar}[\text{fl1},k]}{128\,\text{MVLQU}^2\,\pi^2}-\frac{\text{xiU}[\text{fl1}]\,\text{xiUbar}[\text{fl1}]\,\text{yu}[i,l]\,\text{yubar}[j,k]}{96\,\text{MVLQU}^2\,\pi^2}$
alphaOqu8[i, j, k, l]	$-\frac{\text{xiU}[\text{fl1}]\,\text{xiUbar}[\text{fl1}]\,\text{yu}[i,l]\,\text{yubar}[j,k]}{16\,\text{MVLQU}^2\,\pi^2}$
alphaOuu[i, j, k, l]	0

Real life example

SMEFT modified boson WCs	expression
alphaOHq1[i, j]	$\frac{\text{xiU}[j] \times \text{xiUbar}[i]}{4\,\text{MVLQU}^2}$
alphaOHq3[i, j]	$-\frac{\text{xiU}[j] \times \text{xiUbar}[i]}{4\,\text{MVLQU}^2}$
alphaOHu[i, j]	0
alphaOHd[i, j]	0
alphaOHud[i, j]	0

SMEFT DF=2 WCs	expression
alphaOqq1[i, j, k, l]	$-\frac{\text{xiU}[j] \times \text{xiU}[l] \times \text{xiUbar}[i] \times \text{xiUbar}[k]}{256\,\text{MVLQU}^2\,\pi^2} + \frac{3\,\text{xiU}\ldots}{}$
alphaOqq3[i, j, k, l]	$-\frac{\text{xiU}[j] \times \text{xiU}[l] \times \text{xiUbar}[i] \times \text{xiUbar}[k]}{256\,\text{MVLQU}^2\,\pi^2} + \frac{3\,\text{xiU}\ldots}{}$
alphaOqu1[i, j, k, l]	$-\frac{3\,\text{xiU}[j] \times \text{xiUbar}[i] \times \text{yu}[\text{fl1},l] \times \text{yubar}[\text{fl1},k]}{128\,\text{MVLQU}^2\,\pi^2}$
alphaOqu8[i, j, k, l]	$-\frac{\text{xiU}[\text{fl1}] \times \text{xiUbar}[\text{fl1}] \times \text{yu}[i,l] \times \text{yubar}[j,k]}{16\,\text{MVLQU}^2\,\pi^2}$
alphaOuu[i, j, k, l]	0

```python
def wc_fct_Uonly(wcs):
    xiU_1, xiU_2 = wcs
    xiU = np.array((xiU_1, xiU_2, 0))

    phiq1 =  (1/4) * np.einsum("i,j->ij", xiU, xiU) / MVLQ**2
    phiq3 = -(1/4) * np.einsum("i,j->ij", xiU, xiU) / MVLQ**2
    phiu = 0
    phid = 0
    phiud = 0

    # DF=2 coefficients
    qq1 = (-1 * np.einsum("i,j,k,l->ijkl", xiU, xiU, xiU, xiU) / (256 * _loopfactor)
           +3 * np.einsum("k,l,iA,jA->ijkl", xiU, xiU, _yu, _yubar) / (512 * _loopfactor)
           +3 * np.einsum("i,j,kA,lA->ijkl", xiU, xiU, _yu, _yubar) / (512 * _loopfactor) )

    qq3 = (-1 * np.einsum("i,j,k,l->ijkl", xiU, xiU, xiU, xiU) / (256 * _loopfactor)
           +3 * np.einsum("k,l,iA,jA->ijkl", xiU, xiU, _yu, _yubar) / (512 * _loopfactor)
           +3 * np.einsum("i,j,kA,lA->ijkl", xiU, xiU, _yu, _yubar) / (512 * _loopfactor) )

    qu1 = (-3 * np.einsum("i,j,Al,Ak->ijkl", xiU, xiU, _yu, _yubar) / (128 * _loopfactor)
           -1 * np.einsum("A,A,il,jk->ijkl", xiU, xiU, _yu, _yubar) / (96 * _loopfactor) )

    qu8 =  -1 * np.einsum("A,A,il,jk->ijkl", xiU, xiU, _yu, _yubar) / (16 * _loopfactor)

    uu = 0

    wc_arrays = {"phiq1": phiq1, "phiq3": phiq3, "phiu": phiu, "phid": phid, "phiud": phiud,
                 "qq1": qq1, "qq3": qq3, "qu1": qu1, "qu8": qu8, "uu": uu}
    return C_arrays_to_C_wcxf(wc_arrays)
```

