ARGES – Advanced Renormalisation Group Equation Simplifier

Article (Accepted Version)

ARGES – Advanced Renormalisation Group Equation Simplifier

Daniel F. Litim[a], Tom Steudtner[b,a]

[a] *Department of Physics and Astronomy, University of Sussex, Brighton, BN1 9QH, U.K*
[b] *Fakultät Physik, TU Dortmund, Otto-Hahn-Str.4, D-44221 Dortmund, Germany*

Abstract

We present the initial release of `ARGES`, a toolkit for obtaining renormalisation group equations in perturbation theory. As such, `ARGES` can handle any perturbatively renormalisable four-dimensional quantum field theory. Notable further features include a symbolic rather than numeric computation, input of unconventional scalar and Yukawa sectors, an interactive evaluation and disentanglement as well as capabilities to inject algebraic simplification rules. We provide a conceptual and practical introduction into `ARGES`, and highlight similarities and differences with complementary packages.

Keywords: Renormalisation group equations, quantum field theory

PROGRAM SUMMARY

Program Title: ARGES
Licensing provisions: GPLv3
Programming language: Wolfram Language (Mathematica)
Nature of problem: Computation of renormalisation group equations of perturbatively renormalisable quantum field theories in four space-time dimensions to various orders in the loop expansion.
Solution method: Automatize computation of RGEs from input of field content, representations, couplings and index contractions.
Unusual features: Allows for variable generations, symmetry representations, and gauge groups. Advanced algebraic structures can be defined via simplification rules.

1. Overview

In ancient greek mythology, Arges is the name of a cyclops who, after being exiled by the Titans was eventually freed by Zeus to forge lightning bolts for his battle against their former masters. In this tradition of being a helping hand

Email addresses: `D.Litim@sussex.ac.uk` (Daniel F. Litim), `tom2.steudtner@tu-dortmund.de` (Tom Steudtner)

in times of struggle, we introduce the computational tool `ARGES` — Advanced Renormalisation Group Equation Simplifier.
As the name suggests, `ARGES` is a framework to obtain perturbative renormalisation group equations for renormalisable, four-dimensional quantum field theories. The task is achieved algebraically, by extracting expressions from known formal results in the $\overline{\text{MS}}$ renormalisation scheme [1–16].
Written in the *Wolfram Language*, `ARGES` has been tested for *Mathematica* versions 8.0 – 12.0 [17] and is available via [18] under *GNU General Public License* [19]. In this work, we explain and provide examples for `ARGES` key features and capabilities, as well as highlight similarities and differences with alternative software packages `SARAH` 4 [20–24] and `PyR@TE` 3 [25–27].

2. Statement of the problem & conventions

The computation of renormalisation group equations (RGEs) of running couplings continues to be of key importance for many applications of quantum field theory (QFT) in particle physics and critical phenomena. Thankfully, for weakly coupled and perturbatively renormalisable QFTs, the procedure can be partially automatised. This is achieved by considering a template action in which any concrete model can be embedded, which allows the computation to be broken down into two parts. In a first step, momentum integrals and spinor traces are resolved, and all RGEs expressed in terms of generalised couplings. Universal expressions of this type have been made available in the literature [1–16], and can be mapped back onto any renormalisable QFT in a second step.
In practice, however, the embeddings of couplings and fields are in general non-trivial. In fact, for increasingly complex QFTs involving a large number of quantum fields and interaction terms, the second step becomes rapidly more involved than the first one. Moreover, with increasing loop order, the complexity increases even further. For these reasons, computer codes such as `ARGES` are required, which automatise the embedding and its resolution to enable universal access to the results in the literature and the extraction of RGEs for concrete models.
With this aim in mind, `ARGES` adopts the conventions developed in [1–3] and expresses the template action for a perturbatively renormalisable four-dimensional QFT as

$$\begin{aligned}\mathcal{L} =& -\frac{1}{4}F_I^{\mu\nu}F^I_{\mu\nu} + \frac{1}{2}D^\mu\phi_a D_\mu\phi_a + i\psi_j^\dagger\sigma^\mu D_\mu\psi_j + \mathcal{L}_{\text{gh}} + \mathcal{L}_{\text{gf}} \\ & - \frac{1}{2}\left(Y^a_{jk}\,\psi_j\varepsilon\psi_k\phi_a + Y^{a*}_{jk}\,\psi_j^\dagger\varepsilon\psi_k^\dagger\phi_a\right) - \frac{1}{4!}\lambda_{abcd}\,\phi_a\phi_b\phi_c\phi_d \\ & - \frac{1}{2}\left[m_{jk}\,\psi_j\varepsilon\psi_k + m^*_{jk}\,\psi_j^\dagger\varepsilon\psi_k^\dagger\right] - \frac{\mu^2_{ab}}{2!}\phi_a\phi_b - \frac{h_{abc}}{3!}\phi_a\phi_b\phi_c\,.\end{aligned} \tag{1}$$

Here $\mathcal{L}_{\text{gf, gh}}$ are gauge fixing and ghost terms and ψ_i, ϕ_a being Weyl fermions and real scalar fields. Their respective universal indices i, j, $k\cdots$ and a, b, $c\cdots$ run over all particle species (including real and imaginary parts of complex

scalars), generations, flavours, and gauge components.[1] In the same spirit, a sum over all gauge groups is implied. Gauge interactions are given by

$$\begin{aligned} D_\mu \phi_a &= \partial_\mu \phi_a + i A^I_\mu \, t^I_{ab} \, \phi_b, \\ D_\mu \psi_i &= \partial_\mu \psi_i + i A^I_\mu \, t^I_{ij} \, \psi_j, \\ F^I_{\mu\nu} &= \partial_\mu A^I_\nu - \partial_\nu A^I_\mu - t^I_{JK} \, A^J_\mu A^K_\nu , \end{aligned} \tag{2}$$

