Updated Oct. 2024

ELVAS

C++ Package for ELectroweak VAcuum Stability

S. Chigusa, T. Moroi, and Y. Shoji

1 Introduction

`ELVAS` is a C++ package for the calculation of the decay rate of a false vacuum at the one-loop level, based on the formulae developed in [1, 2]. The degeneracy of the gauge transverse modes is corrected according to [3].

`ELVAS` is applicable to models with the following features:

- Only one scalar boson is responsible for the vacuum decay.
- Classical scale invariance (approximately) holds. In particular, the potential of the scalar field responsible for the vacuum decay should be well approximated by the quartic form for the calculation of the bounce solution. (Thus, the bounce is nothing but the so-called Fubini instanton.)
- The instability of the scalar potential occurs due to RG effects; thus, the quartic coupling constant becomes negative at a high scale.

If you use `ELVAS` in scholarly work, please cite [1], [2] and [3].

2 Setup

We denote the (complex) scalar field responsible for the vacuum decay as Φ. Then, `ELVAS` assumes that the scalar potential is given in the following form:

$$V(\Phi) = \lambda(\Phi^\dagger\Phi)^2, \tag{1}$$

where λ is a coupling constant. The instability occurs due to the renormalization group (RG) running of λ; λ is positive at a low scale while is negative at a high scale due to RG effect.

The Higgs may interacts with other (real) scalar field σ and chiral fermions ψ_L and ψ_R. The interaction terms are denoted as

$$\mathcal{L}_{\text{int}} = \kappa\sigma^2\Phi^\dagger\Phi + (y\Phi\psi_L\psi_R + \text{h.c.}), \tag{2}$$

where κ and y are coupling constants.

The scalar field Φ may also have gauge interaction. The "gauge coupling constant" in this package is defined such that the corresponding gauge boson mass is given by

$$m_A^2 = 2g^2\langle\Phi^\dagger\Phi\rangle. \tag{3}$$

where the brackets indicate the vacuum expectation value.

For a negative λ, we have the Fubini instsanton, which is given by

$$\bar{\phi}(r) = \sqrt{\frac{8}{|\lambda|}}\frac{1}{R}\left(1+\frac{r^2}{R^2}\right)^{-1}, \tag{4}$$

where $\bar{\phi} = \sqrt{\langle \Phi^\dagger \Phi \rangle / 2}$, r is the distance from the center of the instanton, and R is a constant determining the size of the instanton. We denote the field value at the center of the instanton as $\bar{\phi}_C$, which is related to R by

$$\bar{\phi}_C = \sqrt{\frac{8}{|\lambda|}} \frac{1}{R}. \tag{5}$$

The decay rate with a pair of chiral fermions, a scalar field and a $U(1)$ gauge field is then expressed as

$$\begin{aligned} \gamma &= \int d \ln R^{-1} \frac{\mathcal{V}_G}{R^4} \left[\mathcal{A}'^{(h)} \mathcal{A}^{(\sigma)} \mathcal{A}^{(\psi)} \mathcal{A}'^{(A,\varphi)} e^{-\mathcal{B}} \right]_{\overline{\mathrm{MS}}, \mu \sim 1/\mathrm{R}} \\ &\equiv \int d \ln R^{-1} \exp \left[\ln \frac{d\gamma}{d \ln R^{-1}} \right], \end{aligned} \tag{6}$$

where $\mathcal{V}_G$ is the volume of the group space generated by the broken generators, $\mathcal{B}$ is the tree level action of the instanton, and $\mathcal{A}^{(X)}$ is the quantum correction from particle X. The coupling constants and quantum corrections are renormalized in $\overline{\mathrm{MS}}$-scheme, and the renormalization scale, Q, is taken so that $Q \sim 1/R$ to handle possible large logarithmic corrections. Notice that $\mathcal{V}_G$, has to be measured with "$U(1)$" generators. For the SM or its extensions only with scalars and fermions, it is $2\pi^2$. If Φ has only a U(1) charge, it is 2π.