Future

- As I understand it, "MatchingDB" has this function built in

Future

- As I understand it, "MatchingDB" has this function built in
- Project by Juan Carlos Criado & Jose Santiago (see talk @ SMEFT-Tools 2022 or Gitlab docs)
- Database to contain tree and loop level matching coefficients analytically, plus python interface

Future

- As I understand it, "MatchingDB" has this function built in

with_smelli.py 439 B Edit

```python
import numpy as np
import smelli

from matchingdb import JsonDB

gl = smelli.GlobalLikelihood()
db = JsonDB.load("smeft_dim6_tree.json")

evaluator = db.select_terms(
    fields=["B"], output_format="numeric", parameters={"gphiB", "M_B"}
)

coeff_values = evaluator({"gphiB": np.array([0.3]), "M_B": np.array([2000.0])})
pp = gl.parameter_point(coeff_values, scale=1000)
df = pp.obstable()
print(df.sort_values("pull SM", ascending=True))
```

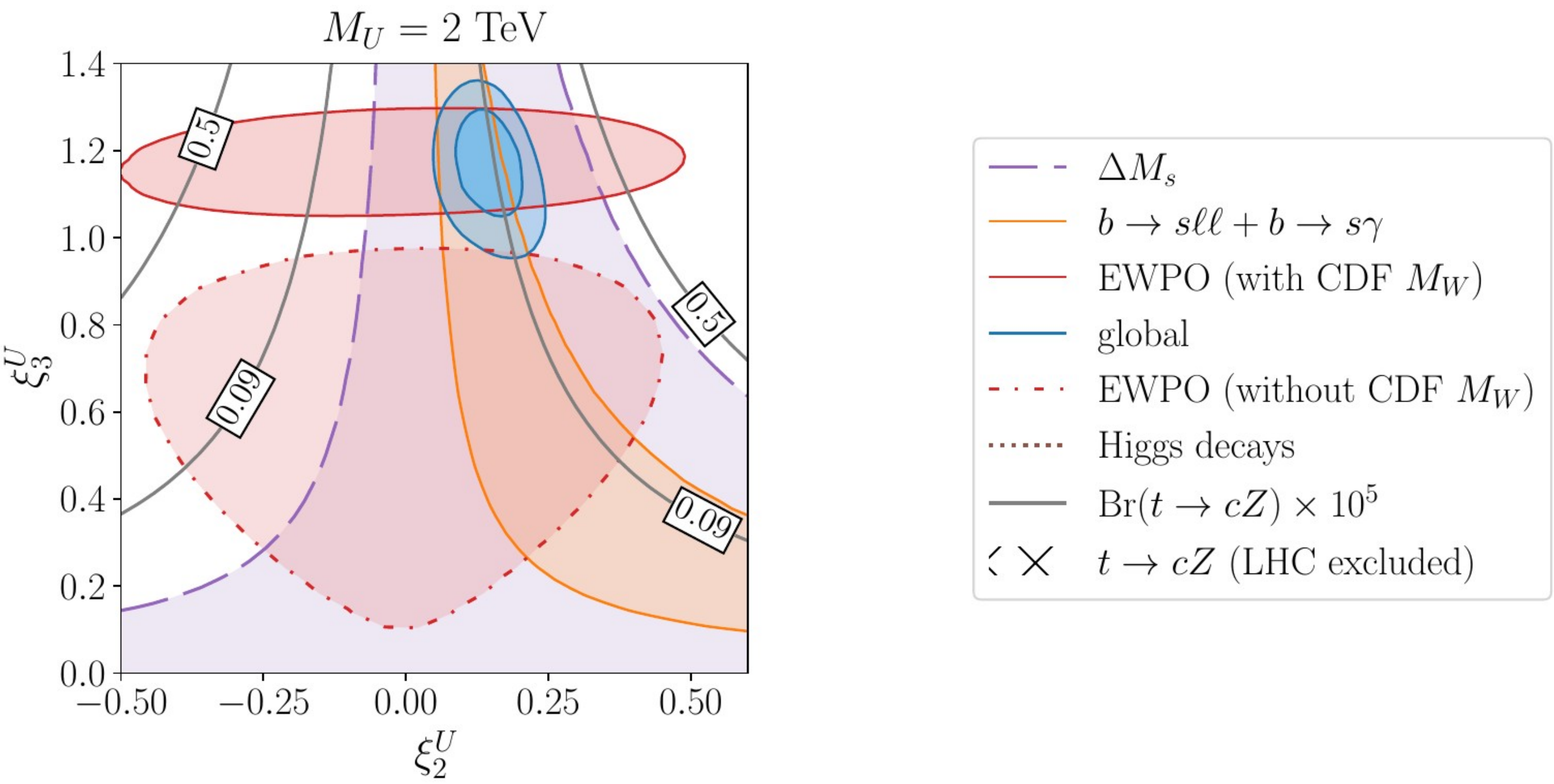
Future

- As I understand it, “MatchingDB” has this function built in
- And there is a plan for MatchMakerEFT → MatchingDB export
- Final piece of the puzzle!