where t^I are generators of the respective representations, and gauge couplings g_α are implied via $A^I_\mu = g_\alpha A^{I_\alpha}_\mu$, with α counting over all gauge subgroups and indices $I, J, K \cdots$ denote the adjoint indices of gauge fields. Finally μ, $\nu \cdots$ are the usual Lorentz indices and $\varepsilon = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$ denotes the two-dimensional Levi-Civita symbol which contracts the implicit spinor components of fermions ψ_i. Overall, any allowed interaction monomial $\sim \alpha_{\ell_1 \cdots \ell_n} X_{\ell_1} \cdots X_{\ell_n}$ involving scalar, fermion and vector fields $X_\ell \in \{\phi_a, \psi_i, A^I_\mu\}$ as well as couplings $\alpha_{\ell_1 \cdots \ell_n} \in \{g_\alpha, Y^a_{jk}, \lambda_{abcd}, m_{jk}, \mu^2_{ab}, h_{abc}\}$ (with the appropriate set of indices ℓ_i) can be accommodated. After renormalisation, these bare couplings are replaced by the respective renormalised ones $\alpha_{\ell_1 \cdots \ell_n}(Q)$ and field strength renormalisation constants $\sqrt{Z}_{\ell_1 \ell_2}$ are introduced

$$X_{\ell_1} \mapsto \left(\sqrt{Z}_{\ell_1 \ell_2}(Q) + \delta \sqrt{Z}_{\ell_1 \ell_2} \right) X_{\ell_2}, \quad \alpha_{\ell_1 \cdots \ell_n} \mapsto \alpha_{\ell_1 \cdots \ell_n}(Q) + \delta \alpha_{\ell_1 \cdots \ell_n}. \tag{3}$$

Here, $\delta\sqrt{Z}$ and $\delta\alpha$ are counter-terms, and Q denotes the renormalisation scale parameter. The renormalisation scale running of $\sqrt{Z}(Q)$ and $\alpha(Q)$ are then encoded in the respective field anomalous dimensions and β-functions

$$\gamma_{\ell_1 \ell_2}(\alpha) = \left(\frac{d \ln \sqrt{Z}(Q)}{d \ln Q} \right)_{\ell_1 \ell_2}, \qquad \beta_{\ell_1 \cdots \ell_n}(\alpha) = \left(\frac{d \, \alpha(Q)}{d \ln Q} \right)_{\ell_1 \cdots \ell_n}. \tag{4}$$

Presently, `ARGES` implements the literature expressions for (4) up to three-loop order in gauge coupling β-functions [1, 4, 5, 8, 11, 12, 14]. RG equations for Yukawa interactions [2, 4, 13], scalar quartics [3, 4, 13, 15] and cubics [3, 4, 13, 15], fermion and scalar masses [2–4, 13, 15], vacuum expectation values [9, 10] as well as scalar and fermion anomalous dimensions [2–4, 15] are implemented up to two-loop order. Finally, purely scalar contributions to scalar quartic and cubic interactions, masses, and field anomalous dimensions are included up to four-loop order [6, 7, 15]. All of the above expressions have originally been obtained using dimensional regularisation [29, 30] and the $\overline{\text{MS}}$ renormalisation scheme [31, 32].

A central capability of `ARGES` is to extract RGEs for the unembedded couplings $\alpha^{(i)}$. In general, these can be *disentangled* from the linear combination of generalised β-functions (4) by determining the coefficients $\eta^{(i)}_{\ell_1 \cdots \ell_n}$ of the couplings

$$\alpha^{(i)} = \sum_{\ell_1 \cdots \ell_n} \eta^{(i)}_{\ell_1 \cdots \ell_n} \alpha_{\ell_1 \cdots \ell_n} \tag{5}$$

[1] An interesting and slightly modified basis has recently been put forward in [14, 28]. However, the development of `ARGES` predates these works.

at tree level and then employing them for the loop-level RGEs

$$\beta^{(i)}(\alpha^{(j)}) \equiv \frac{d\,\alpha^{(i)}}{d\ln Q} = \sum_{\ell_1\cdots\ell_n} \eta^{(i)}_{\ell_1\cdots\ell_n}\beta_{\ell_1\cdots\ell_n}\,. \tag{6}$$

In the following, implementation details will be highlighted in comparison with pre-existing software.

3. Design & comparison with other frameworks

The design goals of `ARGES` are complementary to those of the two alternative software packages available with the same scope, namely `SARAH` and `PyR@TE`. Broadly speaking, `SARAH` and `PyR@TE` are quite different in their implementation: `PyR@TE` is a highly specialised *python* package accepting a single input file and command line parameters. It allows for theories ranging from simplest toy models to complicated SM extensions. `SARAH` on the other hand is a large *Mathematica* package and in fact a wider framework with many more capabilities. Geared towards realistic theories, it requires complex inputs, and is the back end of various farther reaching scientific software. On the other hand, the inner workings and capabilities of `PyR@TE` and `SARAH` are conceptually similar. It is here where `ARGES` takes a different stance. The key differences in `ARGES` functionality will be highlighted in the following.

Index contractions are user input. With the exception of interaction terms involving gauge bosons, contractions of gauge and global indices over each interaction vertex have to be specified as input. `ARGES` knows only very little about the Lie algebras, and shifts the responsibility of formulating a gauge invariant action, consistent with the desired symmetries, to the user. The advantage is that the user has full control over the shape of the action, eliminating any uncertainty e.g. about coupling normalisations. While `SARAH` and `PyR@TE` allow for an explicit input of contractions, they also link to the packages `Susyno` [33] and its *python* clone `PyLie` [27, 34] that automatise the search for index contractions. However, this procedure provides a much less fine-grained control, especially in cases where several contractions are possible. Even though the automatisation avoids explicit violations of gauge invariance, all three codes may give inconsistent results if not all interactions allowed by the symmetries are manually included from the outset.

Groups, representation and multiplicities can be variables. As opposed to `PyR@TE` and `SARAH`, `ARGES` allows for gauge groups to be either completely undetermined, formulating the results in terms of general gauge invariants, or to be an entire family of gauge groups such as $SU(N)$ without specifying N. More so, the representations for each field can be kept a variable as well, and so does the number of generations or matter field multiplicities. This ties in with the previous point: `ARGES` is agnostic to symmetries and algebras, and treats everything as an index. This allows for a systematic study of a large range of models and represents a key advantage over existing codes.

