Abstract Several extensions of the Standard Model of particle physics contain additional scalars implying a more complex scalar potential compared to that of the Standard Model. In general these potentials allow for charge- and/or color-breaking minima besides the desired one with correctly broken $SU(2)_L \times U(1)_Y$. Even if one assumes that a metastable local minimum is realized, one has to ensure that its lifetime exceeds that of our universe. We introduce a new program called Vevacious which takes a generic expression for a one-loop effective potential energy function and finds all the tree-level extrema, which are then used as the starting points for gradient-based minimization of the one-loop effective potential. The tunneling time from a given input vacuum to the deepest minimum, if different from the input vacuum, can be calculated. The parameter points are given as files in the SLHA format (though is not restricted to supersymmetric models), and new model files can be easily generated automatically by the Mathematica package SARAH. This code uses HOM4PS2 to find all the minima of the tree-level potential, PyMinuit to follow gradients to the minima of the one-loop potential, and CosmoTransitions to calculate tunneling times.

1 Introduction

A major part of the phenomenology of the incredibly successful standard model of particle physics (SM) is the spontaneous breaking of some (but not all) of the gauge symmetries of the Lagrangian density by the vacuum expectation value (VEV) of a scalar field charged under a subgroup $SU(2)_L$ of the SM gauge group. The entire scalar sector of the SM consists of a doublet of $SU(2)_L$ which also has a hypercharge under $U(1)_Y$ equal in magnitude to that of the lepton $SU(2)_L$ doublet. The potential energy of the vacuum is minimized by the scalar field taking a constant non-zero value everywhere. The presence of this VEV radically changes the phenomenology of the theory, and allows for masses for particles that would be forced to be massless if the gauge symmetries of the Lagrangian density were also symmetries of the vacuum state.

Since this scalar field is the only field in the SM that can possibly have a non-zero VEV while preserving Lorentz invariance, finding the minima of the potential energy is straightforward, though of course evaluating it to the accuracy required is quite involved [1–3].

Also, with the current measurements for the masses of the top quark and Higgs boson, one finds that the SM potential at one-loop order is actually unbounded from below for a fixed value of the renormalization scale. The value of the Higgs field for which the potential is lower than the desired vacuum is so high that one may worry that large logarithms of the Higgs field over the electroweak scale would render the loop expansion non-convergent. However, the effect of large logarithms can be resummed, and the conclusion that our vacuum is only metastable persists using the renormalization-group-improved effective potential [1–5].

The existence of multiple non-equivalent vacua both raises technical challenges and introduces interesting physics. The technical challenges are now that one has to find several minima and evaluate which is the deepest, as well as calculate the tunneling time from a false vacuum to the true vacuum. However, this is an important ingredient in theories where a first-order phase transition explains the baryon asymmetry of the universe through the sphalerons occurring in the nucleation of bubbles of true vacuum (see [6] and references therein).
Many extensions of the SM introduce extra scalar fields. Sometimes these fields are introduced explicitly to spontaneously break an extended gauge symmetry down to the SM gauge group \([7, 8]\), and they are assumed to have non-zero VEVs at the true vacuum of the theory. Other times they are introduced for other reasons, such as supersymmetry \([9]\), and often non-zero VEVs for such fields would be disastrous, such as breaking \(SU(3)_C\) and/or \(U(1)_{EM}\), which excludes certain parts of the parameter space of the minimal supersymmetric standard model (MSSM) from being phenomenologically relevant.

The technical challenges are much tougher when multiple scalar fields are involved. Even a tree-level analysis involves solving a set of coupled cubic equations, the so-called minimization or tadpole equations. It has generally only been attempted for highly symmetric systems such as two Higgs doublet models (2HDM) \([10, 11]\) or with only a minimal amount of extra degrees of freedom such as the (assumed) three non-zero VEVs of the next-to-minimal supersymmetric standard model (NMSSM) \([12–14]\).

Since a general solution is usually too difficult, the question of the stability of VEV configurations against tunneling to other minima of the potential is often ignored. Instead, potentials are often engineered to have a local minimum at a desired VEV configuration through ensuring that the tadpole equations are satisfied for this set of VEVs. This approach allows one to go beyond tree level straightforwardly, and one-loop tadpoles are the norm, and in supersymmetric models two-loop contributions are often included \([15]\). This local minimum is implicitly assumed to be stable or long-lived enough to be physically relevant. Unfortunately, as some examples will show, local minima which are not the global minimum of their parameter point are often extremely short-lived, excluding some benchmark parameter points for some models.

The program Vevacious has been written to address this. Given a set of tadpole equations and the terms needed to construct the one-loop effective potential, first all the extrema of the tree-level potential are found using homotopy continuation (HOM4PS2), which are then used as starting points for gradient-based minimization (PyMinuit) of the (real part of the) one-loop potential, and finally, if requested, the tunneling time from an input minimum to the deepest minimum found is estimated at the one-loop level (CosmoTransitions). The program is intended to be suitable for parameter scans, taking parameter points in the SLHA format \([16, 17]\) and giving a result within seconds, depending on the number of fields allowed to have non-zero VEVs and the accuracy of the tunneling time required. Vevacious is available to download from http://www.hepforge.org/downloads/vevacious.

### 2 The potential energy function at tree level and one-loop level

The terminology of minimizing the effective potential of a quantum field theory is rather loaded. Hence first we shall clarify some terms and conventions that will be used in the rest of this article. In the following we consider models where only scalars can get a VEV as required by Lorentz invariance.

In principle, the effective potential is a real-valued\(^2\) functional over all the quantum fields of the model. However, under the assumption that the vacuum is homogeneous and isotropic, for the purposes of determining the vacua of the model, the effective potential can be treated as a function of sets of (dimensionful) numbers, which we shall refer to as field configurations. Each field configuration is a set of variables which correspond to the classical expectation values for the spin-zero fields which are constant with respect to the spatial co-ordinates.

The example of the SM is relatively simple: the field configuration is simply a set of two complex numbers, which are the values of the neutral and charged scalar fields assuming that each is constant over all space. These four real degrees of freedom can be reduced to a single degree of freedom by employing global \(SU(2)_L\) and phase rotations, leaving an effective potential that is effectively a function of a single variable.

Henceforth we shall assume that each complex field is treated as a pair of real degrees of freedom, and note that this may obscure continuous sets of physically equivalent degrees of freedom which are manifestly related by phase rotations when expressed with complex fields.

Also we shall refer to the local minima of the effective potential as its vacua, and label the global minimum as the true vacuum, while all the others are false vacua. A potential may have multiple true vacua, either as a continuous set of minima related by gauge transformations as in the SM for example, or a set of disjoint, physically inequivalent minima, each of which may of course be a continuous set of physically equivalent minima themselves. In cases where there is a continuous set of physically equivalent minima, we assume that a single exemplar is taken from the set for the purposes of comparison of physically inequivalent minima.

Furthermore, the term vacuum expectation value can be used in many confusing ways. In this work, VEVs will only

---

\(^1\)The potential may include non-renormalizable terms, as long as the one-loop effective potential is of the form of a polynomial plus \(V^{mass}\) as given in Eq. (4).

\(^2\)As noted in Sect. 2.4, the loop expansion may lead to complex values for the one-loop effective potential. See \([18, 19]\) for detailed discussions.
refer to the sets of constant values which the scalar fields have at the field configurations which minimize the effective potential. Hence we do not consider the effective potential to be a “function of the VEVs", rather a function of a set of numbers that we call a field configuration.

2.1 The tree-level potential and tadpole equations

If one merely considers the SM at tree level, minimizing the potential is straightforward. A global SU(2)_L rotation can bring the part of the potential due to the scalar doublet into the form

\[ V = \frac{2}{4}|\phi|^4 + \frac{\mu^2}{2} |\phi|^2 \]  

(1)

where \( \phi \) is the neutral component of the SU(2)_L doublet. After a little differentiation and algebra, one finds that if \( \lambda > 0, \mu^2 < 0 \), then the potential is minimized for \( |\phi| = v = \sqrt{-\mu^2/\lambda} \).

However, for a set of \( N \) real scalar degrees of freedom \( \phi_i \) (writing complex scalar fields as two separate real scalars), the scalar part of the tree-level potential of a renormalizable quantum field theory in four space-time dimensions is of the form

\[ V_{\text{tree}} = \lambda_{ijkl} \phi_i \phi_j \phi_k \phi_l + A_{ijkl} \phi_i \phi_j \phi_k + \mu_{ij}^2 \phi_i \phi_j \]

(2)

which, when differentiated with respect to the \( N \) independent \( \phi_i \), yields \( N \) polynomial equations of up to degree three. We have assumed that any terms linear in the fields have been removed by constant shifts of the fields.

Although we assume renormalized potentials here for simplicity, the methods used by Vevacious are equally applicable to non-renormalizable potentials, as long as \( V_{\text{tree}} \) is expressed as a finite-degree polynomial. The value of loop corrections to a non-renormalizable potential may be debated, but Vevacious can be restricted to using just the tree-level potential.

While closed-form solutions for cubic polynomials in one variable exist, solving a coupled system in general requires very involved algorithms, such as using Gröbner bases to decompose the system [20, 21], or homotopy continuation to trace known solutions of simple systems as they are deformed to the complicated target system of tadpole equations.

2.1.1 The homotopy continuation method

The homotopy continuation method [22, 23] has found use in several areas of physics [24–26], in particular to find string theory vacua [27, 28] and extrema of extended Higgs sectors [29], where the authors investigated a system of two Higgs doublets with up to five singlet scalars in a general tree-level potential, and [30], where systems of up to ten fields were allowed to have non-zero VEVs. In contrast, the Gröbner basis method is deemed prohibitively computationally expensive for systems involving more than a few degrees of freedom [20].