This package first calculates $\ln \frac{d\gamma}{d \ln R^{-1}}$ for each $\ln R^{-1}$, using a renormalization group data provided by user. Then, it interpolates $\ln \frac{d\gamma}{d \ln R^{-1}}$ and integrates $\frac{d\gamma}{d \ln R^{-1}}$.

3 Download and install

To use `ELVAS` , you need a `c++` compiler that supports C++14, and the `boost` library. The required version of `boost` is 1.59.0[#1].

3.1 Unix

For UNIX systems, follow the instruction below.

1. Clone the `ELVAS` git repository

```
git clone https://github.com/YShoji-HEP/ELVAS.git
```

   Or, you can download it from https://github.com/YShoji-HEP/ELVAS/.

2. Generate a makefile with

```
cmake .
```

   If you do not use the default compiler, use

[#1] We found that `boost 1.58.0` has a serious bug.

```
-DCMAKE_CXX_COMPILER=your++
```

option. When the `boost` library is located at a non-standard directory, specify it with

```
-DBOOST_ROOT=/your/dir
```

or

```
-DBOOST_INCLUDEDIR=/your/inc -DBOOST_LIBRARYDIR=/your/lib
```

If you have installed the google perftools, you can enable `tcmalloc` with

```
-DUSE_TCMALLOC=ON
```

3. Compile `ELVAS` with

```
 make
```

4. Run

```
 ./elvas sm.in sm.dat
```

If everything works well, you will see as a result

```
mHiggs       mTop         log10(gamma x Gyr Gpc^3)
1.250900e+02 1.731000e+02 -5.796066e+02
1.248500e+02 1.731000e+02 -5.396598e+02
1.253300e+02 1.731000e+02 -6.242081e+02
1.250900e+02 1.725000e+02 -9.053941e+02
1.250900e+02 1.737000e+02 -3.968497e+02
```

after a header.

If you have link errors, check the summary of `cmake` and make sure it finds a correct version of `boost`.

If you are not using the default compiler, the installed `boost` library may not be compiled with your compiler. In such a case, you need to use the default compiler or compile `boost` with `toolset` option. Make sure that a path to dynamic library files is correctly added to the environment variable if you use a dynamic library.

From `boost 1.66.0`, library filenames may contain the system architecture tag, *e.g.* `-x32`, but it seems that `cmake 3.11.0-rc1` does not support the tag. In such a case, make symbolic links for `boost_system` and `boost_program_options` so that they do not contain the tag.

3.2 Windows

For Windows, you may use `Cygwin`, `MinGW`, or `Visual C++`. In any case, you need to install `cmake`, which can be downloaded from the official website. For `Cygwin` and `MinGW`, the procedure is almost the same as the UNIX systems. For `Visual C++`, you can generate a VC++ project file with the GUI front-end of `cmake`. Furthermore, a pre-compiled `boost` library can be installed through `NuGet`. After installing it, you can build the project as usual.

4 How to use ELVAS

`ELVAS` provides routines for the calculation of a decay rate together with an interpreter. The easiest way to use this program is to use the interpreter, but you may directly call the routines in your `c++` code.

4.1 Quick guide to use interpreter

The easiest way to calculate vacuum decay rates in your model is to modify `sm.in`, which is provided in the package.

1. Copy `sm.in` so that you can resume the original input file when something goes wrong. (We name the copied file `model.in` in this guide.)

2. Calculate the RG evolution of the couplings in your model. It is preferable to use, at least, two-loop beta functions for the running. The interval of the data points should be short enough; about five points between Q and $10Q$ is enough. If you limit the region of integration, the RG data should cover all the region. If you do not limit the region, you need to solve the RG equations until the Higgs quartic coupling becomes positive again. Since this package does not provide any solver of differential equations, you need to solve them by yourself.