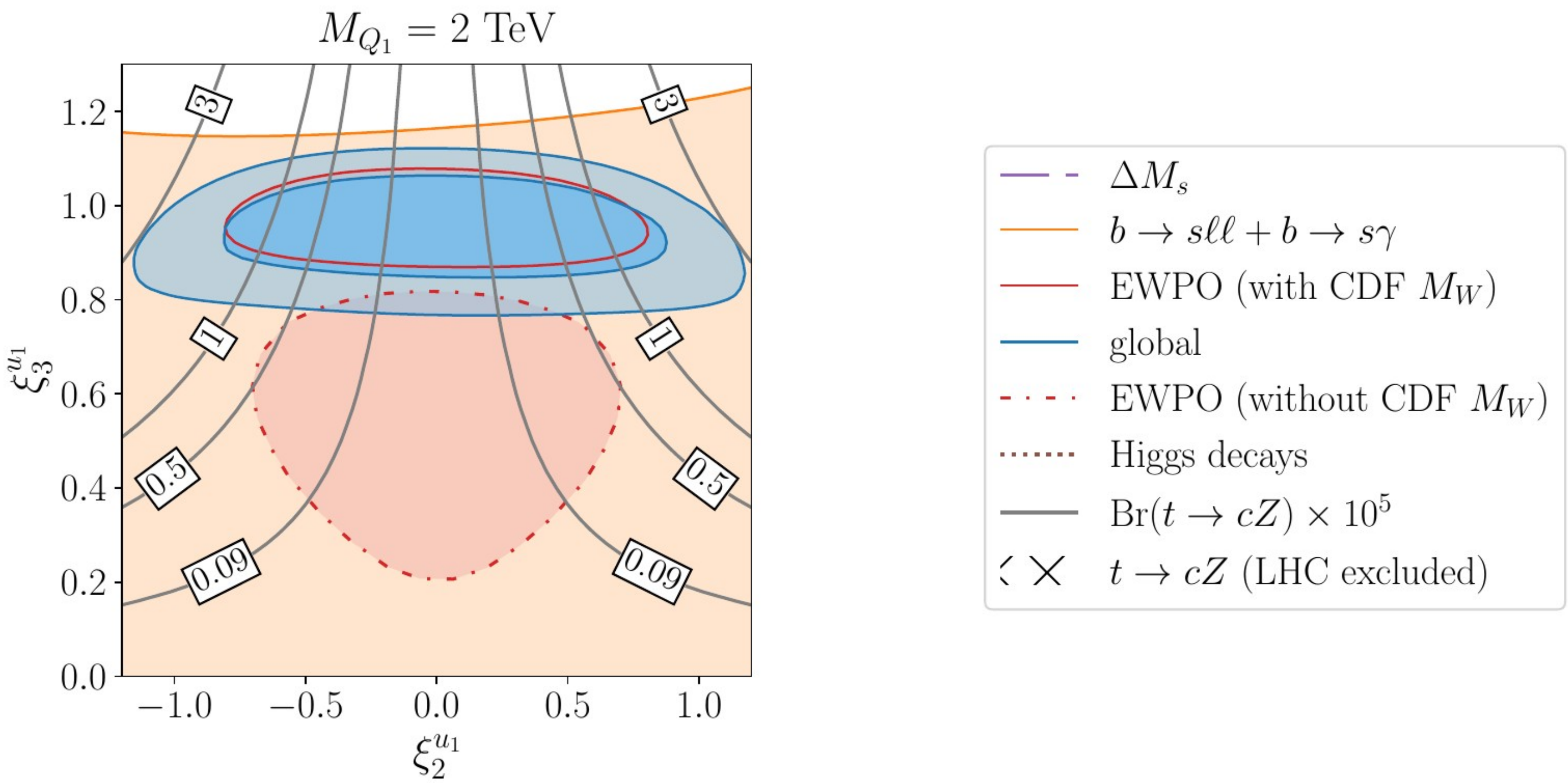
Physics results

- So after all that, what did we learn about the universe?

