FERMILAB-PUB-20-275-T, SISSA 17/2020/FISI, IPPP/20/30

ULYSSES: Universal LeptogeneSiS Equation Solver

A. Granelli[a], K. Moffat[b], Y. F. Perez-Gonzalez[c,d,e], H. Schulz[f], J. Turner[c]

[a] *SISSA/INFN, Via Bonomea 265, I-34136 Trieste, Italy.*

[b] *Institute for Particle Physics Phenomenology, Durham University, Durham, UK*

[c] *Fermi National Accelerator Laboratory, Batavia, IL, 60510-0500, USA*

[d] *Department of Physics & Astronomy, Northwestern University, Evanston, IL 60208, USA*

[e] *Colegio de Física Fundamental e Interdisciplinaria de las Américas (COFI), 254 Norzagaray street, San Juan, Puerto Rico 00901*

[f] *Department of Physics, University of Cincinnati, Cincinnati, OH 45219, USA*

Abstract

ULYSSES is a python package that calculates the baryon asymmetry produced from leptogenesis in the context of a type-I seesaw mechanism. The code solves the semi-classical Boltzmann equations for points in the model parameter space as specified by the user. We provide a selection of predefined Boltzmann equations as well as a plugin mechanism for externally provided models of leptogenesis. Furthermore, the ULYSSES code provides tools for multi-dimensional parameter space exploration. The emphasis of the code is on user flexibility and rapid evaluation. It is publicly available at https://github.com/earlyuniverse/ulysses.

1. Introduction

The two leading theories that explain the excess of matter over antimatter are leptogenesis [1] and electroweak baryogenesis [2, 3]. The latter theory has attracted much attention given its close relation with Higgs physics and much of the model parameter space has been explored. The former, in its various manifestations, appeals to many given its connection to neutrino masses and mixing. Although the mechanisms which generate the baryon asymmetry in both scenarios are vastly different, a common feature is the need to solve Boltzmann equations (BE) for points in the relevant model parameter space. ULYSSES is a *python* package that solves the semi-classical BE for leptogenesis in the context of a type-I seesaw mechanism and, to the authors knowledge, is the first publicly available code for this task.

The provided momentum-averaged BEs are based on the out-of-equilibrium decays of right-handed neutrinos and resonant leptogenesis. Effects such as lepton flavour, scatterings

and spectator processes are also provided if the user wishes to apply them. For a given point in the model parameter space, ULYSSES calculates the final baryon asymmetry (provided in terms of the baryon-to-photon ratio, η_B, the baryonic yield, Y_B, and the baryonic density parameter, $\Omega_B h^2$) and plots the lepton asymmetry number density as a function of the evolution parameter. For the user who wishes to undertake a multi-dimensional exploration of the parameter space, we provide instructions on how to use MULTINEST [4] in combination with ULYSSES. This allows visualisation of the multi-dimensional parameter space which is consistent with the measured baryon-to-photon ratio [5, 6]. We have designed the code in a modular fashion, separating the physics of the baryon asymmetry production from the parameter space exploration.

As this paper is a manual on how to use the ULYSSES code, we refrain from discussing the different regimes and subtleties of the leptogenesis mechanism and instead refer the reader to Refs. ([7, 8, 9]) for broad reviews on various aspects of thermal and resonant leptogenesis.

The paper is organised as follows: in Section (2) we discuss the parametrisation and normalisation conventions ULYSSES applies. In Section (3), we describe the preprovided BEs and follow in Section (4) with installation instructions and a discussion of code dependencies. In Section (5), we explain the structure of the code and show the user how to calculate the baryon asymmetry for a point in the model parameter space. Scripts and examples of multi-dimensional parameter space exploration, as well as user options, are presented in Section (6) and finally we make concluding remarks in Section (7).

2. Conventions

We begin in Section (2.1) by providing details on our parametrisation of the Yukawa matrix and then follow in Section (2.2) with a discussion of our applied normalisation of the BEs.

2.1. Yukawa matrix

One of the simplest extensions of the Standard Model (SM) that explains small neutrino masses is the type-I seesaw mechanism [10, 11, 12]. Leptogenesis can be regarded as a cosmological consequence of the seesaw mechanism and provides an elegant way of explaining tiny neutrino masses and the baryon asymmetry of the Universe.