The numerical polyhedral homotopy continuation method is a powerful way to find all the roots of a system of polynomial equations quickly [31]. Essentially it works by continuously deforming a simple system of polynomial equations with known roots, with as many roots as the classical Bézout bound of the system that is to be solved (i.e. the maximum number of roots it could have). The simple system with known roots is continuously deformed into the target system, with the position of the roots updated with each step. While the method is described in detail in [22, 23], a light introduction can be found for example in [29].

2.2 The one-loop potential

The general form of the renormalized one-loop effective potential [32, 33] is

\[ V^{1\text{-loop}} = V_{\text{tree}} + V_{\text{counter}} + V_{\text{mass}} \]  

(3)

where \( V_{\text{tree}} \) is as above and \( V_{\text{counter}} \) has the same form as \( V_{\text{tree}} \), i.e. a polynomial of the same degree in the same fields, but the coefficients are instead the renormalization-dependent finite parts of the appropriate counterterms [32]. The term \( V_{\text{mass}} \) has the form, for a given field configuration \( \Phi \),

\[ V_{\text{mass}} = \frac{1}{64\pi^2} \sum_n \left[ \log\left( \frac{\bar{M}^2_n(\Phi)}{Q^2} \right) - c_n \right] \]

(4)

where the sum runs over all real scalar, Weyl fermion, and vector degrees of freedom, with \( s_n \) being the spin of the degree of freedom. Complex scalars and Dirac fermions are accounted for as mass-degenerate pairs of real scalars and Weyl fermions respectively.

For scalar degrees of freedom, the \( \bar{M}^2_n(\Phi) \) are the eigenvalues of the second functional derivative of \( V_{\text{tree}} \), i.e. the eigenvalues of \( \bar{M}^2_{\text{tree}}(\Phi) = \lambda_{ijkl} + \lambda_{ikjl} + \lambda_{iijkl} + \cdots \phi_i \phi_j + (A_{ijkl} + A_{ikjl} + A_{ijkl} + \cdots)\phi_i \phi_j + \mu_{ij}^2 + \mu_{ij}^2 \). Thus these \( \bar{M}^2_n(\Phi) \) are the eigenvalues of the tree-level scalar “mass-squared matrix” that would be read off the Lagrangian with the scalars written as fluctuations around the field configuration. Unless the field configuration corresponds to a minimum of the effective potential, these do not correspond to physical masses in any way, of course.

Likewise, the \( \bar{M}^2_n(\Phi) \) for fermionic and vector degrees of freedom are the eigenvalues of the respective “mass-squared matrices” where the scalar fields are taken to have constant values given by the field configuration. (The fermion mass-squared matrix is given by the mass matrix multiplied by its Hermitian conjugate.)
The terms $c_n$ depend on the regularization scheme. In the \( \overline{\text{MS}} \) scheme, \( c_n \) is 3/2 for scalars and Weyl fermions, but 5/6 for vectors, while in the \( \overline{\text{DR}} \) scheme [33, 34], more suitable for supersymmetric models, \( c_n \) is 3/2 for all degrees of freedom. Since this is a finite-order truncation of the expression, the renormalization scale \( Q \) also appears explicitly in the logarithm, as well as implicitly in the scale dependence of the renormalized Lagrangian parameters.

Much of the literature on one-loop potentials (including [33]) assumes a renormalization scheme where \( V^{\text{counter}} \) is zero; however, such a scheme is often inconvenient for other purposes, such as ensuring tadpole equations have a given solution at the one-loop level, see e.g. the appendix of [35]. \( \text{(SARAH automatically generalizes this approach to extended SUSY models as explained in [36, 37]).} \) Finally, we also note that models can be constructed where spontaneous symmetry breaking does not happen at tree level, but does exist when one takes loop corrections into account [18, 38].

2.2.1 Scale dependence

As noted, the one-loop effective potential depends on the renormalization scale. Ideally one would use the “renormalization-group improved” expression for the potential [32] as this is invariant under changes of scale; however, this is often totally impractical except for potentials with only a handful of parameters and a single scalar field.

If one must use a scale-dependent expression for the potential, as is often the case, the renormalization scale should be chosen carefully: if one chooses a scale too high or too low, one may find that with a finite-loop-order (and thus scale-dependent) effective potential, there is no spontaneous breaking of any symmetry, or even that the potential is not bounded from below [39]! This is often simply due to the fact that higher orders become more important in such a case, especially when corrections from the next order would introduce new, large couplings, such as often happens when going from tree level to one loop. It can also be that the scale is so large or small that the loop expansion is no longer a reliable expansion. We also note that rather undermines arguments that radiative effects do not change tree-level conclusions on the absolute stability of vacua such as in [40] (where the argument also fails to take into account that there may not even be a scale at which the renormalization condition used can be satisfied).

Indeed, it is crucial that the scale is chosen so that the loop expansion is valid. Explicitly, large logarithms should not spoil the perturbativity of the expansion in couplings. Loops with a particle \( n \) typically come with a factor of \( \ln(\frac{M_n^2(\Phi)}{Q^2}) \) along with the factor of \( \alpha_n = \frac{\text{[relevant coupling]}}{4\pi} \), and thus \( \alpha_n \ln(\frac{M_n^2(\Phi)}{Q^2}) \) should remain sufficiently smaller than one such that the expansion can be trusted [32]. A rough first estimate then of the region of validity, assuming that the dimensionful Lagrangian parameters are all of the order of the renormalization scale to some power, is where \( \ln(\frac{\nu^2}{Q^2})/(4\pi) \leq 1/2 \), say, for a field configuration with vector length \( \nu \), so where the VEVs are within a factor of \( e^{\alpha_n} \approx 20 \) of the renormalization scale.

Furthermore, it is in general not valid to compare the potential for different field configurations using a different scale for each configuration, if one is using a scale-dependent effective potential. The reason is that there is an important contribution to the potential that is field-independent yet still depends on the scale\(^3\) [41, 42]. Of course, if one knows the full scale dependence of all the terms of the Lagrangian regardless of whether they lead to field-dependent contributions to the effective potential, then one can correctly evaluate different field configurations at different scales.

2.3 Gauge dependence

The one-loop potential is explicitly gauge-dependent [43, 44]. However, as shown in [44, 45], the values it takes at its extrema are independent of the gauge chosen, except for spurious extrema of poorly-chosen gauges. The popular \( R_\xi \) gauges are well-behaved and do not have fictitious gauge-dependent extrema for reasonable choices of \( \xi \) [45].

It is also possible to formulate the effective potential in terms of gauge-invariant composite fields [46], though this may not always be practical. One can also verify the gauge-independence of extrema using more complicated gauges and applying BRST invariance [47].

2.4 Convexity

As shown in [18], the effective potential, which can be thought of as the quantum analogue of the classical potential energy for constant fields, is real for all values of the fields. However, the loop expansion leads to complex values in regions where the classical potential is non-convex. While one can take the convex hull of the truncated expansion of the potential when evaluating the potential for configurations of fields in the convex region, it is not particularly helpful for the purpose of computing tunneling transition times. Fortunately, the one-loop truncation of the effective potential as a function of constant values for the fields can consistently be interpreted as a complex number with real part giving the expectation value of the potential energy density for the given field configuration and imaginary part proportional to the decay rate per unit volume of this configuration [19].

\(^3\)For example, if one is comparing two field configurations of the MSSM potential where the squark fields happen to have zero values, the mass of the gluino is independent of the non-zero fields and yet provides a scale-dependent contribution to the effective potential.
2.5 Comparing two vacua

If there are two or more physically inequivalent minima of a potential, then it is vitally important to know if the phenomenologically desired minimum is the global minimum, or, if not, how long the expected tunneling time to the true vacuum is.

Given the issues raised above in Sects. 2.2.1 and 2.3, it is safe to use a scale- and gauge-dependent one-loop effective potential to compare two inequivalent minima provided that the scale is held fixed and that the two minima are within the region of validity determined by the renormalization scale. Of course, an explicitly gauge- and scale-independent expression for the effective potential would obviously be unburdened by such concerns, but unfortunately it is rare to be able to formulate such an expression.

3 Tunneling from false vacua to true vacua

The usual expression for the decay rate $\Gamma$ per unit volume for a false vacuum is given in [48, 49] as

$$\Gamma/\text{vol.} = A e^{(B - t/\hbar)} (1 + O(\hbar))$$

(5)

where $A$ is a factor which depends on eigenvalues of a functional determinant and $B$ is the bounce action. The $A$ factor is typically estimated on dimensional grounds as it is very complicated to calculate and, because of the exponentiation of $B$, is far less important than getting the bounce action as accurate as possible. If $A$ is taken to be $O((100-1000 \text{ GeV})^4)$, then for $\Gamma/\text{vol.}$ to be roughly the age of the known Universe to the fourth power, $B$ must be around $400\hbar$, and a per-cent variation in $B$ leads to a factor of $e$ variation in the tunneling time.

Given a path through the field configuration space from one vacuum to another with a lower value, which for convenience we shall label as the false vacuum and true vacuum respectively, one can solve the equations of motion for a bubble of true vacuum of critical size in an infinite volume of false vacuum [48, 50]. This allows one to calculate the bounce action and thus the major part of the tunneling time.

Unfortunately, this means that to calculate the tunneling time from a false vacuum to a true vacuum, one needs to evaluate the potential along a continuous path through the field configuration space, and even though the extrema of the potential are gauge-invariant as noted above, the paths between them are not. However, it has been proved that at zero temperature, the gauge dependence at one-loop order cancels out [51].

At finite temperature, the situation is not so clear, though the Landau gauge may be most appropriate [52]. While some studies have shown that for “reasonable” choices of gauge, the differences in finite-temperature tunneling times are small [53], it is still possible to choose poor gauges that can even obscure the possibility of tunneling [54].

4 Vevacious: objectives, outline, features and limitations

4.1 Objectives

The Vevacious program is intended as a tool to quickly evaluate whether a parameter point with a given set of VEVs, referred to henceforth as the input vacuum, has, to one-loop order, any vacua with lower potential energy than the input vacuum, and, optionally, to estimate the tunneling time from the input vacuum to the true vacuum if so.