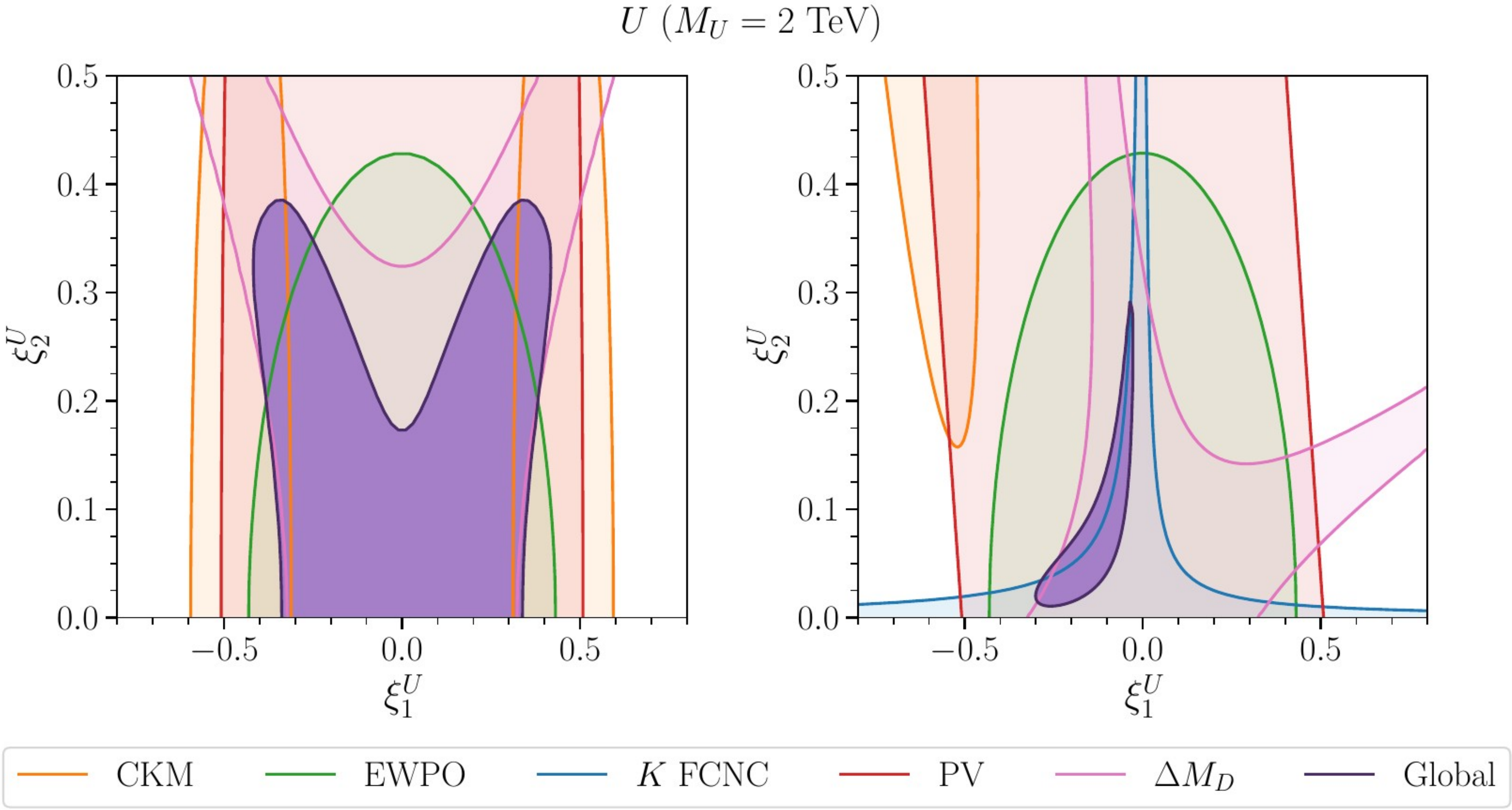
Physics results: $t \to cZ$



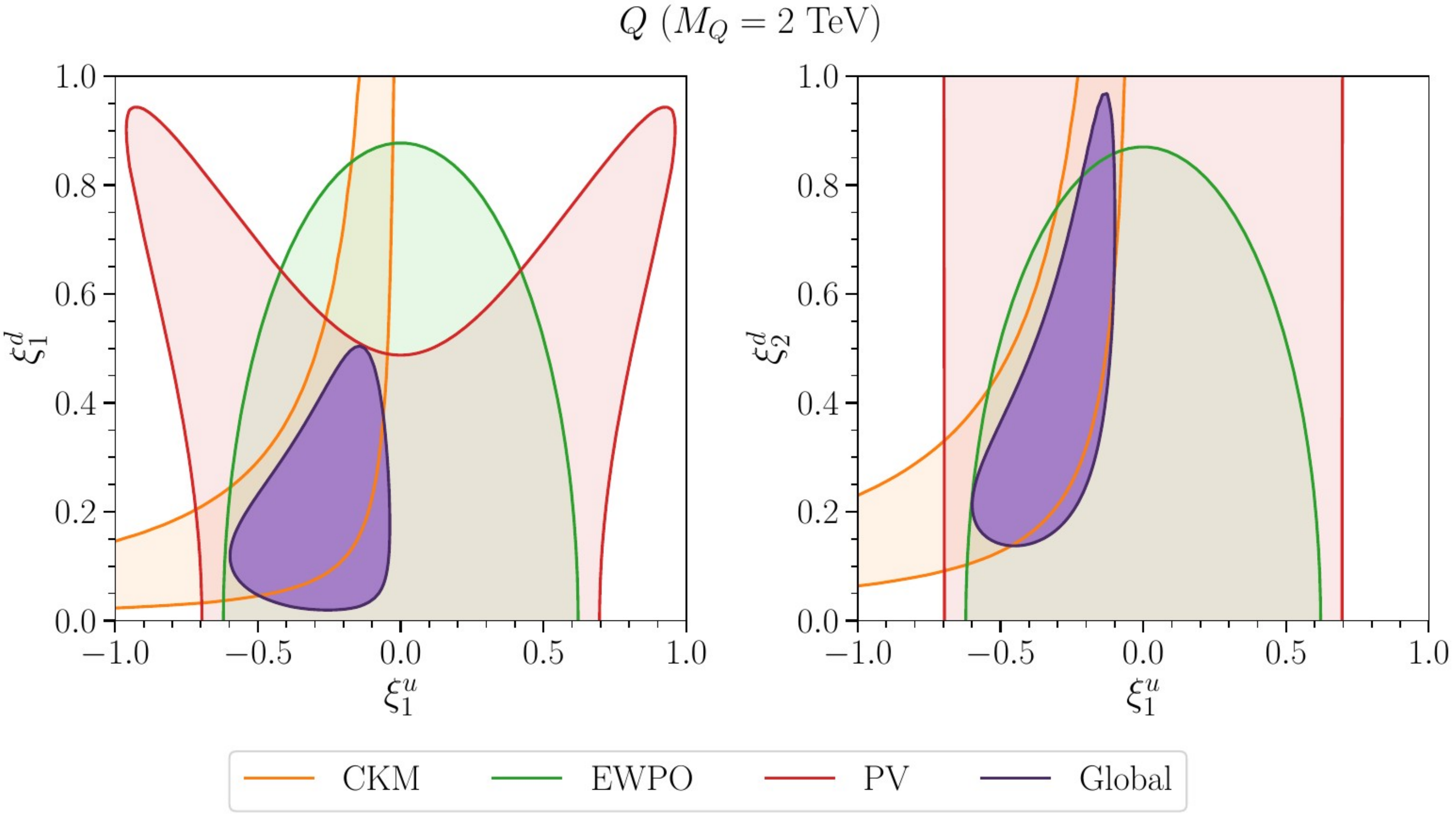
Physics results: $t \to cZ$



Physics results: CAA



Physics results: CAA



Conclusions

- VLQs are interesting BSM models
- Correlation with B physics, M_W, EWPO, ... studied within SMEFT
- Automated 1-loop matching makes analysis very easy

Backup

$t \to c\,Z$ exp limits

	$\mathrm{Br}(t \to cZ) \times 10^5$	$\mathrm{Br}(t \to ch) \times 10^5$
Current LHC ($13\,\mathrm{TeV}, 139\,\mathrm{fb}^{-1}$)	13 [54]	99 [55]
HL-LHC ($14\,\mathrm{TeV}, 3\,\mathrm{ab}^{-1}$)	3.13 [59] (0%) 6.65 [59] (10%)	15 [61]
HE-LHC ($27\,\mathrm{TeV}, 15\,\mathrm{ab}^{-1}$)	0.522 [59] (0%) 3.84 [59] (10%)	7.7 [60] (0%) 8.5 [60] (10%)
FCC-hh ($100\,\mathrm{TeV}, 3\,\mathrm{ab}^{-1}$)		7.7 [64]
FCC-hh ($100\,\mathrm{TeV}, 10\,\mathrm{ab}^{-1}$)		2.39 [63] (5%) 9.68 [62] (10%)