Gauge invariants are not resolved by default. ARGES does not resolve gauge invariants such as Dynkin indices and quadratic Casimirs by default. In general, they are also considered user input. For simple representations of $U(1)$, $SU(N)$, $SO(N)$ and $Sp(2N)$ gauge groups, invariants can be computed automatically and are available as substitution rules.

Scalars can carry flavour. In SARAH and PyR@TE, only fermions carry a single flavour index, while scalars do not. In ARGES, scalars carry two such indices instead, which can be contracted freely, and whose ranges can be kept as open parameters. This is sufficient to embed any scalar field allowed by (1), including with parametrical global multiplicities.

Flexible flavour structure in Yukawa sector. While the design of SARAH and PyR@TE constrains Yukawa couplings to be of matrix form $y_{ij}\,\overline{\psi}_i\,\phi\,\psi'_j$+h.c. in the fermion generations, ARGES also allows for a generalised contraction h of flavour indices and a single Yukawa coupling y in a vertex $y\,h_{ijkl}\,\overline{\psi}_i\,\phi_{kl}\,\psi'_j$ + h.c. This ensures that ARGES' capabilities completely cover the set of actions (1), including *e.g.* BSM models with extended flavour structures such as in GUTs, or in settings where the multiplicities of gauge or matter fields remain free parameters throughout. The latter features are absent in SARAH and PyR@TE.[2]

Disentanglement of RGEs by the user. In PyR@TE and SARAH, the disentanglement of couplings and RGEs as described in (5) and (6) is fully automatised. However, such algorithms can be computationally expensive for even unsuccessful. This may occur in valid models, but is bound to happen if the specified couplings cannot be disentangled.[3] In ARGES, the disentangling of couplings and RGEs is out-sourced to the user (examples are provided in Sec. 5). Besides enhancing the stability of the code, this strategy bears two further advantages: for one, there are often several choices of external indices, and the user is free to choose the one minimising the computational efforts for the RGEs, e.g. with the most symmetries of external legs. Secondly, the setup allows the user to input external indices as variables, and reconcile tree- with loop-level contractions. This also offers a simple way to check whether couplings allowed by the symmetry but omitted in the action are nevertheless switched on by quantum fluctuations under the renormalisation group.

Efficient handling of unspecified interactions. ARGES allows index contraction to be merely defined by a number of algebraic relations, providing a mechanism to insert such information and allow for an efficient computation of RGEs (an example for this is provided in Sec. 5).

Irreducible representations, kinetic mixing and gauge fixing. At the

[2]Finite tensorial index structures can be flattened-out by expanding flavour contractions at each vertex, and by treating flavour components of couplings and fields as independent vertices and particles to make this accessible for SARAH. In PyR@TE, this can be done if fermion generations are integers and not open parameters.

[3]The latest version of PyR@TE offers a very reliable and well-performing algorithm with couplings being flagged if they cannot be disentangled.

time of writing, the capabilities of ARGES are limited to irreducible representations without kinetic mixing of $U(1)$ gauge groups, which is implemented in the most recent versions of SARAH and PyR@TE. All three packages assume an R_ξ gauge fixing.

4. Installation

ARGES is designed to be easily distributable, the code has no external dependencies and is located in a single file ARGES.m. Moreover, another design goal is that ARGES does neither require nor encourage a notebook, or any graphical user interface in general, access to a *Mathematica* kernel is sufficient. The source code can be acquired from [18], e.g. by cloning the git repository.

```
git clone https://github.com/TomSteu/ARGES
```

The relevant file can be loaded by the kernel directly via

```
Get["~/path/to/ARGES.m", ARGES`];
```

or alternatively, moved into a location contained in $Path manually, using Install[__] or the graphical user interface and included via:

```
<<ARGES`
```

If no output is produced by whatever the method of choice, then the installation was successful. Next, we will proceed with an example on how to define a valid input model.

5. ARGES by example

In this section we demonstrate the basic and advanced functionality of ARGES by application, in good faith that generalisations to other models are then largely straightforward.

5.1. Defining a model

The input required by ARGES can either be provided by a model file or by an interactive session. Here, we proceed with the model

$$\begin{aligned}\mathcal{L} = &-\tfrac{1}{4}F_A^{\mu\nu}F^A_{\mu\nu} + \overline{Q}\, i\not{D}\, Q + \operatorname{tr}\left[\partial_\mu\phi^\dagger\partial^\mu\phi\right] + \mathcal{L}_{\text{gh}} + \mathcal{L}_{\text{gf}} \\ &- y\left[Q_i^{L\dagger}\phi_{ij}Q_j^R + Q_i^{R\dagger}\phi_{ij}^\dagger Q_j^L\right] - u\operatorname{tr}\left[\phi^\dagger\phi\phi^\dagger\phi\right] - v\operatorname{tr}\left[\phi^\dagger\phi\right]\operatorname{tr}\left[\phi^\dagger\phi\right]\end{aligned} \tag{7}$$

featuring a $SU(N_c)$ gauge sector, N_f quark-like Dirac fermions as well as a complex and uncharged $N_f \times N_f$ scalar meson matrix, coupled together by a single Yukawa interaction and two scalar quartic couplings. The family of perturbatively renormalisable models (7) contains scalar, fermionic, and gauge degrees of freedom; it is known to display regimes with asymptotic freedom, infrared freedom, and regimes with interacting IR and UV fixed points, depending on the field multiplicities N_c and N_f. The models (7) have received some attention

in recent years as they provide perturbatively controlled examples for asymptotically safe QFTs in four spacetime-dimensions [35–37].[4]
The model (7) has been chosen to highlight some of ARGES key capabilities over complementary packages. Specifically, neither SARAH nor PyR@TE can handle general $SU(N_c)$ gauge groups with N_c unspecified, and neither of them can define a $N_f \times N_f$ two-index scalar field as input. Also, N_f Dirac fermions with N_f unspecified cannot be handled by SARAH (but PyR@TE 3). Finally, specifying the correct contractions of the Yukawa vertex and quartics is problematic in both packages, and so is the disentangling of the quartic RGEs.[5]
For ARGES, no such limitations apply which we start to illustrate by loading the code.

```
<<ARGES`
Reset[];
```