3. Format the RG data as follows. (We name this file `rg.dat` in this guide.)

```
[DATASET] (1.20000e+02 1.70000e+02)
2.40000e+02 6.45699e-01 4.62621e-01 9.00513e-01 ...
3.80374e+02 6.43314e-01 4.63832e-01 8.78139e-01 ...
...
[DATASET] (1.20000e+02 1.70200e+02)
...
```

   The statement of `[DATASET]` indicates the beginning of a “dataset”. The numbers in parentheses are “dataset variables”, which are optional common variables for the dataset. If you do not provide any dataset variables, you can omit the parentheses and just `[DATASET]` is enough. Each row below the `[DATASET]` statement is a “record” that belongs to the dataset. A record should contain information about the renormalization scale and the values of coupling constants at the scale. The dataset ends when it reaches the end of file or a new [DATASET] statement. You may use several formats for numbers like 1, 1.2, 1.23e10 and 2.E-10. If you want to use different delimiters, modify DATASET_DELIM and RECORD_DELIM in the [GENERAL] section.

4. Label the dataset variables. It can be done with DATASET_VARS in the [GENERAL] section. It looks like

```
DATASET_VARS = {mHiggs, mTop}
```

The names should be comma-separated and be in braces. Make sure that the order is the same as the entries in rg.dat. You can use alphabets, numbers and underscore for the name, but it cannot start with a number. If you do not provide any dataset variables, set DATASET_VARS = {}.

5. Label the values in a record with RECORD_VARS in the [GENERAL] section, similarly as dataset variables. It looks like

```
RECORD_VARS = {Q, g2, g1, yt, yb, lambda}
```

6. Set the ratio between the renormalization scale, Q, and the instanton scale, R^{-1}. It can be set by the following entry in the [INITIALIZE] section.

```
LN_QR = 0
```

The above entry sets $\ln QR = 0$, *i.e.*, $Q = R^{-1}$.

7. Set the volume of the group space generated by the broken generators, $\ln \mathcal{V}_G$. It can be set with the following entry in the [INITIALIZE] section.

```
lnVg = log(2. * pi^2)
```

8. Set an upper bound on the region of integration over $\ln R^{-1}$. You can find the following entry in the [INITIALIZE] section.

```
upper_bound = log(2.435e18)
```

It sets upper bounds both on $\ln R^{-1}$ and $\ln \bar{\phi}_C$. The integral stops when either of these becomes larger than the Planck scale in the above example.

9. Set a lower bound on the region of integration over $\ln R^{-1}$.

```
lower_bound = log(mTop * 10)
```

It sets lower bounds both on $\ln R^{-1}$ and $\ln \bar{\phi}_C$. The integral starts when both of these becomes ten times larger than the top mass in the above example. This line is in the [BEGIN_ROUTINE] section because it uses a dataset variable, mTop. If you do not use any dataset variables in this line, you may move it to the [INITIALIZE] section.

10. In the following entry in the [MAIN_ROUTINE] section, describe the Higgs quartic coupling that is normalized as in eq. (1).

    ```
    HIGGS_QUARTIC_COUPLING = lambda
    ```

    You can use the variables in DATASET_VARS and RECORD_VARS. If needed, you can normalize it in this line like HIGGS_QUARTIC_COUPLING = lambda / 2.

11. In the following entry in the [MAIN_ROUTINE] section, describe the instanton scale, $\ln R^{-1}$, using LN_QR and the variables in DATASET_VARS and RECORD_VARS.

    ```
    LN_RINV = log(Q) - LN_QR
    ```

12. In the following entries in the [MAIN_ROUTINE] section, set totalQC and maxQC, which are $\sum_X [-\ln \mathcal{A}^{(X)}]_{\overline{\mathrm{MS}}}$ and $\max_X |[-\ln \mathcal{A}^{(X)}]_{\overline{\mathrm{MS}}}|$, respectively.