This mechanism introduces a set of heavy right-handed Majorana neutrino fields N_i and

augments the SM Lagrangian to include the following terms

$$\mathcal{L} = i\overline{N_i}\partial\!\!\!/ N_i - Y_{\alpha i}\overline{L_\alpha}\tilde{\Phi}N_i - \frac{1}{2}M_i\overline{N_i^c}N_i + \text{h.c.}\,, \tag{1}$$

where Y is the Yukawa matrix and Φ the Higgs doublet, $\Phi^T = (\phi^+, \phi^0)$ and $\tilde{\Phi} = i\sigma_2\Phi^*$, and $L^T = \left(\nu_L^T, l_L^T\right)$ is the leptonic doublet. For convenience and without loss of generality, we have chosen the basis in which the Majorana mass term is diagonal. After electroweak symmetry breaking, at tree-level, the light neutrino mass matrix (at first order in the seesaw expansion) is

$$m^{\text{tree}} \approx m_D M^{-1} m_D^T\,, \tag{2}$$

where $m_D = Yv$ is the Dirac mass matrix that develops once the Higgs acquires the vacuum expectation value, v. We use the conventions that $v = 174.0$ GeV and m^{tree} does not have a minus sign. We parametrise the Yukawa matrix in analogy with Casas and Ibarra [13]:

$$Y = \frac{1}{v}U\sqrt{\hat{m}_\nu}R^T\sqrt{M_R}\,, \tag{3}$$

where U is the leptonic mixing matrix, $\hat{m}_\nu$ is the diagonal light neutrino mass matrix, R is a complex, orthogonal matrix and M_R is the diagonal mass matrix of the heavy right-handed neutrinos. Using this parametrisation the model parameter space is 18 dimensional where nine parameter are associated to the low-energy scale physics and the remaining nine parameters are associated to the high-scale physics of the right-handed neutrinos. This parametrisation has the benefit that neutrino masses and mixing from oscillation data are recovered[1].

We apply the PDG convention [5] to parametrise the PMNS matrix:

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{pmatrix} \begin{pmatrix} c_{13} & 0 & s_{13}e^{-i\delta} \\ 0 & 1 & 0 \\ -s_{13}e^{i\delta} & 0 & c_{13} \end{pmatrix} \begin{pmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{i\frac{\alpha_{21}}{2}} & 0 \\ 0 & 0 & e^{i\frac{\alpha_{31}}{2}} \end{pmatrix}\,, \tag{4}$$

where $c_{ij} \equiv \cos\theta_{ij}$, $s_{ij} \equiv \sin\theta_{ij}$, δ is the Dirac phase and α_{21}, α_{31} are the Majorana phases which vary between $0 \leq \alpha_{21}, \alpha_{31} \leq 4\pi$. The R-matrix has the form:

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{\omega_1} & s_{\omega_1} \\ 0 & -s_{\omega_1} & c_{\omega_1} \end{pmatrix} \begin{pmatrix} c_{\omega_2} & 0 & s_{\omega_2} \\ 0 & 1 & 0 \\ -s_{\omega_2} & 0 & c_{\omega_2} \end{pmatrix} \begin{pmatrix} c_{\omega_3} & s_{\omega_3} & 0 \\ -s_{\omega_3} & c_{\omega_3} & 0 \\ 0 & 0 & 1 \end{pmatrix}\,, \tag{5}$$

[1]While the Casas-Ibarra parametrisation is convenient and widely used, ULYSSES allows the user to provide their own Yukawa matrix and we detail how to do this in Section (5).

where $c_{\omega_i} \equiv \cos\omega_i$, $s_{\omega_i} \equiv \sin\omega_i$ and the complex angles are given by $\omega_i \equiv x_i + iy_i$ for x, y free, real parameters.

The above parametrisation does not account for the radiative corrections to the light neutrino masses from the Z, W and Higgs boson. In some regions of the parameter space these corrections can be sizeable, such that the tree and one-loop contributions to the mass are comparable in magnitude [14]. As the tree and one-loop level contributions enter with different signs, a small neutrino mass compatible with data may be the consequence of cancellation between these two contributions. Such fine-tuning was quantified in [15] and depending on the user specified fine-tuning tolerance[2], the correction to the Casas-Ibarra parametrisation can be implemented in ULYSSES using [16]

$$Y = \frac{1}{v} U \sqrt{\hat{m}_\nu} R^T \sqrt{f(M)^{-1}}\,, \tag{6}$$

where

$$m_\nu = m^{\text{tree}} + m^{\text{1-loop}}\,, \tag{7}$$

with

$$\begin{aligned} m^{\text{1-loop}} &= \\ &- m_D \left(\frac{M}{32\pi^2 v^2} \left(\frac{\log\left(\frac{M^2}{m_H^2}\right)}{\frac{M^2}{m_H^2} - 1} + 3\frac{\log\left(\frac{M^2}{m_Z^2}\right)}{\frac{M^2}{m_Z^2} - 1} \right) \right) m_D^T \\ &= -\frac{1}{32\pi^2 v^2} m_D \text{diag}\left(g\left(M_1\right), g\left(M_2\right), g\left(M_3\right)\right) m_D^T\,, \end{aligned} \tag{8}$$