ARGES is stateful, and Reset[] wipes any previous input without affecting the kernel memory. It is not necessary to invoke it for the first run, but recommended in notebooks as cells may be re-evaluated. Next, the gauge sector will be specified, starting with the number of gauge groups.

```
NumberOfSubgroups = 1;
Gauge[g, SU[Nc], {Nc^2 - 1}];
```

Then, the functions Gauge[__] are called once for each gauge group, which implies the ordering of gauge indices. The first argument is the symbol of the gauge coupling, followed by the group, or any place holder for an unknown unspecified one thereof. Finally a list of length NumberOfSubgroups denotes the multiplicity of the gauge bosons[6] under each gauge group, in the implied ordering. Next, the matter content can be specified. We will start with registering the fermions in terms of their Weyl components.

```
WeylFermion[QL, Nf, {Nc}];
WeylFermion[QR, Nf, {Nc}];
```

Hereby, the first argument is the name of the fermion, the second the number of flavours and the third a list of its gauge multiplicities in the same order as before. For $U(1)$ gauge groups, the charge of the field is to be inserted here. Scalar matter can be inserted with

```
ComplexScalar[H, {Nf, Nf}, {1}];
```

which will add two real components Re[H] and Im[H]. Whenever the complex field is specified somewhere, the decomposition H = (Re[H] + iIm[H]) $/\sqrt{2}$ is then automatically inserted. Alternatively, one may add the components manually via RealScalar[_,_,_] with the same calling conventions. The syntax

[4] Further works cover large-N equivalences [38], extensions to semi-simple gauge groups [39], supersymmetry [40], and SM extensions [41–44].

[5] However, as long as only the intact $SU\left(N_f\right) \times SU\left(N_f\right)$ flavour symmetry is considered, there is a trick to promote these to two gauge symmetries with vanishing couplings which should in principle allow model input for specific values of N_f and N_c.

[6] In the $U(1)$ case, it marks the charge instead.

is similar to the fermionic case, only the second argument is now a list of two elements, as scalars always carry two flavour indices. We are now in the position to add interactions. A Yukawa term with a single coupling `y` is inserted via

```
Yukawa[y, H, adj[QL], QR, {KroneckerDelta[#2,#3]&}, (
    KroneckerDelta[#1,#3] KroneckerDelta[#2,#4])& ];
```

The second to fourth arguments represent the scalar and fermionic fields involved. The fifth argument is a list of `NumberOfSubgroups` elements, each being a function of three arguments representing the contractions of gauge indices of the scalar and two fermionic fields involved, in the order of appearance at this vertex. To optimise the simplification, *Mathematica*'s built-in function `KroneckerDelta[_,_]` should be used for each contraction. The final argument is the contraction function of flavour indices and expects four arguments, the first two being indices of the scalar, and the second two for each of the fermions, again in the order of appearance. Obviously, we have used *Mathematica*'s capability to define anonymous functions as `(_)&`, with the nth argument denoted by $\#n$. In the same manner, quartics can now be added, keeping in mind that scalars have two flavour indices each.

```
ScalarQuartic[u, adj[H], H, adj[H], H, {1&}, (KroneckerDelta
    [#2,#3] KroneckerDelta[#4,#5] KroneckerDelta[#6,#7]
    KroneckerDelta[#8,#1])& ];
ScalarQuartic[v, adj[H], H, adj[H], H, {1&}, (KroneckerDelta
    [#2,#3] KroneckerDelta[#4,#1] KroneckerDelta[#6,#7]
    KroneckerDelta[#8,#5])& ];
```

The model is now defined, and we can already compute all the gauge invariants available via

```
ComputeInvariants[];
```

which will be stored as a substitution rule in `subInvariants`.
In addition, the functions `ScalarCubic[__]`, `ScalarMass[__]` and `FermionMass[__]` allow to specify couplings h_{abc}, μ^2_{ab} and m_{ij} in (1). For instance, a scalar mass term

```
ScalarMass[m2, adj[H], H, {1&}, (KroneckerDelta[#2,#3]
    KroneckerDelta[#4,#1])& ];
```

is compatible with the global symmetry of (7).

Alternatively, models with more conventional Yukawa and fermion mass matrices can be entered via `YukawaMat[__]` and `FermionMassMat[__]` respectively. For instance, a gauge theory with fermions $\psi_{L,R}$ and complex singlet scalar ϕ can contain a Yukawa term $\frac{1}{n}Y_{ij}(\psi^\dagger_L)_{ia}\,\phi\,(\psi_R)_{ja}$ + h.c. Here, generations are counted via $i, j, \cdots$ while a refers to gauge components of the fermions, and n is a normalisation factor. This vertex is entered as

```
YukawaMat[Y, phi, adj[psiL], psiR, {KroneckerDelta[#2,#3]&}, n];
```

into `ARGES`. Moreover,

```
VEV[v, Re[phi], {1,1,1}, 1/Sqrt[2]];
```

assigns a correctly normalised VEV to the real part of the scalar, while the third argument relates to the two generation indices and the gauge index of the affected component.

5.2. Obtaining output

Now, the model (7) from the previous section

```
<<ARGES`
Reset[];
NumberOfSubgroups = 1;
Gauge[g, SU[Nc], {Nc^2 - 1}];
WeylFermion[QL, Nf, {Nc}];
WeylFermion[QR, Nf, {Nc}];
ComplexScalar[H, {Nf, Nf}, {1}];
Yukawa[y, H, adj[QL], QR, {KroneckerDelta[#2,#3]&}, (
    KroneckerDelta[#1,#3] KroneckerDelta[#2,#4])& ];
ScalarQuartic[u, adj[H], H, adj[H], H, {1&}, (KroneckerDelta
    [#2,#3] KroneckerDelta[#4,#5] KroneckerDelta[#6,#7]
    KroneckerDelta[#8,#1])& ];
ScalarQuartic[v, adj[H], H, adj[H], H, {1&}, (KroneckerDelta
    [#2,#3] KroneckerDelta[#4,#1] KroneckerDelta[#6,#7]
    KroneckerDelta[#8,#5])& ];
ComputeInvariants[];
```

will be analysed up to one loop in the gauge, Yukawa and scalar couplings. The β-functions for the gauge coupling `g` is obtained via

```
In[1]:=   (4 π)^2 β[g, 1] //. subInvariants // Expand
Out[1]=  -11/3 Nc g^3 + 2/3 Nf g^3
```

where the second argument indicates the loop order. For any other couplings, RG equations are extracted by specifying external fields and their indices. In order to properly normalise and disentangle the system of couplings, this has to be done at tree level first. For the Yukawa interaction, the syntax is

```
In[2]:=   β[Re[H], adj[QL], QR, {i1, i2, 1}, {j1, a}, {j2, b}, 0]
Out[2]=   y δ_{i1,j1} δ_{i2,j2} δ_{a,b} / Sqrt[2]
```

where the first three arguments specify the scalar and fermions at the vertex of interest, followed by lists of the quantum numbers of each field, in that order. The leading elements of that list are flavour (one for fermions, two for scalars), and the remaining components gauge indices.[7] Finally the last argument is again the loop order, with 0 indicating tree-level. From this output, the desirable normalisation and index structure can be read off.