A typical use envisaged is a parameter scan for a single model. Some effort needs to be put into creating the model file in the first place, though this is straightforward if using SARAH as described in Sect. 6.1; once the model file has been created, parameter points given in the form of SLHA files should be evaluated within a matter of seconds, depending on how complicated the model is, what simplifications have been made, and how accurately the tunneling time should be calculated if necessary.

Given a model (through a model file) and a parameter point (through an SLHA file), Vevacious determines the global minimum of the one-loop effective potential, and a verdict on whether the input minimum is absolutely stable, by it being the global minimum, or metastable. The user provides a threshold for which the metastability is rated as long-lived or short-lived. Whether the tunneling time or just an upper bound is calculated depends on whether the upper bound is above or below the threshold, or may be forced by certain options (see Sect. 6.3).

4.2 Outline

Here we present the steps taken by Vevacious, which are schematically shown in Fig. 1.

1. An input file in the SLHA format [16] is read in to obtain the Lagrangian parameters defining the parameter point, required to evaluate the potential. We emphasize that even though the SUSY Les Houches Accord is used as the format, the model itself does not need to be supersymmetric, as long as the SLHA file contains appropriate blocks.

2. All the extrema of the tree-level potential are found using the homotopy continuation method [22] to solve the tree-level tadpole equations. The publicly available program HOM4PS2 [31] is used for this.

3. The tree-level extrema are used as starting points for gradient-based minimization of the one-loop effective potential. The MINUIT algorithms [55] are used

---

4As mentioned in Sect. 2.2.1, a renormalization-group-improved effective potential would be better than the one-loop effective potential, but at the moment it seems infeasible to implement.
The member functions of VevaciousRunner are shown from top to bottom in the order in which they are called by Vevacious.exe, as can be seen by looking into the Vevacious.cpp source file. All steps labeled with † are elaborated in the text. (Most file names can be changed, see Sect. 6.3)

here through the Python wrapper PyMinuit [56]. The points where PyMinuit stops are checked to see if they are really minima. Any saddle point is then split into two further points, displaced from the original in the directions of steepest descent by amounts given by the <saddle_nudges> arguments (see Sect. 6.3), which are then also used as starting points for PyMinuit. By default, PyMinuit is restricted to a hypercube of field configurations where each field is only allowed to have a magnitude less than or equal to one hundred times the renormalization scale of the SLHA file. This is rather

5PyMinuit stops when it has found a sufficiently flat region without checking whether it is at a minimum.
excessive by the reasoning of Sect. 2.2.1, which would lead one to take at most maybe ten or twenty times the scale as an upper limit; however, it was considered better to allow the user to decide whether the results of Vevacious are within a trustworthy region.

(4) The minima are sorted, and, if necessary, the tunneling time from the input vacuum to the true vacuum is calculated. The $A$ factor of Eq. (5) is taken to be equal to the fourth power of the renormalization scale, as this is expected to be the typical scale of the potential and thus the expected scale of the solitonic solutions, and the bounce action is calculated with the code CosmoTransitions [50]. To save time, first CosmoTransitions is called to calculate the bounce action with a bubble profile given by a straight line in field configuration space from the false vacuum to the true vacuum to get an upper bound on the tunneling time. If this upper bound is below the user-given threshold $<\text{direct\_time}>$ (see Sect. 6.3), then no refinement is pursued. If, however, the upper bound is above the threshold, the bounce action is calculated again allowing CosmoTransitions to deform the path in field configuration space to find the minimal surface tension for the bubble. If one wishes to calculate the tunneling time with a different $A$ factor, one can edit a line of Python code as described in Sect. 4.3.

(5) The results are printed in a results file and also appended to the SLHA input file.

4.3 Features

Finds all tree-level extrema The homotopy continuation method is guaranteed to find all the solutions of the system of tadpole equations (to the limitations of the finite precision of the machine following the algorithm) [22]. One does not have to worry that there may be solutions just beyond the range of a scan looking for the solutions.

Rolls to one-loop minima Vevacious rolls from the tree-level extrema to the minima of the one-loop effective potential before comparing them, because in general the VEVs get shifted. In addition, extrema that change their nature with radiative corrections, such as the field configuration with zero values for all the fields in the Coleman–Weinberg model of radiative spontaneous symmetry breaking [38], which is a minimum of the tree-level potential but a local maximum of the one-loop effective potential, are found.

Calculates tunneling times or upper bounds on them A parameter point in a model is not necessarily ruled out on the basis that the desired minimum of the potential is not the global minimum, since a metastable configuration with a lifetime of roughly the observed age of the Universe or longer is compatible with the single data point that we have. Vevacious creates CosmoTransitions objects with its effective potential function to evaluate the bounce action and thus the tunneling time from a false vacuum to the true vacuum of a parameter point.

Fast An important aspect of Vevacious is that it is fast enough to be used as a check in a parameter scan of a model. For example, on a laptop with a 2.4 GHz processor, a typical parameter point for the MSSM allowing six real non-zero VEVs (two Higgs, two stau, two stop) can report within 3.2 seconds that no deeper vacuum than the input vacuum was found, or, for a different parameter point, can report an upper bound on the tunneling time within 18 seconds. However, borderline cases which require a full calculation of the minimal bounce action can take up to 500 seconds. Reducing the number of degrees of freedom to four (fixing the stop values at zero) reduces the calculation times to 0.6, 2.3 and 27 seconds respectively.

Flexible Vevacious has been written in a way that should allow useful customizations with small changes to the main Python code. For example, one can change a single line (line 36) in Vevacious.py so that the tree-level potential is used for the analysis rather than the one-loop effective potential:

```python
effectivePotentialFunction = VPD.TreeLevelPotential
```

and no further changes are necessary (line breaks should not be there in the Python: they are only here so that these lines fit this text). Vevacious.exe does not overwrite Vevacious.py, so any changes to the Python code will be kept. This was chosen as the best compromise to allow non-trivial changes without forcing the user to go very deep into the code, though it does rely on the user learning some Python to be able to do so. Another customization that one may wish to make could be to change the $A$ factor for the calculation of the tunneling time (and hence the thresholds for the bounce action calculations). This would be done by editing line 433 of Vevacious.py to fix the fourth root of $A$ from the renormalization scale:

```python
fourthRootOfSolitonicFactorA = VPD.energyScaleFourth
```

Fast An important aspect of Vevacious is that it is fast enough to be used as a check in a parameter scan of a model. For example, on a laptop with a 2.4 GHz processor, a typical parameter point for the MSSM allowing six real non-zero VEVs (two Higgs, two stau, two stop) can report within 3.2 seconds that no deeper vacuum than the input vacuum was found, or, for a different parameter point, can report an upper bound on the tunneling time within 18 seconds. However, borderline cases which require a full calculation of the minimal bounce action can take up to 500 seconds. Reducing the number of degrees of freedom to four (fixing the stop values at zero) reduces the calculation times to 0.6, 2.3 and 27 seconds respectively.

Flexible Vevacious has been written in a way that should allow useful customizations with small changes to the main Python code. For example, one can change a single line (line 36) in Vevacious.py so that the tree-level potential is used for the analysis rather than the one-loop effective potential:

```python
effectivePotentialFunction = VPD.TreeLevelPotential
```

and no further changes are necessary (line breaks should not be there in the Python: they are only here so that these lines fit this text). Vevacious.exe does not overwrite Vevacious.py, so any changes to the Python code will be kept. This was chosen as the best compromise to allow non-trivial changes without forcing the user to go very deep into the code, though it does rely on the user learning some Python to be able to do so. Another customization that one may wish to make could be to change the $A$ factor for the calculation of the tunneling time (and hence the thresholds for the bounce action calculations). This would be done by editing line 433 of Vevacious.py to fix the fourth root of $A$ from the renormalization scale:

```python
fourthRootOfSolitonicFactorA = VPD.energyScaleFourth
```

fast An important aspect of Vevacious is that it is fast enough to be used as a check in a parameter scan of a model. For example, on a laptop with a 2.4 GHz processor, a typical parameter point for the MSSM allowing six real non-zero VEVs (two Higgs, two stau, two stop) can report within 3.2 seconds that no deeper vacuum than the input vacuum was found, or, for a different parameter point, can report an upper bound on the tunneling time within 18 seconds. However, borderline cases which require a full calculation of the minimal bounce action can take up to 500 seconds. Reducing the number of degrees of freedom to four (fixing the stop values at zero) reduces the calculation times to 0.6, 2.3 and 27 seconds respectively.

Flexible Vevacious has been written in a way that should allow useful customizations with small changes to the main Python code. For example, one can change a single line (line 36) in Vevacious.py so that the tree-level potential is used for the analysis rather than the one-loop effective potential:

```python
effectivePotentialFunction = VPD.TreeLevelPotential
```

and no further changes are necessary (line breaks should not be there in the Python: they are only here so that these lines fit this text). Vevacious.exe does not overwrite Vevacious.py, so any changes to the Python code will be kept. This was chosen as the best compromise to allow non-trivial changes without forcing the user to go very deep into the code, though it does rely on the user learning some Python to be able to do so. Another customization that one may wish to make could be to change the $A$ factor for the calculation of the tunneling time (and hence the thresholds for the bounce action calculations). This would be done by editing line 433 of Vevacious.py to fix the fourth root of $A$ from the renormalization scale:

```python
fourthRootOfSolitonicFactorA = VPD.energyScaleFourth
```

can be changed to

```python
fourthRootOfSolitonicFactorA = 246.0
```

to change the $A$ factor to (246 GeV)$^4$ for example, if one feels that the electroweak scale is a more appropriate choice.
4.4 Limitations

**Garbage in, garbage out**  Vevacious performs very few sanity checks, so rarely protects the user from their own mistakes. For instance, Vevacious does not check if the potential is bounded from below. However, there are some checks, such as those which result in the warning that the given input minimum was actually rather far from the nearest minimum found by Vevacious (though Vevacious carries on regardless after issuing the warning). Another important sanity check that is not performed is to check that the SLHA BLOCKs required by the model file are actually present in the given SLHA input file. The user is fully responsible for providing a valid SLHA file to match the model. As model files are expected to be produced automatically by software such as SARAH, it is expected that the SLHA files for the model will also be prepared consistently with the expected BLOCKs. Unfortunately it is quite easy to miss this point when using the example model files provided by default with Vevacious: if one does use these files, one must use the correct model file for the input SLHA files, as described in Sect. 7.