    ```
    higgsQC = HiggsQC()
    topQC = 3. * FermionQC(yt)
    WbosonQC = 2. * GaugeQC(g2^2 / 4.)
    ZbosonQC = GaugeQC((g2^2 + g1^2 * 3. / 5.) / 4.)

    totalQC = higgsQC + topQC + WbosonQC + ZbosonQC
    maxQC = max(abs(topQC), abs(WbosonQC), abs(ZbosonQC), abs(higgsQC))
    ```

    Here, you may use HiggsQC(), ScalarQC(kappa), FermionQC(y), and GaugeQC(g_squared) to calculate $[-\ln \mathcal{A}^{(X)}]_{\overline{\mathrm{MS}}}$ of the Higgs field, a scalar field with $\kappa =$ kappa, a pair of chiral fermions with $y =$ y, and a $U(1)$ gauge field with $g^2 =$ g_squared, respectively.

13. List up what you want to print on the output file.

    ```
    print(mHiggs, mTop, (lngamma + 378.229) / log(10))
    ```

    Here, lngamma is $\ln \gamma$, where γ is given in the same unit as R^{-4}. The delimiter can be set with OUTPUT_DELIM in the [GENERAL] section. You may optionally print a header by modifying

    ```
    print("mHiggs       mTop         log10(gamma x Gyr Gpc^3)")
    ```

    in the [INITIALIZE] section. You may change the decimal precision for the output with output_precision(n), which should be put in the [INITIALIZE] section.

14. Execute the program by typing

    ```
    ./elvas -o result.out model.in rg.dat
    ```

    in a terminal. The result will be written to result.out.

    You may provide multiple input files, which will be joined internally. Allowed options are

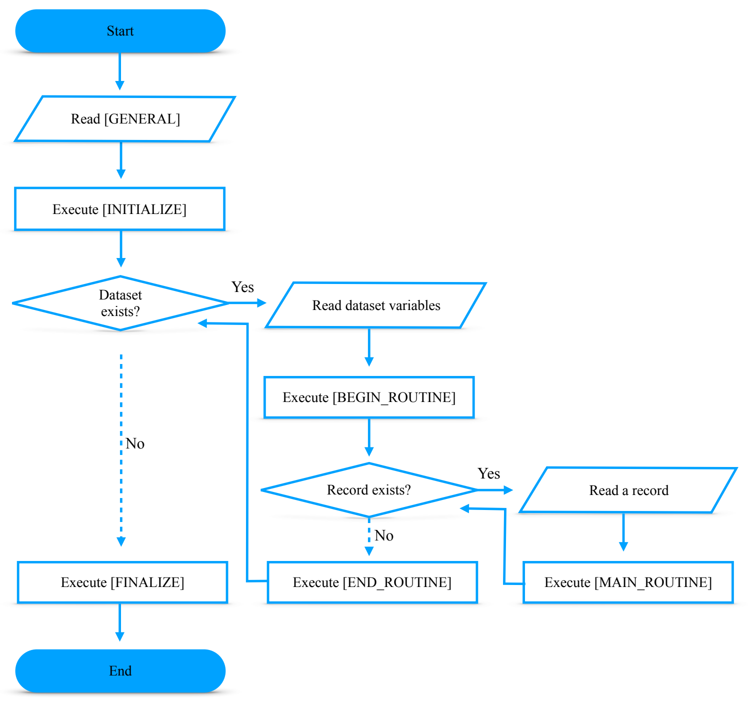

Figure 1: The flowchart of the interpreter.

-o output file

-h display help message

-v output version information

-n disable header printing

If input/output file is not supplied, the program use the standard input/output.

4.2 Using interpreter

The input of the interpreter has the following sections: `[GENERAL]`, `[INITIALIZE]`, `[BEGIN_ROUTINE]`, `[MAIN_ROUTINE]`, `[END_ROUTINE]`, `[FINALIZE]` and zero or more `[DATASET]`'s. You can omit any sections if you do not use them. As shown in the flowchart (Fig. 1), the interpreter first reads `[GENERAL]` and recognize the formats used in `[DATASET]` and output. Then, it executes `[INITIALIZE]`. When the `[DATASET]` section begins, `[BEGIN_ROUTINE]` is executed. At each record in the `[DATASET]` section, `[MAIN_ROUTINE]` is executed. After all the record in `[DATASET]` are evaluated, `[END_ROUTINE]` is executed and the interpreter will search for the next `[DATASET]` section. After all `[DATASET]`'s are evaluated, `[FINALIZE]` is executed and the program terminates.