Now, the one-loop β-function of `y` is extracted by changing the last argument, and taking the correct normalisation into account.

```
In[3]:=   (4 π)^2 Sqrt[2] β[Re[H], adj[QL], QR, {1,1,1}, {1,1},
    {1,1}, 1] //. subInvariants // Expand
Out[3]=   3/Nc g^2 y - 3 Nc g^2 y + Nc y^2 conj[y] + Nf y^2 conj[y]
```

[7] In case of a $U(1)$ gauge group, a dummy index 1 has to be provided.

For the scalar quartics, syntax and procedure is very similar. As there are two couplings u and v, we need to compute two vertices at tree level and disentangle the quartics. One of many possible choices is

```
In[4]:=   β[adj[H], H, adj[H], H, {1,1,1}, {1,1,1}, {1,1,1},
    {1,1,1}, 0] // Expand
Out[4]=   u + v
In[5]:=   β[adj[H], H, adj[H], H, {2,2,1}, {2,2,1}, {1,1,1},
    {1,1,1}, 0] // Expand
Out[5]=   v/3
```

which implies that one-loop expressions for the quartics can be obtained via the following expression.

```
In[6]:=   bv = 3 (4 π)^2 β[adj[H], H, adj[H], H, {2,2,1}, {2,2,1},
    {1,1,1}, {1,1,1}, 1] //. subInvariants // Expand
Out[6]=   12 u^2 + 16 Nf u v + 16 v^2 + 4 Nf^2 v^2 + 4 Nc v y conj[y]
In[7]:=   bu = - bv + (4 π)^2 β[adj[H], H, adj[H], H, {1,1,1},
    {2,2,1}, {1,1,1}, {1,1,1}, 1] //. subInvariants // Expand
Out[7]=   8 Nf u^2 + 24 u v + 4 Nc u y conj[y] - 2 Nc y^2 conj[y]^2
```

In fact, the syntax `β[__]`, with the leading arguments being all fields at the vertex followed by their respective lists of indices, is also utilised to obtain scalar cubic interactions as well as fermion and scalar mass terms. To avoid confusion with the latter, scalar and fermionic field anomalous dimensions

```
In[8]:=   (4 π)^2 γ[adj[H], H, {1,1,1}, {1,1,1}, 1] //.
    subInvariants // Expand
Out[8]=   Nc y conj[y]
In[9]:=   (4 π)^2 γ[adj[QL], QL, {1,1}, {1,1}, 1] //. subInvariants
    // Expand
Out[9]=   - ξ g^2 /(2 Nc) + Nc ξ/2 g^2 + Nf/2 y conj[y]
```

use the syntax `γ[__]` instead. The output obtained in this section matches the analytic results in [35], and it is straightforward to extend results to higher loop orders [37].
This concludes the presentation of `ARGES` basic input and output functionality, which should be sufficient for a first instruction. In the next section we will turn towards a more advanced example, demonstrating a more specialised use of `ARGES`.

5.3. Advanced capabilities

Under the hood, `ARGES` optimises index summations by employing simplification rules like resolving `KroneckerDelta` contractions directly instead of brute-forcing `Sum[__]`. This is critical for `ARGES` performance, as such contraction sums are typically very long. Internally, they are simplified as expressions of the shape `SimplifySum[__]`, and only in the last step converted back to *Mathematica*'s built-in function `Sum[__]` (the syntax is compatible), if required.[8] This simplification can be enforced on any expression using `SimplifyProduct[_]`.

[8] Disabling this last step with `DisableNativeSums[]` may in fact give a performance boost.

The user may inject custom simplification rules into this mechanism by adding them to the list `subSimplifySum`. This will be demonstrated by example of a theory with a $SU(n)$ global symmetry and real scalars in the adjoint. The action of interest is

$$\begin{aligned}\mathcal{L} =& \tfrac{1}{2}\partial_\mu\phi^A\partial^\mu\phi^A - \tfrac{1}{2}m^2\phi^A\phi^A \\ & - \tfrac{1}{4}\lambda_1\left(\phi^A\phi^A\right)^2 - \tfrac{1}{2}\lambda_2\,\phi^A\phi^B\phi^C\phi^D\left(T^{ABCD}+T^{DCBA}\right),\end{aligned} \tag{8}$$

where the object

$$T^{ABC\ldots} = \mathrm{tr}\left[t^A t^B t^C \ldots\right]. \tag{9}$$

are traces of t^A, the generator of the fundamental representation of $SU(n)$. These structures are difficult to resolve in general, but due to the Fierz relation

$$t^A_{ab}t^A_{cd} = \tfrac{1}{2}\left(\delta_{ad}\delta_{bc} - \tfrac{1}{n}\delta_{ab}\delta_{cd}\right), \tag{10}$$

together with the Dynkin index definition and the tracelessness of the generators, the following relations can be inferred

$$\begin{aligned}&T = \mathrm{tr}\left[\mathbb{1}\right] = n,\\ &T^A = 0,\\ &T^{AB} = \tfrac{1}{2}\delta^{AB},\\ &\sum_A T^{B_1\cdots B_i\,A\,C_1\cdots C_j\,A\,D_1\cdots D_k} = \\ &\qquad\tfrac{1}{2}\left(T^{B_1\cdots B_i\,D_1\cdots D_k}T^{C_1\cdots C_j} - \tfrac{1}{n}T^{B_1\cdots B_i\,C_1\cdots C_j\,D_1\cdots D_k}\right)\\ &\sum_A T^{B_1\cdots B_i\,A\,C_1\cdots C_j}T^{D_1\cdots D_k\,A\,E_1\cdots E_l} = \\ &\qquad\tfrac{1}{2}\left(T^{B_1\cdots B_i\,E_1\cdots E_l\,D_1\cdots D_k\,C_1\cdots C_j} - \tfrac{1}{n}T^{B_1\cdots B_i\,C_1\cdots C_j}T^{D_1\cdots D_k\,E_1\cdots E_l}\right).\end{aligned} \tag{11}$$