*May be excessively optimistic about the region of validity*  By default, Vevacious allows VEVs to have values up to a hundred times the renormalization scale, and it is up to the user to decide whether any given set of results is meaningful and within the region of validity of the one-loop effective potential used. However, it is straightforward to change the allowed region to a smaller multiple of the renormalization scale by editing line 85 of the default Vevacious.py from

```
minuitObject.limits[ vevVariable ]
= (-100.0, 100.0)
```
to, for example,

```
minuitObject.limits[ vevVariable ]
= (-20.0, 20.0)
```
to limit VEVs to be no larger than twenty times the renormalization scale (again, the line breaks should not be present in the Python). One could also insert more complicated Python code here, but one should be aware that the PyMinuit object deals with a potential where the field values are in units of the renormalization scale.

*Not guaranteed to find minima induced purely by radiative effects*  While Vevacious does find all the extrema of the tree-level effective potential, there is no guarantee that these correspond to all the minima of the effective potential at the one-loop level. The strategy adopted by Vevacious will find all the minima of the one-loop effective potential that are in some sense “downhill” from tree-level extrema, but any minima that develop which would require “going uphill” from every tree-level extremum will not be found. Such potentials are not impossible: if the quadratic coefficient in the Coleman–Weinberg potential [38] is small enough while still positive, the single tree-level minimum can remain a minimum at the one-loop level while deeper minima induced by radiative corrections still appear. However, if the tree-level minimum is sufficiently shallow then the finite numerical derivatives used by Vevacious may be enough to push it over the small “hills” into the one-loop minima.

*Extreme slow-down with too many degrees of freedom*  Like many codes, the amount of time Vevacious needs to produce results increases worse than linearly with the number of degrees of freedom. A proper quantification of exactly how Vevacious scales with degrees of freedom remains on the to-do list, but as a guide, some typical running times (again on a 2.4 GHz core) for the HOM4PS2 part are: 3 degrees of freedom: 0.03 seconds; 5: 0.28 seconds; 7: 5.1 seconds; 10: 20 minutes; 15: 10 days. The PyMinuit part depends on the number of solutions found by HOM4PS2, but in general takes several seconds. The CosmoTransitions part is strongly dependent on the details of a particular potential, and how rapidly the path deformations converge; models with the same degrees of freedom can vary wildly from seconds to hours to be computed. For this reason, Vevacious only calls the full calculation of CosmoTransitions if the quick estimate of the upper bound on the tunneling time is over the user-given threshold, and also gives the user the option to never use the full calculation, rather only the quick upper-bound calculation, which in general takes only a few seconds at most.

*Homotopy continuation method requires discrete extrema*  The homotopy continuation method relies on tracking the paths of a discrete number of simple solutions to a discrete number of target solutions. There is no guarantee that a system with a continuous set of degenerate solutions will be solved by HOM4PS2, and unfortunately Vevacious can not check that the system has redundant degrees of freedom such as those corresponding to a gauge transformation. Hence the user must choose the degrees of freedom of the model appropriately.

*Homotopy continuation path tracking resolution*  The homotopy continuation method guarantees that there is a path from each solution of the simple system to its target solution, however, there is a danger that a finite-precision path-tracking algorithm will accidentally “jump” from the path it should be following onto a very close other path to a different solution, possibly leading to one or more solutions remaining unfound (though not necessarily, since several simple solutions may map to the same (degenerate) target solution).
Tunneling path resolution Calculating the tunneling time requires finding a continuous path in field configuration space from the false to the true vacuum. However, this must be discretized to a finite number of points on a finite machine, and may even lead to a very small barrier between the vacua disappearing entirely. However, in such cases the tunneling time should be very short indeed, so Vevacious notes this and takes a fixed very small tunneling time as the result.

5 Subtleties: renormalization schemes and allowed degrees of freedom

Currently, Vevacious performs very few sanity checks. In particular, it remains blissfully ignorant of any physical meanings the user intends for the values of the Lagrangian parameters which are given. Thus is it entirely up to the user to ensure that these values correspond correctly to the intended renormalization conditions. However, Vevacious does assume some form of dimensional regularization (e.g. switching between DR and MS depends on the value for $c_n$ given for the vector mass-squared matrix in the model file, see Appendix B, and providing explicit terms for the \(\epsilon\) scalars”). So far, Lagrangian parameters as appearing in the model files and as printed by the SPheno produced by SARAH 4 are consistent with the renormalization conditions as specified in the appendix of [35]. One should note though that the VEVs from Vevacious are given for the Landau gauge by default, and have slightly different values to those they have in the Feynman–’t Hooft gauge that is used within SPheno, for example.

In principle, every single degree of freedom should be checked for a VEV, but this is often totally impractical, given that about ten degrees of freedom is at the limit of what might be considered tolerable with current processors. Thus the user will often want to consider only a subset of scalars as being allowed non-zero VEVs. For example, when considering a supersymmetric model, one might restrict oneself to possible VEVs only for the third generation, or when considering a model specifically engineered for light staus but all other sfermions being very heavy, one might only worry about stau VEVs as a first check. Hence Vevacious should be used bearing in mind the caveat that it will not find vacua with non-zero VEVs for degrees of freedom which are not allowed non-zero VEVs in the model file. It is up to the user to decide on the best compromise between speed and comprehensiveness by choosing which degrees of freedom to use.

Importantly, the user is responsible for ensuring that the model file has a tree-level potential which has a discrete number of minima. Mostly this means that the user has to identify unphysical phases and hence remove the associated degrees of freedom from the imaginary parts of such complex fields. Another way that problems can arise is when there are flat directions such as the tan$\beta = 1$ direction of the MSSM with unbroken supersymmetry, even though the degeneracy of these directions may be lifted by loop corrections.

6 Using Vevacious

Vevalous needs at least two input files: the model file and the parameter file. The model file contains the information about the physical setup. This file is most easily generated by SARAH, as explained in Sect. 6.1. Should the user intend to modify this file by hand, we give more information about the format in Appendix B. The parameter file should be in the SLHA format, extended within the spirit of the format, as required in general for extended models, as described in Sect. 6.2. Finally, there is the option of supplying an initialization file in XML to save giving several command-line arguments, and this file is described along with these arguments in Sect. 6.3.

6.1 Preparing the input file for SARAH

SARAH [57–60] is a tool to derive many analytical properties of a particle physics model, like mass matrices, tadpole equations, vertices and renormalization group equations, from a very short user input. This information can be used, for instance, to write model files for several matrix generators or source code for SPheno [61, 62]. While previous versions of SARAH were optimized for supersymmetric models but supported also to some extent non-supersymmetric models, SARAH 4 will provide a simplified input and new features also for non-supersymmetric models [63]. In addition, SARAH 4 supports also the output of input files for Vevacious. To get this file, run in Mathematica

```mathematica
<<\!/path/to/SARAH/SARAH.m;
Start["MyModel"];
MakeVevacious[Options];
```

The possible options are:

- **ComplexParameters**, Value: list of parameters, Default: {};
  By default, all parameters are assumed to be real when writing the Vevacious input files. However, the user can define those parameters which should be treated as complex.
- **IgnoreParameters**, Value: list of parameters, Default: {};
  The user can define a list of parameters which should be set to zero when writing the Vevacious input.
1. Defining the particles which can get a VEV

2. Changing the rotation of the vector bosons:

```plaintext
DEFINITION [EWSB] [GaugeSector] =
    { [[ VB1, VB2, VB3, VB4, ZZ ]],
      [[ [ VB1, WB2, WB3 ]], [ fWm, fWp, fW0 ], ZZ ]};
```

With non-zero stau VEVs the photon won’t be massless any more but will mix with the massive gauge bosons. In general, there can be a mixing between the $B$ gauge boson ($VB$) and the three $W$ gauge bosons ($WB1 ... WB4$). The mixing matrix is called $ZZ$.

3. Changing the rotation of matter fields:

```plaintext
DEFINITION [EWSB] [MatterSector] =
    { ...,
      [[ phi, phiM, phiP, phiV, phiL, phiR ], [ fB, fW0, FHd0, FHu0, FvL, FeL, FeR ], fWm, FHdm, fWp, FHup ], [ L0, ZN ]],
      ... };;
```

The stau VEVs generate new bilinear terms in the scalar potential which trigger a mixing between the neutral and charged Higgs fields, the charged sleptons and the sneutrinos. Note that even if we had used above complex stau VEVs to present the new syntax, we don’t take the potential mixing between CP-eve n and -odd eigenstates into account here. To incorporate this, the new basis would read

```plaintext
DEFINITION [EWSB] [MatterSector] =
    { ...,
      [[ phi, phiM, phiP, phiV, phiL, phiR ], [ fB, fW0, FHd0, FHu0, FvL, FeL, FeR ], fWm, FHdm, fWp, FHup ], [ L0, ZN ]],
      ... };
```

Also, stau VEVs lead to a mixing of all the uncolored fermions, which are now also Majorana in nature. In this model file, they are now all labeled as $L0$ with mixing matrix $ZN$. The spinor sector also has to be adjusted:

```plaintext
DEFINITION [EWSB] [DiracSpinors] =
    { Fd -> [ DDL, conj[]],
      Fu -> [ FUL, conj[]],
      Chi -> [ L0, conj[]],
      Glu -> [ fG, conj[]] };
```

6.2 Preparing the SLHA data

To check for the global minimum in a given model for a specific parameter point all necessary numerical values of parameters have to be provided via an SLHA spectrum file. The conventions of this file have to be, of course, identical to the ones used for preparing the Vevacious model file.
The SLHA (1 [16] and 2 [17]) conventions are only concerned with the MSSM and the NMSSM, but do specify that extra BLOCKs within the same format should be acceptable within SLHA files, and should be ignored by programs that do not recognize them. Vevacious is intended to be used for many other models, so accepts any BLOCKs that are mentioned in its model file and looks for them in the given SLHA file. In this sense, the user is free to define BLOCKs as long as the names are unbroken strings of alphanumeric characters (e.g. BLOCK THISISVALIDNAME or BLOCK MY_BLOCK_0123). However, we strongly advise against redefining those BLOCKs specified by [16] and [17], or any of their elements, to have meanings other than those given in [16] and [17].