The details of `[DATASET]` can be found in the previous subsection and those of the others are given below.

Section `[GENERAL]`

This section is for general settings. It should appear first in an input file if you use it.

If you have one or more dataset variables, you need to set

- `DATASET_DELIM = "delim"` – The delimiter for dataset variables.
- `DATASET_VARS = {dataset_var_name1, dataset_var_name2, ...}` – The names for dataset variables. You can use alphabets, numbers and underscore, but a name cannot start with a number. If you have no dataset variables, set it to `{}`.

If you have one or more records, you need to set

- `RECORD_DELIM = "delim"` – The delimiter for records.
- `RECORD_VARS = {data_var_name1, data_var_name2, ...}` – The names for variables in a record. You can use alphabets, numbers and underscore, but a name cannot start with a number.

If you output numbers, you need to set

- `OUTPUT_DELIM = "delim"` – The delimiter for output.

Section `[INITIALIZE]` `[BEGIN_ROUTINE]` `[MAIN_ROUTINE]` `[END_ROUTINE]` `[FINALIZE]`

In these sections, you can make your own routine with a simple grammar. We first list basic specifications.

- Input
  - The routine is evaluated line by line. If you want to do a line break, put `\` before you start a new line.
  - White spaces and tabs are ignored.
  - Any characters after `#` are ignored.
- Basic calculation
  - You may use `+`, `-`, `*`, `/`, and `^`, for addition, subtraction, multiplication, division, and power, respectively. You may use `pow(x,y)`, instead of `x^y`. If `y` is integer, `x^y` is faster than `pow(x,y)`.
  - For relational operators, you may use `<`, `>`, `<=`, `>=`, `==`, and `!=`, which correspond to $<$, $>$, $\leq$, $\geq$, $=$, and $\neq$, respectively. If the result is true, it returns one, otherwise it returns zero. Since numbers are processed as `double`, it is not preferred to use `x == y`. Instead, you may use `abs(x-y) < some_small_number`.

  - You may combine the relational conditions by using `&` and | as “and” and “or” operators, respectively.
  - Parentheses, “`(`” and “`)`”, can be used to group expressions.
  - Constant π is defined as `pi`.

- Functions
  - You may use the following elementary functions: `pow(x,y)`, `abs(x)`, `sqrt(x)`, `exp(x)`, `log(x)`, `log10(x)`, `sin(x)`, `cos(x)`, `tan(x)`, `asin(x)`, `acos(x)`, and `atan(x)`. Make sure that the result is always real.
  - The maximum and the minimum value can be obtained with `max(x,y,...)` and `min(x,y,...)`, respectively. The number of arguments should be equal or larger than two.

- Definition of constants and functions
  - You may define a constant with `=` operator. For example, `c = 2 * 3` sets 6 to `c`. You can use alphabets, numbers and underscore for the name of a constant, but it cannot start with a number. It returns the result of the right-hand side. Once it is defined, it remains until the program terminates.
  - You may define a function with `:=` operator. For example, `f(x):=x^2` defines a function `f(x)` that returns the square of `x`. You can use alphabets, numbers and underscore for the name of a function, but it cannot start with a number. Once it is defined, it remains until the program terminates. If the number of arguments is zero, you can define it either by `g():=...` or by `g:=...`, and call it either by `g()` or by `g`. (When there is a confliction between the names of a constant and a function, the constant overrides the function.)
  - It is allowed to use `:=` and multiple `=`'s at the same time. For example, `f(x):=m=x^2` sets the result of x^2 to `m` when it is evaluated.