This also holds true when index ranges such as $B_1\cdots B_i$ are empty. Using `ARGES`, the model (8) can be analysed by entering (11) into `subSimplifySum` using patterns. It can be assumed that sums over gauge and flavour indices are expanded and each term contained in `SimplifySum[_]` expression which is not nested.

```
<<ARGES`
Reset[];

NumberOfSubgroups=0;
RealScalar[phi, {n^2-1,1}, {}];
ScalarMass[m2, phi, phi, {}, KroneckerDelta[#1,#3]/2&];
ScalarQuartic[λ1, phi, phi,  phi, phi, {}, KroneckerDelta
    [#1,#3] KroneckerDelta[#5,#7]/4&];
ScalarQuartic[λ2, phi, phi,  phi, phi, {}, (T[#1,#3,#5,#7] + T
    [#7,#5,#3,#1])/2&];

subSimplifySum = {

   T[] -> n,
```

```
13      T[_] :> 0,
14      T[A_, B_] :> 1/2 KroneckerDelta[A,B],
15
16      SimplifySum[c_ T[B___, A_, C___, A_, D___], sum1___, {A_, __
            }, sum2___] :> SimplifySum[c/2 (T[B, D] T[C] - T[B, C, D
            ]/n ), sum1, sum2],
17      SimplifySum[T[B___, A_, C___, A_, D___], sum1___, {A_, __},
            sum2___] :> SimplifySum[1/2 (T[B, D] T[C] - T[B, C, D]/n
             ), sum1, sum2],
18
19      SimplifySum[c_ T[B___, A_, C___] T[D___, A_, E___], sum1___,
             {A_, __}, sum2___] :> SimplifySum[c/2 (T[B, E, D, C]  -
             T[B, C] T[D, E]/n ), sum1, sum2],
20      SimplifySum[T[B___, A_, C___] T[D___, A_, E___], sum1___, {
            A_, __}, sum2___] :> SimplifySum[1/2 (T[B, E, D, C]  - T
            [B, C] T[D, E]/n ), sum1, sum2],
21      SimplifySum[c_ T[B___, A_, C___]^2, sum1___, {A_, __},
            sum2___] :> SimplifySum[c/2 (T[B, C, B, C]  - T[B, C]^2/
            n ), sum1, sum2],
22      SimplifySum[T[B___, A_, C___]^2, sum1___, {A_, __}, sum2___]
             :> SimplifySum[1/2 (T[B, C, B, C]  - T[B, C]^2/n ),
            sum1, sum2]
23  };
```

While the first three lines in `subSimplifySum` are the same as in (11), the second and third block cover the last two lines in (11) respectively, and account for complications due to possible prefactors and squares in the expression. Commencing with the evaluation, mass anomalous dimensions can be obtained in the usual way.

```
In[1]:=  β[phi, phi, {1,1}, {1,1}, 0]
Out[1]=  m2/24
In[2]:=  (4 π)^2 24/m2 β[phi, phi, {1,1}, {1,1}, 1] // Expand
Out[2]=  2 λ1 + 2 n^2 λ1 - 6 λ2/n + 4 n λ2
In[3]:=  (4 π)^4 24/m2 β[phi, phi, {1,1}, {1,1}, 2] // Expand
Out[3]=  - 10 λ1^2 - 10 n^2 λ1^2 + 60 λ1 λ2/n - 40 n λ1 λ2 + 30 λ2^2
         - 90 λ2^2/n^2 - 5 n^2 λ2^2
```

In the quartic sector, the structures `T[___]` will reappear at both tree and loop level, and β-functions can be extracted from their prefactors. Alternatively, the `T[___]` may be completely removed by contracting external indices. In fact, we can even enforce the evaluation of the algebra as defined above by calling `SimplifyProduct[_]` on the contraction defined in terms of `SimplifySum[___]` instead of `Sum[___]`. At tree level, one possibility is for instance

```
In[4]:=  c1 = SimplifyProduct[
          SimplifySum[
           12 β[phi, phi, phi, phi, {a,1}, {a,1}, {1,1}, {1,1}, 0],
           {a, 1, n^2-1}
          ]
        ]
Out[4]=  λ1 + n^2 λ1 - 3 λ2/n + 2 n λ2
In[5]:=  c2 = SimplifyProduct[
             SimplifySum[
                96 T[a, b, c, d] β[phi, phi, phi, phi, {a,1}, {b,1},
                    {c,1}, {d,1}, 0],
```

```
            {a, 1, n^2-1},
            {b, 1, n^2-1},
            {c, 1, n^2-1},
            {d, 1, n^2-1}
        ]
    ]
Out[5]= 6 λ1/n - 10 n λ1 + 4 n^3 λ1 + 24 λ2 - 18 λ2/n^2 - 7 n^2 λ2 + n^4 λ2
```

which is indeed free of the trace. This has to be matched up with the loop-level expressions

```
In[6]:= b1 = SimplifyProduct[
            SimplifySum[
               12 (4 π)^2 β[phi, phi, phi, phi, {a,1}, {a,1}, {1,1},
                    {1,1}, 1],
               {a, 1, n^2-1}
            ]
         ] // Simplify
Out[6]= 2 (7 + 8 n^2 + n^4) λ1^2 + 4 (7 + n^2) (-3 + 2 n^2) λ1 λ2/n
        + 2 (63 - 30 n^2 + 7 n^4) λ2^2/n^2
In[7]:= b2 = SimplifyProduct[
            SimplifySum[
               96 (4 π)^2 T[a, b, c, d] β[phi, phi, phi, phi, {a,1},
                    {b,1}, {c,1}, {d,1}, 1],
               {a, 1, n^2-1},
               {b, 1, n^2-1},
               {c, 1, n^2-1},
               {d, 1, n^2-1}
            ]
         ] // Simplify
Out[7]= 4 (21/n - 32 n + 9 n^3 + 2 n^5) λ1^2
        + 8 (93 - 63/n^2 - 37 n^2 + 7 n^4) λ1 λ2
        + 4 (-1 + n^2) (-3 + n^2)(63 - 6 n^2 + n^4) λ2^2/n^3
```