With this in mind, two sets of pre-generated model files for the MSSM (each file within a set allowing for different scalars to have non-zero VEVs) are provided: one set being that produced by SARAH 4, which assumes a certain extension of the SLHA BLOCK HMIX, the other restricted to quantities completely specified by the SLHA 1 and 2 conventions. The extensions of HMIX are three additional elements: 101, providing the value of \( m_3^2 \) (often written as \( B_μ \)) directly, and 102 and 103 providing the values of \( v_d \) and \( v_u \) respectively. All three can be derived from elements 2, 3, and 4 of HMIX, but are much more suited to conversion between renormalization schemes and different gauges (as \( v_d \) and \( v_u \), and thus \( \tan β \), are gauge-dependent quantities, with values which also depend on the renormalization conditions).

6.2.1 Scale dependence in the SLHA file

Vevacious is a tool for finding minima for a one-loop effective potential evaluated at a single scale, and thus it is important that the Lagrangian parameters are provided consistently at this scale. The scale is also required to be given explicitly. Since the SLHA convention specifies that running parameters are given in BLOCKs each with their own scale, at first glance this may seem problematic. Even worse, the format allows for multiple instances of the same BLOCK, each with its own scale. However, the default behaviour of SPheno, SoftSUSY, SuSuspect, and ISAJET when writing SLHA output is to give all running parameters consistently at a single scale. This is the behaviour that Vevacious assumes.

The explicit value of the scale \( Q \) used in \( V^{\text{mass}} \) in Eq. (4) is that given by the BLOCK GAUGE. All other BLOCKs are assumed to be at this same scale although it is not the default behaviour of any of the popular spectrum generators to give the same BLOCKs at different scales, if Vevacious finds multiple instances of the same BLOCK, the BLOCK with the lowest scale is used and the others ignored. In addition, Vevacious performs a consistency check that all the BLOCKs used have the same scale, aborting the calculation if not.

6.2.2 SLHA expression of parameters at different loop orders

The output BLOCKs enumerated in the SLHA papers are specified to be in the \( \overline{\text{DR}} \) renormalization scheme, but some users may prefer a different renormalization scheme. The SLHA does not insist on private BLOCKs adhering to the same standards of those explicitly part of the accord, so Vevacious allows for a certain pattern of private BLOCKs to give values for a different renormalization scheme. (Again, we strongly advise against using the BLOCKs explicitly mentioned in [16, 17] to convey values that do not adhere to the definitions in [16, 17].) The additional renormalization schemes that Vevacious allows are those where the finite parts of Lagrangian parameters are themselves apportioned into loop expansions, e.g. \( \mu + \delta \mu \), where \( \delta \mu \) is considered to be a parameter already of at least one order higher than \( \mu \).

To allow for different renormalization conditions of this type, Vevacious first looks for extra (“private”) SLHA BLOCKs that specify particular loop orders. Since Vevacious deals with one-loop effective potentials, it has two categories of parameters: “tree-level” and “one-loop”. When writing the minimization conditions for the tree-level potential, it uses exclusively the “tree-level” values. When writing the full one-loop effective potential, it uses both sets appropriately to avoid including spurious two-loop terms. With reference to Eqs. (2), (3), and (4), Vevacious writes combines the sum \( V^{\text{tree}} + V^{\text{comet}} \) by inserting the “one-loop” parameter values into \( V^{\text{tree}} \) as part of \( V^{1\text{-loop}} \). (The tree-level potential function that is also written automatically for convenience, as mentioned in Sect. 4.3, uses the “tree-level” values, of course.) The term \( V^{\text{mass}} \) is already a loop correction, so “tree-level” values are used in the \( M^2(\Phi) \) functions. If only a single value for any parameter is given, it is assumed to be in a scheme where it has a single value which is to be used in all parts of the effective potential.

As an example, within the renormalization used by SPheno3.1.12, at the point SPS1a', \( \mu \) has the value 374.9 GeV at tree level, and 394.4 GeV at one loop. Vevacious inserts the value 374.9 for \( \mu \) in the minimization conditions (as the units are assumed to be in GeV, as per the SLHA standard), and also into the “mass-squared” matrices.

The \( \overline{\text{DR}} \) scheme is just called the \( \overline{\text{DR}} \) scheme in [16, 17], however.
that are part of the evaluation of the $V^{\text{mass}}$ contributions to the one-loop effective potential. Vevacious inserts the value 394.4 for $\mu$ in the polynomial part of the potential, accounting for the contributions of both $V^{\text{tree}}$ and $V^{\text{counter}}$ together.

In detail, when inserting a “tree-level” value, Vevacious looks for an SLHA BLOCK with the prefix “TREE” first, and uses that value as its “tree-level” value. If there is no BLOCK with that prefix, or if there is such a BLOCK but it does not specify the appropriate element, then the BLOCK without the prefix is assumed to have the appropriate value. Likewise, when inserting a “one-loop” value, BLOCKS with the prefix “LOOP” have priority. Hence for the SPS1a example above, Vevacious could be given an SLHA file with 394.4 as element 1 of HMIX and 374.9 as element 1 of TREEHMIX. When Vevacious is writing the “tree-level” value of $\mu$, it would first look for element 1 of TREEHMIX, and since it would find 374.9, this value would be used, and element 1 of HMIX would not be looked for. When writing the “one-loop” value, element 1 of LOOPHMIX would be looked for, but not found, and because of this, element 1 of HMIX would then be looked for, and 394.4 would be found and used. One can specify element 1 of both TREEHMIX and LOOPHMIX, and then Vevacious would never use element 1 of HMIX, which could give the DR value, for example, without worrying that it might mix schemes in the calculation. (If one prefers to use other prefixes, both “TREE” and “LOOP” can be replaced by other strings in the model file in the <block_prefixes> element; e.g.

```xml
<block_prefixes tree="LO" loop="NLO"/>
```

so that LOHMIX and NLOHMIX would be looked for appropriately.)

Those aspects are taken into account in the SPheno [61, 62] output of SARAH 4 [63]. For this purpose, a new flag has been introduced which can be used in the SLHA input file

```xml
Block SPhenoInput  # SPheno specific input
530 1.  # Use Vevacious conventions
```

In that case, the new tree and one-loop level block will be present.

6.3 Setting up and running Vevacious

There are many options that can be passed to Vevacious. If values other than the defaults are required, they can either be passed by command-line arguments or with an initialization file in XML format. If an option is given by both command-line argument and in an initialization file, the command-line argument takes precedence.

We enumerate the options as they would be as command-line arguments setting the options to the defaults. All floating-point numbers may be given as standard decimals, such as 0.1234, or in scientific E notation, such as 1.234E-1 (uppercase 'E' or lowercase 'e', with or without preceding ‘0’ characters, with or without ‘+’ in the exponent, e.g. 987 or 9.874E+002 or 0098.7e001).

1. `--hom4ps2_dir=./HOM4PS2/`
   This is a string giving the path to the directory where the HOM4PS2 executable is.

2. `--homotopy_type=1`
   This is an integer used to decide which mode HOM4PS2 should run in: 1 is used for polyhedral homotopy and 2 for linear homotopy.

3. `--imaginary_tolerance=0.000001`
   This is a floating-point number giving the tolerance for imaginary parts of VEVs found as solutions to the tree-level minimization conditions, since it is possible that a numerical precision error could lead to what should be an exact cancellation leaving behind a small imaginary part. It is in units of GeV, as the other dimensionful values are assumed to be so since that is how they are in the SLHA standard.

4. `--model_file=./MyModel.vin`
   This is a string giving the name of the model file discussed in Sect. 6.1, including the (relative or absolute) directory path.

5. `--slha_file=./MyParameters.slha.out`
   This is a string giving the name of the SLHA file discussed in Sect. 6.2, including the (relative or absolute) directory path.

6. `--result_file=./MyResult.vout`
   This is a string giving the name of the XML output file, discussed in Sect. 6.2, to write, including the (relative or absolute) directory path.

7. `<saddle_nudges>1.0,5.0,20.0</saddle_nudges>`
   (Unfortunately this option does not work very well as a command-line argument, so instead here we display the XML element as it should appear in the XML initialization file. No initialization file has to be used, of course, if the default 1.0, 5.0, 20.0 is fine.) As discussed in Sect. 4.2, PyMinuit may get stuck at saddle points, and Vevacious creates pairs of nearby points as new starting points for PyMinuit in an attempt to get it to roll away to minima. Using the default list shown, if Vevacious finds that PyMinuit stopped at a saddle point, it will create two new starting points displaced by 1.0 GeV either side of this saddle point. If PyMinuit rolls from either or both of these displaced points to new saddle points (or just does not roll as it is still...
in a region that is too flat), Vevacious will repeat the process for each new saddle point, but this time displacing the new starting points by 5.0 GeV. Vevacious can repeat this a third time, using 20.0 GeV, but after this gives up. Giving a longer comma-separated list of floating-point numbers will lead to Vevacious performing this “nudging” as many times as there are elements of the list.