- Utilities
  - Conditional branches can be expressed with `if(cond, x)` or `if(cond, x, y)`. If `cond` is equal or larger than 0.5, only `x` is evaluated and returns the result, otherwise only `y` is evaluated and returns the result. If `y` is not provided and `cond` is less than 0.5, it returns zero without evaluating `x`.
  - You may evaluate multiple expressions at the same time with `eval(expr1, expr2, ...)`. You can use any operators except for `:=` in the arguments. It returns the result of the last argument.
  - If you want to skip the rest of the entries in a section, use `continue()`.

  - If you want to skip a dataset, use `break()`. Once it is called, the interpreter skips the rest of the records in the current `[DATASET]` section and the coming `[END_ROUTINE]` section.
  - If `exit()` is called, the program terminates.

- Output
  - `print("string")` function prints `string` on the output file.
  - `print(num1, num2, ...)` function prints numbers on the output file. The delimiter can be set with the `OUTPUT_DELIM` entry in the `[GENERAL]` section. It returns the result of the last argument.
  - `output_precision(n)` sets the decimal precision for output to `n`.

In addition, there are constants and functions used for the calculation of vacuum decay rates. They should be set or called in right places.

■ In section `[INITIALIZE]`

const `LN_QR` – It fixes $\ln QR$, which should be a constant and relates the renormalization scale, Q, and the instanton scale, R.

■ In section `[BEGIN_ROUTINE]`

func `initialize()` – It clears the accumulated data of $\ln \bar{\phi}_C$ and $\ln d\gamma/dR^{-1}$. If you have multiple `[DATASET]`'s, this function must be called.

■ In section `[MAIN_ROUTINE]`

const `HIGGS_QURTIC_COUPLING` – The Higgs quartic coupling normalized as in eq. (1).

const `LN_RINV` – The instanton scale, $\ln R^{-1}$.

func `InstantonB()` – It returns the tree level action, $\mathcal{B}$. `HIGGS_QUARTIC_COUPLING` should be set before this function is called.

func `HiggsQC()` – It returns the quantum corrections from the Higgs fluctuations, $\left[-\ln \mathcal{A}^{(h)}\right]_{\overline{\mathrm{MS}}}$. `HIGGS_QUARTIC_COUPLING` and `LN_QR` should be set before this function is called.

func `ScalarQC(kappa)` – It returns the quantum corrections from the scalar fluctuations, $\left[-\ln \mathcal{A}^{(\sigma)}\right]_{\overline{\mathrm{MS}}}$ with $\kappa =$ `kappa`. `HIGGS_QUARTIC_COUPLING` and `LN_QR` should be set before this function is called.

func `FermionQC(y)` – It returns the quantum corrections from the fermion fluctuations, $\left[-\ln \mathcal{A}^{(\psi)}\right]_{\overline{\mathrm{MS}}}$ with $y =$ `y`. `HIGGS_QUARTIC_COUPLING` and `LN_QR` should be set before this function is called.

func `GaugeQC(g_squared)` – It returns the quantum corrections from the gauge fluctuations, $\left[-\ln \mathcal{A}^{(A,\varphi)}\right]_{\overline{\rm MS}}$ with g^2 = `g_squared`. `HIGGS_QUARTIC_COUPLING` and `LN_QR` should be set before this function is called.

func `save_phiC()` – It saves $\ln \bar{\phi}_C$ to memory. If you use `get_max_lnQ(x)` or `get_min_lnQ(x)`, you need to execute this function. `HIGGS_QURTIC_COUPLING` and `LN_RINV` should be set before this function is called.

func `save_dlngamma_dRinv(dlngamma)` – It saves $\ln d\gamma/dR^{-1}$ = `dlngamma` to memory. If you use `get_lngamma(x,y)`, you need to execute this function. `LN_RINV` should be set before this function is called.