which is left as an exercise to the reader. Putting everything together, we find that ARGES provides us with the one and two loop anomalous dimensions and the one loop renormalisation group equations as

$$\begin{aligned}
\gamma_{m^2}^{1\ell} &= 2(n^2+1)\lambda_1 + \frac{4n^2-6}{n}\,\lambda_2\,, \\
\gamma_{m^2}^{2\ell} &= -10(n^2+1)\lambda_1^2 - \frac{20(2n^2-3)}{n}\lambda_1\lambda_2 - \frac{5(n^4-6n^2+18)}{n^2}\lambda_2^2\,, \\
\beta_{\lambda_1}^{1\ell} &= 2(7+n^2)\lambda_1^2 + \frac{4(2n^2-3)}{n}\,\lambda_1\lambda_2 + \frac{6(n^2+3)}{n^2}\lambda_2^2\,, \\
\beta_{\lambda_2}^{1\ell} &= 24\lambda_1\lambda_2 + \frac{4(n^2-9)}{n}\lambda_2^2\,.
\end{aligned} \tag{12}$$

This study can be extended to higher loop orders. At one-loop, our results confirm the findings of [45] up to normalisation conventions. This concludes the illustration of some of the more specialised capabilities of ARGES.

5.4. Validity checks

The capabilities and the accuracy of ARGES have been tested extensively. This includes many cross-checks against existing results in the literature, cross-checks

against the packages SARAH and PyR@TE, and cross-checks with the traditional "by hand" extraction of expressions up to three loops, including for the model (7) [37], and others. ARGES has also been used as the primary tool for the derivation of β-functions in a number of studies [13, 15, 37, 38, 43, 46], and we have confirmed that it represents a capable alternative to SARAH and PyR@TE for processing extensions of the Standard Model.

6. Summary & conclusion

The sheer increase of complexity with loop order or field content in the computation of renormalisation group equations in particle physics necessitates powerful automation tools. Here, we have introduced ARGES, a new framework for the computation of renormalisation group equations. A key design feature of ARGES over other tools such as SARAH and PyR@TE is that it follows a more algebraic approach to resolve index contractions. Most notably, ARGES can handle models which no pre-existing framework has been able to process, including theories with general gauge groups and representations, unspecified multiplicity of matter fields, matrix scalars, highly complex potentials, and even unknown vertex contractions. Its capabilities cover any perturbatively renormalisable quantum field theory with Lagrangian (1). Setup, input and output of ARGES are straightforward, and its interactive API invites a quick and dynamic working style. At the same time, ARGES is flexible and powerful, also offering a maximum of user control. We hope that ARGES might prove useful for theorists, model builders, and practitioners alike.

Acknowledgements
We thank to Lohan Sartore and Alexander V. Bednyakov for useful comments on the manuscript. TS has been supported by the Deutsche Akademische Austauschdienst (DAAD) under the Grant 57314657. DL is supported by the Science and Technology Facilities Council (STFC) under the Consolidated Grant ST/T00102X/1.

Note added
While finishing up this publication, we have been informed that a similar framework is currently in development [47], which is not associated with ARGES. The project appears to be younger, but shares some of ARGES features due to overlapping design goals. The most prominent difference is the use of the basis [14, 28] for the template action. The corresponding preprint [48] has appeared while this work was under review.

References

[1] M. E. Machacek and M. T. Vaughn, *Two Loop Renormalization Group Equations in a General Quantum Field Theory. 1. Wave Function Renormalization*, *Nucl. Phys.* **B222** (1983) 83.

[2] M. E. Machacek and M. T. Vaughn, *Two Loop Renormalization Group Equations in a General Quantum Field Theory. 2. Yukawa Couplings*, *Nucl. Phys.* **B236** (1984) 221.

[3] M. E. Machacek and M. T. Vaughn, *Two Loop Renormalization Group Equations in a General Quantum Field Theory. 3. Scalar Quartic Couplings*, *Nucl. Phys.* **B249** (1985) 70.

[4] M.-x. Luo, H.-w. Wang and Y. Xiao, *Two loop renormalization group equations in general gauge field theories*, *Phys. Rev.* **D67** (2003) 065019 [hep-ph/0211440].

[5] T. Curtright, *Three loop charge renormalization effects due to quartic scalar selfinteractions*, *Phys.Rev.D* **21** (1980) 1543.

[6] I. Jack and H. Osborn, *Analogs for the c Theorem for Four-dimensional Renormalizable Field Theories*, *Nucl. Phys. B* **343** (1990) 647.

[7] I. Jack and C. Poole, *Scheme invariants in ϕ^4 theory in four dimensions*, *Phys. Rev. D* **98** (2018) 065011 [1806.08598].

[8] A. G. M. Pickering, J. A. Gracey and D. R. T. Jones, *Three loop gauge beta function for the most general single gauge coupling theory*, *Phys. Lett.* **B510** (2001) 347 [hep-ph/0104247].

[9] M. Sperling, D. Stöckinger and A. Voigt, *Renormalization of vacuum expectation values in spontaneously broken gauge theories*, *JHEP* **07** (2013) 132 [1305.1548].

[10] M. Sperling, D. Stöckinger and A. Voigt, *Renormalization of vacuum expectation values in spontaneously broken gauge theories: Two-loop results*, *JHEP* **01** (2014) 068 [1310.7629].

[11] L. N. Mihaila, J. Salomon and M. Steinhauser, *Renormalization constants and beta functions for the gauge couplings of the standard model to three-loop order*, *Phys.Rev.D* **86** (2012) 096008 [1208.3357].

[12] L. Mihaila, *Three-loop gauge beta function in non-simple gauge groups*, *PoS* **RADCOR2013** (2013) 060.