8. --max_saddle_nudges=3
   This is an integer giving the length of the list of floating-point numbers of the saddle_nudges option: if it is larger than the length of the list Vevacious already has, the list is extended with copies of the last element; if it is shorter, the list is truncated after the given number of elements.

9. --ct_path=/CosmoTransitions
   This is a string giving the path to the directory where the CosmoTransitions files pathDeformation.py and tunneling1D.py are.

10. --roll_tolerance=0.1
    This is a floating-point number giving a tolerance for extrema are identified with each other, since PyMint will roll to the same minimum from two different starting points, but not stop at exactly the same point numerically. If the length of the vector that is the difference of the two field configurations is less than the tolerance multiplied by the length of the longer of the two vectors that are the displacements of the two field configurations from the origin, then the two field configurations are taken to be the same minimum within errors; e.g. if A is \( v_d = 24.42 \), \( v_u = 245.0 \) and B is \( v_d = 24.39 \), \( v_u = 242.7 \), the length of A is 246.2140, the length of B is 243.9225, so the longer length is 246.2140; the length of their difference is 2.300196 which is less than 0.1 \( \times 246.2140 \), so A and B are considered to be the same extremum. (This is important for avoiding attempting to calculate a tunneling time from a point back to itself.)

11. --direct_time=0.1
    This is a floating-point number giving a threshold tunneling time as a fraction of the age of the Universe for whether the metastability is decided by the upper bound from the fast CosmoTransitions calculation taking a straight line from the false vacuum to the true vacuum as described in Sect. 4.2. If the upper bound resulting from this calculation is below this number, the input vacuum is considered to be short-lived and no refinement in calculating the tunneling time is pursued; e.g. if it is 0.1, and the upper bound on the tunneling time is found to be \( 10^{-20} \) times the age of the Universe, the input vacuum is judged to be short-lived since the tunneling time is definitely below 0.1 times the age of the Universe. If the value given for direct_time is negative, this fast calculation is skipped.

12. --deformed_time=0.1
    This is a floating-point number giving a threshold tunneling time as a fraction of the age of the Universe for whether the input vacuum is consider short-lived or long-lived when the full CosmoTransitions calculation is performed. If the tunneling time was calculated to have an upper bound above the threshold given by the direct_time option (or if the calculation of the upper bound was skipped because direct_time was given a negative value), then CosmoTransitions is called to calculate the bounce action allowing it to deform the path in VEV space to find the minimal bounce action. If the tunneling time calculated from this bounce action is less than the deformed_time value, the input vacuum is considered short-lived, otherwise it is reported to be long-lived. (In order to prevent overflow errors when exponentiating a potentially very large number, the bounce action is capped at 1000.) If deformed_time is set to a negative value, this calculation is skipped.

As mentioned above, an XML initialization file can be provided with values for these options. By default, Vevacious looks for ./VevaciousInitialization.xml for these options, but a different file can be specified with the command-line option --input=./example/MyVevaciousInit.xml to use ./example/MyVevaciousInit.xml as the initialization file. Any other command-line arguments take priority over options set in the initialization file.

An example XML initialization file called Vevacious-Initialization.xml is provided with the download, which shows how to set each option.

It does not matter what the root element is called (the example file provided with the download uses <Vevacious_defaults>, but it really doesn’t matter, as long as it is closed properly). Taking the option slha_file as an example, the body of the XML element with the name slha_file is used as the value of the option (stripped of leading and trailing whitespace). Hence

```xml
<slha_file>
/some/path/SPheno.spc.MSSM.SPS1ap
</slha_file>
```

would serve instead of

```bash
--slha_file
=/some/path/SPheno.spc.MSSM.SPS1ap
```

being passed as a command-line argument. Likewise,

```xml
<direct_time> 0.01 </direct_time>
```

would set the quick calculation threshold to a more conservative time of a hundredth of the age of the Universe.
6.4 The results of Vevacious

When Vevacious is finished it returns the results twice:
(i) as separate file with the name defined in the initialization file, (ii) attached to the used SLHA spectrum file. The format of the output file is again in XML. As an example of a point which is reported to be stable, we present the result of a run on the CMSSM point SPS1a [65] (which is strongly excluded by experimental non-observation, but suffices as an example). This parameter point was checked with the model file described in Sect. 6.1 for the MSSM allowing real VEVs for the neutral components of the Higgs doublets (vdR and vuR) and for the stau (vLR3 and vER3), and reads

```
<Vevacious_result>
  <reference version="1.0.7" citation="arXiv:1307.1477 (hep-ph)"/>
  <stability> stable </stability>
  <global_minimum relative_depth="-89467096.8481" vDr="24.2105220258" vER3="0.0" vLR3="0.0" vuR="241.158873762"/>
  <input_minimum relative_depth="-89467096.8481" vDr="24.2105220258" vER3="0.0" vLR3="0.0" vuR="241.158873762"/>
  <lifetime action_calculation="unnecessary">-1.0 </lifetime>
</Vevacious_result>
```

(where some line breaks have been inserted so that the output fits the width of the page). We can see that the numerical minimization did not quite roll properly to a zero VEV for $\tilde{t}_L$ at the input minimum, but stopped extremely close to it. Here, an upper limit of the lifetime has been calculated using a direct path between the input and the global minimum. The same information is also written to the SLHA file using the new block VEVACIOUSRESULTS, which has elements given by two integer indices followed by a floating-point number and a string of characters.

```
BLOCK VEVACIOUSRESULTS # results from Vevacious...
  0 0 -1.000000000E+00 short-lived
  0 1 +4.38300042E-026 direct_path_bound
  1 0 -1.09122206E+008 relative_depth
  1 1 +0.000000000E+000 vLR3
  1 2 +1.13686838E-010 vLR3
  1 3 +6.19344186E+000 vLR3
  1 4 +2.41242513E+002 vuR
  2 0 -2.11285984E+013 relative_depth
  2 1 +5.55167597E+003 vLR3
  2 2 +5.11506350E+003 vLR3
  2 3 +4.13233030E+003 vuR
  2 4 +5.24198769E+003 vuR
```

The conventions are that the (0, 0) entry of this block gives the information about the stability ($-1$ for short-lived, 0 for long-lived, 1 for stable as the floating-point number, followed by the description as the second part of the information for these indices). For metastable points, the lifetime is saved in (0, 1) (capped at 1000), with the string following giving the type of calculation (unnecessary if the input minimum was the true vacuum, direct_path_bound if the upper bound from a direct path in field configuration space was below the threshold, or full_deformed_path if CosmoTransitions had to calculate the action using path deformation). The entries (1, 1) to (1, n) contain the numerical values of all n VEVs and their names as strings at the input minimum and (2, 1) to (2, n) give the values and names of the VEVs at the global minimum. Entries (1, 0) and (2, 0) are the depths of the input and global minimum (followed by the string relative_depth).

Vevacious first deletes any files with the same name as was given as the output file, and if Vevacious was unable to end properly, no output is produced. In the case that
there were problems during the run which did not crash the program, Vevacious prints warnings, and creates an additional XML element <warning> in the results file, and also appends these warnings in a BLOCK VEVACIOUSWARNINGS. Possible warnings are (with <...> standing for strings describing field configurations, potential depths, or the string given by PyMinuit when it throws an exception)

- No tree-level extrema were found. This might happen for example if HOM4PS2 did not find any real solutions (recollecting that complex fields have already been written as pairs of real scalars) because it was given a system with continuous degenerate solutions—however, it is not necessarily the case that this is indicative of this problem, and also it is not necessarily guaranteed to result from such problematic systems.

- PyMinuit threw exceptions:

- PyMinuit got stuck at saddle points with very shallow descending or possibly flat directions:

- No one-loop extrema were found. (This shouldn’t ever happen, even for a potential that is unbounded from below, as by default, PyMinuit is restricted to a hypercube of field configurations where no field is allowed a value greater than a hundred times the scale $Q$.)

- The nearest extremum to the input field configuration is actually a saddle point:

- The nearest extremum to the input field configuration is further away than the permitted tolerance:

    PyMinuit rolled quite far from the input VEVs!
    (from <...> to <...>)

- The energy barrier between the false and true vacua is thinner than the resolution of the tunneling path:

    Energy barrier from input VEVs to global minimum thinner than resolution of tunneling path!

7 Comparison with existing tools and examples with supersymmetric models

The major components of Vevacious have been tested and used already in the literature: HOM4PS2, MINUIT, and CosmoTransitions. The innovations of Vevacious are the automatic preparation and parsing of input and output of the various components in a consistent way, optimization of the running time with respect to how short-lived a metastable vacuum might be, and the feature that SARAH 4 can automatically generate Vevacious model files for any new model that can be implemented in SARAH.

Very few tools are currently available to do the same job as Vevacious. One can of course implement the minimization conditions of a tree-level potential in HOM4PS2 or other implementations of the homotopy continuation method, or any implementation of the Gröbner basis method, on a case-by-case basis.

To our knowledge, there is only one publicly-available program that purports to find the global minimum of a potential of a quantum field theory: ScannerS [66]. However, at the time of writing, the routines to actually find the global minimum are still under development and are not available.

The model files generated by SARAH use the internal expressions that have already been cross-checked in [59, 64]. In addition, the potential was always found to have a local minimum at the input field configuration, as expected, for a wide range of test points. Furthermore, using the model file

SARAH-SPhenoNMSSM_JustNormalHiggsAndSingletVevs.vin described below and with the modification to Vevacious to use only a tree-level analysis as described in Sect. 4.3, we confirmed the results of [20].