FCC-hh ($100\,\mathrm{TeV}, 30\,\mathrm{ab}^{-1}$)	0.0887 [59] (0%) 3.54 [59] (10%)	0.96 [60] (0%) 3.0 [60] (10%) 4.3 [64]

MatchMakerEFT

- `RGEmaker` mode:
  - Complete RGEs for the ALP-SMEFT up to mass dimension-5 as computed in [64]. Exact agreement was found up to a typo in the original reference.
  - RGEs for the purely bosonic and two-fermion operators in the Warsaw basis [66] as computed in [15–17] and implemented in `DSixTools` [28, 29]. Complete agreement was found.
- `Matching` mode:
  - Scalar singlet. The complete matching up to one-loop order of an extension of the SM with a scalar singlet was recently completed in [67], after several partial attempts [36, 68]. We have found complete agreement with the results in [67].
  - Type-I see-saw model, as computed in [69]. Complete agreement was found.
  - Scalar leptoquarks, as computed in [62]. We have found some minor differences that we are discussing with the authors.
  - Charged scalar electroweak singlet, as computed in [70]. We agree with the result except for a sign in Eqs. (4.14), the terms with Pauli matrices in (4.15), (B.4) and (B.5) (the latter is the culprit of the opposite sign in terms with Pauli matrices) and a factor of 2 in Eq. (4.17) and of 4 in (B.7). We have contacted the authors about these differences.

2112.10787

MatchMakerEFT

- Two step matching:

  1) Create model – quick, low cost

  2) Match model – "slow", high cost

CKM treatment

- Theory prediction needs CKM elements
- CKM elements are determined from observables
- Observables might be affected by NP

CKM treatment

- (a) Solution

The CKM parameters in the SMEFT 1812.08163

Sébastien Descotes-Genon, Adam Falkowski, Marco Fedele, Martín González-Alonso, Javier Virto

- Used by smelli with these 4 observables:
  - $\Delta M_d/\Delta M_s$, $B \to X_c e\nu$, $B \to \tau\nu$, $\dfrac{K \to \mu\nu}{\pi \to \mu\nu}$
- Thus these missing in fit