[13] I. Schienbein, F. Staub, T. Steudtner and K. Svirina, *Revisiting RGEs for general gauge theories*, *Nucl. Phys.* **B939** (2019) 1 [1809.06797].

[14] C. Poole and A. E. Thomsen, *Constraints on 3- and 4-loop β-functions in a general four-dimensional quantum field theory*, *JHEP* **09** (2019) 055 [1906.04625].

[15] T. Steudtner, *General scalar renormalisation group equations at three-loop order*, *JHEP* **12** (2020) 012 [2007.06591].

[16] L. Sartore, *General RGEs for dimensionful couplings in the* $\overline{\text{MS}}$ *scheme*, *Phys. Rev. D* **102** (2020) 076002 [2006.12307].

[17] Wolfram Research, Inc., *Mathematica, Version 12.0*, https://www.wolfram.com/mathematica (Champaign, IL, 2020) .

[18] T. Steudtner, *ARGES*, https://github.com/tomsteu/ARGES (2017) .

[19] Free Software Foundation, *GNU General Public License v3*, https://www.gnu.org/licenses/gpl-3.0.html (2007) .

[20] F. Staub, *SARAH*, 0806.0538.

[21] F. Staub, *From Superpotential to Model Files for FeynArts and CalcHep/CompHep*, *Comput. Phys. Commun.* **181** (2010) 1077 [0909.2863].

[22] F. Staub, *Automatic Calculation of supersymmetric Renormalization Group Equations and Self Energies*, *Comput. Phys. Commun.* **182** (2011) 808 [1002.0840].

[23] F. Staub, *SARAH 3.2: Dirac Gauginos, UFO output, and more*, *Comput. Phys. Commun.* **184** (2013) 1792 [1207.0906].

[24] F. Staub, *SARAH 4 : A tool for (not only SUSY) model builders*, *Comput. Phys. Commun.* **185** (2014) 1773 [1309.7223].

[25] F. Lyonnet, I. Schienbein, F. Staub and A. Wingerter, *PyR@TE: Renormalization Group Equations for General Gauge Theories*, *Comput. Phys. Commun.* **185** (2014) 1130 [1309.7030].

[26] F. Lyonnet and I. Schienbein, *PyR@TE 2: A Python tool for computing RGEs at two-loop*, *Comput. Phys. Commun.* **213** (2017) 181 [1608.07274].

[27] L. Sartore and I. Schienbein, *PyR@TE 3*, 2007.12700.

[28] C. Poole and A. Thomsen, *Weyl Consistency Conditions and* γ_5, *Phys. Rev. Lett.* **123** (2019) 041602 [1901.02749].

[29] C. Bollini and J. Giambiagi, *Lowest order "divergent graphs" in* ν*-dimensional space*, *Phys. Lett. B* **40** (1972) 566.

[30] C. Bollini and J. Giambiagi, *Dimensional Renormalization: The Number of Dimensions as a Regularizing Parameter*, *Nuovo Cim. B* **12** (1972) 20.

[31] G. 't Hooft, *Dimensional regularization and the renormalization group*, *Nucl. Phys. B* **61** (1973) 455.

[32] W. A. Bardeen, A. Buras, D. Duke and T. Muta, *Deep Inelastic Scattering Beyond the Leading Order in Asymptotically Free Gauge Theories*, *Phys. Rev. D* **18** (1978) 3998.

[33] R. M. Fonseca, *Calculating the renormalisation group equations of a SUSY model with Susyno*, *Comput. Phys. Commun.* **183** (2012) 2298 [1106.5016].

[34] F. Lyonnet, *PyLie*, https://github.com/dibus2/pylie (2016) .

[35] D. F. Litim and F. Sannino, *Asymptotic safety guaranteed*, *JHEP* **12** (2014) 178 [1406.2337].

[36] D. F. Litim, M. Mojaza and F. Sannino, *Vacuum stability of asymptotically safe gauge-Yukawa theories*, *JHEP* **01** (2016) 081 [1501.03061].

[37] A. D. Bond, D. F. Litim, G. Medina Vazquez and T. Steudtner, *UV conformal window for asymptotic safety*, *Phys. Rev. D* **97** (2018) 036019 [1710.07615].

[38] A. D. Bond, D. F. Litim and T. Steudtner, *Asymptotic safety with Majorana fermions and new large N equivalences*, *Phys. Rev. D* **101** (2020) 045006 [1911.11168].

[39] A. D. Bond and D. F. Litim, *More asymptotic safety guaranteed*, *Phys. Rev. D* **97** (2018) 085008 [1707.04217].

[40] A. D. Bond and D. F. Litim, *Asymptotic safety guaranteed in supersymmetry*, *Phys. Rev. Lett.* **119** (2017) 211601 [1709.06953].

[41] A. D. Bond, G. Hiller, K. Kowalska and D. F. Litim, *Directions for model building from asymptotic safety*, *JHEP* **08** (2017) 004 [1702.01727].

[42] G. Hiller, C. Hormigos-Feliu, D. F. Litim and T. Steudtner, *Anomalous magnetic moments from asymptotic safety*, *Phys. Rev. D* **102** (2020) 071901 [1910.14062].

[43] G. Hiller, C. Hormigos-Feliu, D. F. Litim and T. Steudtner, *Model Building from Asymptotic Safety with Higgs and Flavor Portals*, *Phys. Rev. D* **102** (2020) 095023 [2008.08606].

[44] S. Bißmann, G. Hiller, C. Hormigos-Feliu and D. F. Litim, *Multi-lepton signatures of vector-like leptons with flavor*, 2011.12964.

[45] M. Hnatič, G. A. Kalagov and M. Y. Nalimov, *On the first-order phase transition in $SU(N)$ matrix models*, *Nucl. Phys. B* **955** (2020) 115060.

[46] T. Steudtner, *Asymptotic safety: from perturbatively exact models to particle physics*, Ph.D. thesis, Sussex U., 2020.

[47] A. E. Thomsen, *RGBeta*, https://github.com/aethomsen/RGBeta (2019) .

[48] A. E. Thomsen, *RGBeta: A Mathematica Package for the Evaluation of Renormalization Group β-Functions*, 2101.08265.