Several example model and parameter files are provided with the download of Vevacious. Three model files are given for the MSSM with different allowed non-zero VEVs:

- SARAH-SPhenoNMSSM_JustNormalHiggsVevs.vin with only the normal real neutral Higgs VEVs allowed;
These model files were generated automatically with SARAH 4. They assume that the SLHA parameter file will use the standard SARAH-generated SPhenoMSSM (SPheno using the MSSM SARAH model file) output, which uses the SLHA 2 flavor violation conventions, i.e. that the blocks TE, TD, TU, MSQ2, etc., are present, and also the extra HMIX parameters 101, 102, and 103, that SPhenoMSSM prints out. There are no tadpoles for the first two generations of sfermions, which is strictly inconsistent with non-zero VEVs for the third generation along with the Higgs doublets in the presence of non-zero off-diagonal Yukawa and trilinear soft SUSY-breaking terms. However, the assumption is that any point with e.g. non-zero stop VEVs has such small sup and scharm VEVs that the stability of the input vacuum can be judged by comparison to the minimum in the zero-sup-and-scharm-VEV plane nearest the true global minimum.

Three variants which use only SLHA 2-specified BLOCKs are also present:

\begin{itemize}
  \item pure_SLHA2_MSSM_JustNormalHiggsVevs.vin, \item pure_SLHA2_MSSM_RealHiggsAndStauVevs.vin, \item pure_SLHA2_MSSM_RealHiggsAndStauStopVevs.vin.
\end{itemize}

These model files also assume that the SLHA parameter file will use the SLHA 2 flavor violation conventions, i.e. that the blocks TE, TD, TU, MSQ2, etc., are present, but do not require the extra HMIX parameters of the SARAH–SPheno versions.

Three variants which use only SLHA 1-specified BLOCKs are also present:

\begin{itemize}
  \item pure_SLHA1_MSSM_JustNormalHiggsVevs.vin, \item pure_SLHA1_MSSM_RealHiggsAndStauVevs.vin, \item pure_SLHA1_MSSM_RealHiggsAndStauStopVevs.vin.
\end{itemize}

These model files assume that the SLHA parameter file will use the SLHA 1 conventions without flavor violation, i.e. that the blocks AE, AD, AU, etc., are present, that the sfermion soft SUSY-breaking sfermion mass-squared parameters are given by the MSQ2 BLOCK instead of in MSQ2 etc., and also do not require the extra HMIX parameters of the SARAH–SPheno versions.

Three model files are also provided for a constrained version of the NMSSM (where the dimensionful superpotential parameters $\mu$, $\mu'$, and $\xi_F$, and the soft SUSY-breaking parameters $m^2_1$, $m^2_2$, and $\xi_5$, in the notation of [17], are set to zero) with different allowed non-zero VEVs:

\begin{itemize}
  \item pure_SLHA2_NMSSM_JustNormalHiggsAndSingletVevs.vin, \item pure_SLHA2_NMSSM_RealHiggsAndSingletVevs.vin, \item pure_SLHA2_NMSSM_RealHiggsAndSingletAndStauVevs.vin, \item pure_SLHA2_NMSSM_RealHiggsAndSingletAndStauAndStopVevs.vin.
\end{itemize}

(There are no SLHA 1 variants, as the NMSSM was not specified in the SLHA 1 conventions. It is important to note though that these SLHA 2 model files still also require the flavor violation conventions, i.e. that TE, TD, TU, MSQ2, etc., are present, even though an NMSSM SLHA 2 file without flavor violation, using AE instead of TE etc., is still a valid SLHA 2 file.)

In addition, several example parameter points have been provided to demonstrate the existence of metastable points:

\begin{itemize}
  \item SPS1a, the CMSSM point SPS1a from [65] (which is stable); \item CMSSM_CCB, which corresponds to the CMSSM best-fit point including LHC and \( m_h = 126 \) GeV constraints from [67] (which has a global charge- and color-breaking minimum); \item NUHM1_CCB, which corresponds to the NUHM1 best-fit point ("low") from [68] (which also has a global charge- and color-breaking minimum); \item CNMSSM_wrong_neutral, which corresponds to benchmark point P1 from [69] (which also has a neutral global minimum which however is not the input minimum).
\end{itemize}
These parameter files are given in the SLHA output of SPheno as SPS1a.slha.out and so on, with example output as SPS1a.vout and so on. (The SLHA input files are SPS1a.slha.in and so on.)

We note that the CCB vacua found for CMSSM_CCB, for example, have VEVs of the order of five times the renormalization scale (shown in Sect. 6.4). This should not cause much concern, as \( \ln(5^2)/(4\pi) \approx 0.256 \) so the one-loop effective potential should still be reasonable around this minimum. However, even if one restricts the VEVs to be less than twice the scale, as described in Sect. 4.4, Vevacious still finds a tunneling time (upper bound) of \( 10^{-6} \) times the age of the known Universe to a CCB configuration at the edge of the bounding hypercube.

The MSSM model and parameter files are in the MSSM subdirectory and those of the NMSSM are in the NMSSM subdirectory.

8 Conclusion

Several extensions of the Standard Model contain additional scalar states. Usually one engineers the model such that one obtains a phenomenologically acceptable vacuum with the desired breaking of \( SU(2)_L \times U(1)_Y \) to \( U(1)_{EM} \). However, in general it is not checked if the minimum obtained is the global minimum, as this in general quite an involved task already at tree level, where hardly any analytical conditions can be given. Often loop corrections become important too.

To tackle this problem, we have presented the program Vevacious as a tool to quickly evaluate one-loop effective potentials of a given model. It finds all extrema at tree level, which allows for a first check for undesired minima. Starting from these extrema, it calculates the one-loop effective potential to obtain a more reliable result. This is important as loop contributions can potentially change the nature of an extremum. In the case that the original minimum turns out to be merely a local minimum rather than the global minimum, the possibility is given to evaluate the tunneling time. As test cases we have considered supersymmetric models, but the program can be used for a general model provided that the tree-level potential is polynomial in the scalar fields.

Acknowledgements The authors would like to thank Max Wainwright for advice on CosmoTransitions. This work has been supported by the DFG project No. PO-1337/2-1 and the research training group GRK 1147.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

Appendix A: Installation and pre-requisites

To fully evaluate a parameter point with Vevacious and the connected tools, you need a Linux or MacOS system with

- … a C++ compiler such as gcc
- … the Python environment, at least version 2.7.1
- … the python-dev headers (Python.h) for gcc or the equivalent—this is important! (MacPorts for OSX automatically installs Python.h as part of the installation of Python 2.7)

If you want to create new input files using SARAH, you need at least Mathematica 7. To compile SPheno to create the numerical input for Vevacious, a Fortran compiler such as gfortran or ifort is needed.

Vevacious makes use of several public tools. We give here only a brief introduction to the installation of these tools but refer to the corresponding references and authors for more information.

We assume in this description that all codes are downloaded and extracted in the same directory. The placeholder for the path to this directory is called $VPATH in the following.

1. HOM4PS2

   (a) Download HOM4PS2 from [http://www.math.nsysu.edu.tw/~leetsung/works/HOM4PS_soft_files/HOM4PS_Linux.htm](http://www.math.nsysu.edu.tw/~leetsung/works/HOM4PS_soft_files/HOM4PS_Linux.htm) (without the line break)

   (b) Extract the tar file

       > tar -xf HOM4PS2_64-bit.tar.gz

       which should create $VPATH/HOM4PS2/.

   (c) No compilation is necessary but now would be a good point to check that HOM4PS2 works with one of its examples.

2. MINUIT and PyMinuit

   Detailed instructions about the installation and compilation of PyMinuit and MINUIT are given at [https://code.google.com/p/pyminuit/wiki/HowToInstall](https://code.google.com/p/pyminuit/wiki/HowToInstall) (without the line break) We summarize only briefly the main steps

   (a) Download MINUIT from [http://code.google.com/p/pyminuit/](http://code.google.com/p/pyminuit/). Choose Minuit-1_7_9-patch1.tar.gz.

   (b) Extract the tarball

       > tar -xf Minuit-1_7_9-patch1.tar.gz

       which should create $VPATH/Minuit-1_7_9/.

   (c) Configure and compile MINUIT

       > cd Minuit-1_7_9

       > ./configure --prefix = $VPATH/Min/

       > make

       > make install

       which should create $VPATH/Min/.
3. Make sure that you have properly exported PYTHONPATH (which should include the path to minuit.so, in $VPATH/pym/lib/python/ in our example) and LD_LIBRARY_PATH (which should include the path to liblcg_Minuit.a, in $VPATH/Min/lib/ in our example). Now would be a good time to check that PyMinuit works by running the following test program:

```python
>>> import minuit
>>> def (x, y):
...    return (x**2 + y**2)**2
...    -0.5 * x**2
...    
...   
>>> m = minuit.Minuit(f)
>>> m.values = {x: 0.1, y: 0.2}
>>> m.migrad()
>>> m.values, m.fval
```

4. CosmoTransitions

(a) Ensure that the Python packages Numpy and SciPy are installed (if not, please use your favorite Internet search engine to find out how to install them on your system)

(b) Download CosmoTransitions_package_v1.0.2.zip or a more recent version from http://chasm.uchicago.edu/cosmotransitions

(c) unzip the archive

```bash
> unzip CosmoTransitions_package_v1.0.2.zip
```

which should create $VPATH/CosmoTransitions_package_v1.0.2/ (with the command broken over two lines to fit on the page)

5. LHPC

(a) Download LHPC from http://www.hepforge.org/downloads/lhpc/

(b) Extract the files from the compressed tarball (change the version number appropriately)

```bash
> tar -xf LHPC-0.8.5.tar.gz
```

(c) Enter the LHPC directory and compile it

```bash
> cd LHPC-0.8.5
> make
```

6. Vevacious

(a) Download the most recent version from http://www.hepforge.org/downloads/vevacious/

(b) Extract the files from the compressed tarball and compile Vevacious stating the path to LHPC

```bash
> tar -xf Vevacious-1.0.7.tar.gz
> cd Vevacious-1.0.7
> make LHPCDIR=$VPATH/LHPC-0.8.5/
```

Now would be a good time to check that Vevacious works. Edit $VPATH/Vevacious-1.0.7/bin/VevaciousInitialization.xml to correctly set all the paths, then run

```bash
> ./bin/Vevacious.exe
--input=./bin/VevaciousInitialization.xml
```

(with the command broken over two lines to fit on the page) which should produce the result file with the name given in the initialization file.