■ In section `[END_ROUTINE]`

func `is_data_enough()` – It checks if there are enough data points of $\ln d\gamma/dR^{-1}$ to calculate the decay rate. It returns one if they are enough, or zero if not.

func `get_max_lnRinv(upper_bound)` – It returns the maximum allowed value of $\ln R^{-1}$ where both $\ln \bar{\phi}_C$ and $\ln R^{-1}$ are below `upper_bound`. If all the saved data points have smaller $\ln R^{-1}$, it returns the maximum $\ln R^{-1}$ among the data points.

func `get_min_lnRinv(lower_bound)` – It returns the minimum allowed value of $\ln R^{-1}$ where both $\ln \bar{\phi}_C$ and $\ln R^{-1}$ are above `lower_bound`. If all the saved data points have larger $\ln R^{-1}$, it returns the minimum $\ln R^{-1}$ among the data points.

func `get_lngamma(lnRinv_min,lnRinv_max)` – It interpolates the accumulated $\ln d\gamma/dR^{-1}$ and integrates over interval [`lnRinv_min`,`lnRinv_max`]. The return value is $\ln\gamma$. There should be a sufficient number of saved data that cover the region of integration.

4.3 Call routines in a c++ code

If you use our routines in your `c++` code, include `elvas.h` and compile `elvas.cpp` together with your code. The following routines are defined. They do not require the `boost` library.

- `Elvas::instantonB(lambdaAbs)`

  Calculate the tree level action of the instanton.

  in `const double& lambdaAbs`

  The value of $|\lambda|$.

  out The value of tree level action, $\mathcal{B}$.

- `Elvas::higgsQC(lambdaAbs, lnQR)`

  Calculate the quantum corrections from the Higgs field.

in `const double& lambdaAbs`
    The value of $|\lambda|$.

in `const double& lnQR`
    The value of $\ln QR$.

out The value of $\left[-\ln \mathcal{A}'^{(h)}\right]_{\overline{\mathrm{MS}}}$.

- `Elvas::scalarQC(kappa, lambdaAbs, lnQR)`

  Calculate the quantum corrections from a scalar.

  in `const double& kappa`
      The value of κ.

  in `const double& lambdaAbs`
      The value of $|\lambda|$.

  in `const double& lnQR`
      The value of $\ln QR$.

  out The value of $\left[-\ln \mathcal{A}^{(\sigma)}\right]_{\overline{\mathrm{MS}}}$.

- `Elvas::fermionQC(y, lambdaAbs, lnQR)`

  Calculate the quantum corrections from a pair of chiral fermions.

  in `const double& y`
      The value of y.

  in `const double& lambdaAbs`
      The value of $|\lambda|$.

  in `const double& lnQR`
      The value of $\ln QR$.

  out The value of $\left[-\ln \mathcal{A}^{(\psi)}\right]_{\overline{\mathrm{MS}}}$.

- `Elvas::gaugeQC(gSquared, lambdaAbs, lnQR)`

  Calculate the quantum corrections from a $U(1)$ gauge boson.

  in `const double& gSquared`
      The value of g^2.

  in `const double& lambdaAbs`
      The value of $|\lambda|$.

  in `const double& lnQR`
      The value of $\ln QR$.

  out The value of $\left[-\ln \mathcal{A}^{(A,\varphi)}\right]_{\overline{\mathrm{MS}}}$.

- `Elvas::lnPhiC2LnRinv(lnPhiC, lnPhiC2lnRinv)`
  Convert $\ln \bar{\phi}_C$ to $\ln R^{-1}$.

  in `const double& lnPhiC`
  The value of $\ln \bar{\phi}_C$.

  in, out `vector<pair<double, double>>&lnPhiC2lnRinv`
  A vector of $(\ln \bar{\phi}_C, \ln R^{-1})$.

  out The corresponding value of $\ln R^{-1}$.

- `Elvas::getLnGamma(lndgam, lnRinvBeg, lnRinvEnd)`
  Integrate over $\ln R^{-1}$, using accumulated data of $\ln \frac{d\gamma}{d \ln R^{-1}}$.

  in, out `vector<pair<double, double>>& lndgam`
  A vector of $\left(\ln R^{-1}, \ln \frac{d\gamma}{d \ln R^{-1}}\right)$.

  in `const double& lnRinvBeg`
  The lower boundary of the region of integration.

  in `const double& lnRinvEnd`
  The upper boundary of the region of integration.

  out The value of $\ln \gamma$.

A Example model file

```
##########################################################################
#
#
#                     Input File for the Standard Model
#
#
##########################################################################

[GENERAL]
#Labels for dataset variables
DATASET_VARS = {mHiggs, mTop}

#Labels for RG data
RECORD_VARS = {Q, g2, g1, yt, yb, lambda}

#Delimiter for dataset variables
DATASET_DELIM = " "

#Delimiter for RG data
RECORD_DELIM = " "
```

```
#Delimiter for output
OUTPUT_DELIM = " "

[INITIALIZE]
#Set ln(Q x R)
LN_QR = 0.

#Volume of the group space generated by the broken generators
lnVg = log(2. * pi^2)

#Upper bound on lnPhiC and lnR^(-1)
upper_bound = log(2.435e18)

#Print a header
print("mHiggs       mTop          log10(gamma x Gyr Gpc^3)")

[BEGIN_ROUTINE]
#Clear phiC and dlngamma/dR^(-1)
initialize()

#Lower bound on lnPhiC and lnR^(-1)
lower_bound = log(mTop * 10)

[MAIN_ROUTINE]
#The Higgs quartic coupling.
HIGGS_QUARTIC_COUPLING = lambda

#If Higgs quartic coupling is positive, skip this record
if(HIGGS_QUARTIC_COUPLING > 0, continue())

#The instanton scale ln R^(-1)
LN_RINV = log(Q) - LN_QR

#If ln R^(-1) is away from the region of integration, skip this record
if(LN_RINV > upper_bound + log(10) | LN_RINV < lower_bound - log(10),\
   continue())

#Calculate the tree level action
tree = InstantonB()

#Calculate the quantum corrections
higgsQC = HiggsQC()
topQC = 3. * FermionQC(yt)
WbosonQC = 2. * GaugeQC(g2^2 / 4.)
ZbosonQC = GaugeQC((g2^2 + g1^2 * 3. / 5.) / 4.)

totalQC = higgsQC + topQC + WbosonQC + ZbosonQC
maxQC = max(abs(topQC), abs(WbosonQC), abs(ZbosonQC), abs(higgsQC))
```

```
#If quantum corrections are too large, skip this record
if(maxQC > 0.8 * tree | abs(totalQC) > 0.8 * tree, continue())

#Save phiC and dlngamma/dR^(-1) to memory
save_phiC()
save_dlngamma_dRinv(lnVg + 4. * LN_RINV - tree - totalQC)

[END_ROUTINE]
#Calculate ln gamma
lngamma = \
if(is_data_enough(), \
  eval( \
    lnRinv_minimum = get_min_lnRinv(lower_bound), \
    lnRinv_maximum = get_max_lnRinv(upper_bound), \
    if(lnRinv_maximum > lnRinv_minimum, \
      get_lngamma(lnRinv_minimum, lnRinv_maximum), \
      -inf \
    ) \
  ), \
  -inf \
)

#Output
print(mHiggs, mTop, (lngamma + 378.229) / log(10))
```

References

[1] S. Chigusa, T. Moroi and Y. Shoji, Phys. Rev. Lett. **119** (2017) no.21, 211801 doi:10.1103/PhysRevLett.119.211801 [arXiv:1707.09301 [hep-ph]].

[2] S. Chigusa, T. Moroi and Y. Shoji, Phys. Rev. D **97** (2018) no.11, 116012 doi:10.1103/PhysRevD.97.116012 [arXiv:1803.03902 [hep-ph]].

[3] P. Baratella, M. Nemevšek, Y. Shoji, K. Trailović and L. Ubaldi, [arXiv:2406.05180 [hep-ph]].