For example, in Debian-based Linux distribution one can ensure that the paths are properly set by adding the following to .bashrc:

```bash
export PYTHONPATH =
$VPATH/pym/lib/python/:$PYTHONPATH
export PYTHONPATH =
$VPATH/CosmoTransitions_package_v1.0.2:/
cosmoTransitions/:$PYTHONPATH
export LD_LIBRARY_PATH =
$VPATH/Min/lib/:$LD_LIBRARY_PATH
```

(without the line breaks inserted after the ‘=’ characters or in the middle of the CosmoTransitions path, of course). To get the input for a point, you can use a spectrum generator based on SPheno based on the corresponding SARAH output
1. SARAH
   (a) Download the most recent version of SARAH 4 from http://sarah.hepforge.org
   (b) Extract SARAH and run Mathematica
      > tar -xf SARAH4b-0.0.5.tar.gz
      > mathematica
   (c) Load SARAH, evaluate a model and generate the SPheno output
      << "$VPATH/SARAH4b-0.0.5/SARAH.m";
      Start["MyModel"]; MakeSPheno[];
      MyModel can be for instance MSSM or NMSSM.

2. SPheno
   (a) Download SPheno from http://spheno.hepforge.org
   (b) Extract SPheno
      > tar -xf SPheno-3.2.3.tar.gz
   (c) Enter the SPheno directory and create a new subdirectory for the new model
      > cd SPheno-3.2.3
      > mkdir MyModel
   (d) Copy the source code produced by SARAH to the new subdirectory
      > cp $VPATH/SARAH4b-0.0.5/Output/
         MyModel/EWSB/SPheno/* MyModel/
      (without the line break)
   (e) Compile SPheno and the new SPheno module
      > make Model=MyModel

Appendix B: Model file format

Vevacious is not restricted to a specific model, but any necessary information to evaluate the one-loop effective potential in a given model, assuming a particular set of VEVs, must be provided by the user as a model file. This file can usually be produced by SARAH as described in Sect. 6.1. However, in the case that the user wants to modify things manually, we give here more information about the format.

The Lagrangian parameters that will take values given by the SLHA file are written in the form SLHA::BLOCKNAME [ENTRY], where ENTRY is a comma-separated list of indices. For example, element 1 of the HMIX BLOCK is written as SLHA::HMIX[1] and the 2,3 element of the YD BLOCK is written as SLHA::YD[2,3] (floating point numbers are still interpreted as integers, so SLHA::YD[2.0,3.0] would also work). Any BLOCKs with no index should use an empty list, e.g. SLHA::ALPHA[].

This file is in XML and begins/ends with

```xml
<Vevacious_stuff>
...
</Vevacious_stuff>
```

(though the actual name of the root element is not important: one could use

```xml
<MyModelForVevacious>
...
</MyModelForVevacious>
```

for example instead). The model file must contain the following information:

1. The names of all VEVs which can be present

```xml
<input_vevs vdR="SLHA::HMIX[102]"
   vuR= "SLHA::HMIX[103]"
   vLR3 = "0" vER3 = "0" >
   <taken_positive> vdR, vER3 </taken_positive>
</input_vevs>
```

(though the line breaks should not be present in the <input_vevs> opening tag). The XML element <input_vevs> serves the dual purpose of enumerating the allowed non-zero VEVs along with specifying what is considered to be the input minimum. In the MSSM example here, the Higgs fields of course are allowed VEVs, and the input minimum is specified by vdR (vd) being provided in the SLHA file in the BLOCKHMIX as entry 102, and vuR (vu) by entry 103. The model file is also allowing non-zero stau VEVs (vLR3, vER3), and also at the same time specifying that their value at the input minimum is 0. (One can actually put valid Python code here within the quote marks for the input minimum values, though it is not recommended, and surrounding the code with brackets is advised if one insists on going through with it.)

In addition, the is usually some redundancy in the sets of VEVs which minimize the tree-level scalar potential, since different minima can be related by phase rotations. To reduce the number of redundant solutions, it can be explicitly defined that some VEVs have to be positive using <taken_positive> ...
</taken_positive>.

The Lagrangian parameters that will take values given by the SLHA file are written in the form SLHA::BLOCKNAME [ENTRY], where ENTRY is a comma-separated list of indices. For example, element 1 of the HMIX BLOCK is written as SLHA::HMIX[1] and the 2,3 element of the YD BLOCK is written as SLHA::YD[2,3] (floating point numbers are still interpreted as integers, so SLHA::YD[2.0,3.0] would also work). Any BLOCKs with no index should use an empty list, e.g. SLHA::ALPHA[].

This file is in XML and begins/ends with

```xml
<Vevacious_stuff>
...
</Vevacious_stuff>
```

(though the actual name of the root element is not important: one could use

```xml
<MyModelForVevacious>
...
</MyModelForVevacious>
```

for example instead). The model file must contain the following information:

1. **The names of all VEVs which can be present**

```xml
<input_vevs vdR="SLHA::HMIX[102]"
   vuR= "SLHA::HMIX[103]"
   vLR3 = "0" vER3 = "0" >
   <taken_positive> vdR, vER3 </taken_positive>
</input_vevs>
```

(through the line breaks should not be present in the <input_vevs> opening tag). The XML element <input_vevs> serves the dual purpose of enumerating the allowed non-zero VEVs along with specifying what is considered to be the input minimum. In the MSSM example here, the Higgs fields of course are allowed VEVs, and the input minimum is specified by vdR (vd) being provided in the SLHA file in the BLOCKHMIX as entry 102, and vuR (vu) by entry 103. The model file is also allowing non-zero stau VEVs (vLR3, vER3), and also at the same time specifying that their value at the input minimum is 0. (One can actually put valid Python code here within the quote marks for the input minimum values, though it is not recommended, and surrounding the code with brackets is advised if one insists on going through with it.)

In addition, the is usually some redundancy in the sets of VEVs which minimize the tree-level scalar potential, since different minima can be related by phase rotations. To reduce the number of redundant solutions, it can be explicitly defined that some VEVs have to be positive using <taken_positive> ...
</taken_positive>. 

---

 Springer
2. The tree-level tadpole equations

\[
\begin{align*}
\text{<tadpoles>} \\
\{ \\
(0.5+vlR3+SLHA::MSL2[1,3]) \\
+(0.5+vlR3+SLHA::MSL2[3,1]) \\
+(0.7071067811865475+vdR+ver3+SLHA::TE[3,1]) \\
+(0.5+vdR^2+vlR3+SLHA::YE[1,1]+SLHA::YE[1,3]) \\
+(0.5+vdR^2+vlR3+SLHA::YE[2,1]+SLHA::YE[2,3]) \\
+(0.5+vdR^2+vlR3+SLHA::YE[3,1]+SLHA::YE[3,3]) \\
+(0.5+vlR3^2+vlR3+SLHA::YE[1,3]+SLHA::YE[3,1]) \\
; \\
\} \text{<tadpoles>}
\end{align*}
\]

(digits from the second decimal expansion of $\sqrt{2}$ have been truncated just to fit on the page). This element contains a list of entries in the format $(t_1; t_2; t_3; \ldots; t_n)$. Here, $t_i$ are the tadpole equations of the tree-level scalar potential, i.e. $t_i = \frac{\partial V}{\partial \phi_i} = 0$. Note that each equation has to end with a semi-colon. In addition, to circumvent problems during parsing this file, it is convenient to use for each term a separate line and to put it into brackets. Furthermore, to associate the different parameters with the numerical values given later on via an SLHA spectrum file, all parameters but the field configurations have to be replaced by their corresponding entries in the SLHA file, in the SLHA::BLOCKNAME [ENTRY] format explained above. For instance, using the SLHA 2 conventions [17], the hypercharge $Y_1$ is replaced by SLHA::GAUGE [1] and the top Yukawa coupling $Y_t = Y_u^{33}$ by SLHA::YU [3,3].

3. The polynomial part of the scalar potential

\[
\begin{align*}
\text{<polynomial_part>} \\
(0.03125+vdR^4+SLHA::GAUGE[1]^2) \\
+ (-0.125+vdR^2+ve3^2+SLHA::GAUGE[1]^2) \\
+ (0.125+ve3^4+SLHA::GAUGE[1]^2) \\
+ (0.0625+vdR^2+vlR3^2+SLHA::GAUGE[1]^2) \\
+ (-0.125+ve3^2+vlR3^2+SLHA::GAUGE[1]^2) \\
; \\
\end{align*}
\]

This element contains the scalar potential $V(\phi_1; g_1, Y_1, T_1, \ldots, m_{1}^2)$ as function of all possible field configurations and the other parameters like gauge and Yukawa couplings or mass terms. The conventions are similar to those of <tadpoles>: (i) choose a separate line for each term, (ii) put each term into brackets, (iii) replace all parameters but the field configurations by their SLHA entries.

4. All mass-squared matrices to calculate the full one-loop effective potential

To calculate the one-loop effective potential all field-configuration-dependent “masses” $M^2(\Phi)$ have to be specified. This happens by using for each mass-squared matrix the XML element <mass-squared_matrix> ...

References

1. J. Elias-Miro, J.R. Espinosa, G.F. Giudice, G. Isidori, A. Riotto et al., Phys. Lett. B 709, 222 (2012). doi:10.1016/j.physletb.2012.02.013
68. O. Buchmueller, R. Cavanaugh, M. Citron, A. De Roeck, M. Dolan et al., Eur. Phys. J. C 72, 2243 (2012). doi:10.1140/epjc/s10052-012-2243-3

69. A. Djouadi, M. Drees, U. Ellwanger, R. Godbole, C. Hugo- nie et al., J. High Energy Phys. 0807, 002 (2008). doi:10.1088/1126-6708/2008/07/002