and

$$g\left(M_i\right) \equiv M_i \left(\frac{\log\left(\frac{M_i^2}{m_H^2}\right)}{\frac{M_i^2}{m_H^2} - 1} + 3\frac{\log\left(\frac{M_i^2}{m_Z^2}\right)}{\frac{M_i^2}{m_Z^2} - 1} \right)\,. \tag{9}$$

The contribution from two-loop corrections is usually small as these will be suppressed by an extra factor of the Yukawa couplings squared and a further factor $\mathcal{O}(10^{-2})$ from the loop integral. The matrix m_ν is rewritten in the factorised form using the leptonic mixing matrix:

$$m_\nu = U \hat{m}_\nu U^T\,, \tag{10}$$

where $\hat{m}_\nu$ is the positive diagonal matrix of light neutrino masses. The inclusion of the loop effect is a command line argument that we detail in Section (5).

[2]One can check that the two-loop contribution to the light neutrino mass is not larger than the one-loop contribution. This procedure is outlined in [15].

2.2. Normalisation and conversion of lepton to baryon asymmetry

The baryon asymmetry may be parametrised by the baryon-to-photon ratio, η_B, which is defined to be

$$\eta_B \equiv \frac{n_B - n_{\overline{B}}}{n_\gamma}\,, \tag{11}$$

where n_B, $n_{\overline{B}}$ and n_γ are the number densities of baryons, anti-baryons and photons, respectively. This quantity can be measured using two independent methods that probe the Universe at different stages of its evolution. Big-Bang nucleosynthesis (BBN) [5] and Cosmic Microwave Background radiation (CMB) data [6] are given by

$$\begin{aligned} \eta_{B\mathrm{BBN}} &= (5.80 - 6.60) \times 10^{-10}\,, \\ \eta_{B\mathrm{CMB}} &= (6.02 - 6.18) \times 10^{-10}\,, \end{aligned} \tag{12}$$

at 95% CL, respectively. As the uncertainties of the CMB measurement are smaller than those from BBN, this is the value taken in the code for the MultiNest scans. For completeness, ULYSSES also returns the baryonic yield and baryonic density parameter which follow from the baryon-to-photon ratio:

$$Y_B = \eta_B \cdot \frac{45\zeta(3)}{\pi^4 g_{*,s}(t_{\mathrm{rec}})}\,, \qquad \Omega_B h^2 = \eta_B \cdot \frac{m_p n_\gamma}{\rho_c h^{-2}}\,, \tag{13}$$

where $g_{*,s}(t_{\mathrm{rec}}) = 43/11$ are the entropic effective degrees of freedom at present, m_p is the proton mass and ρ_c is the critical density of the Universe [5].

The ULYSSES code solves BEs in terms of number densities of particles, or particle asymmetries, normalised to a comoving volume which contains one photon. This is equivalent to choosing the normalised equilibrium abundance of the right-handed neutrino to be $N_N^{eq}(z) = 3/8 \cdot z^2 K_2(z)$ which is the same convention applied in [17]. Therefore, the conversion from the $B - L$ number density to the baryon-to-photon ratio is as follows:

$$\eta_B \equiv \frac{N_B}{N_\gamma^{\mathrm{rec}}} = a_{\mathrm{sph}} \frac{N_{B-L}}{N_\gamma^{\mathrm{rec}}} = \frac{28}{79}\frac{1}{27} N_{B-L} = 0.013 N_{B-L}\,, \tag{14}$$

where N_{B-L} is the final $B - L$ asymmetry, $a_{\mathrm{sph}} = 28/79$ is the Standard model sphaleron factor and the $1/27$ factor derives from the dilution of the baryon asymmetry by photons for our choice of normalisation[3]. New physics can change the sphaleron factor, for instance in the supersymmetric Standard Model, $a_{\mathrm{sph}} = 8/23$. This will alter the overall normalisation factor, referred to as "normfact", which multiplies N_{B-L}. To allow for such new physics, normfact can be altered by the user through a command line option as detailed in Section (6.1).

[3]We note that another common convention is to normalise to one ultrarelativistic right-handed neutrino per comoving volume, see for example Ref. [18].

3. Built-in Boltzmann equations

In this section, we list and briefly discuss the preprovided BEs that are shipped with ULYSSES. We refer to BEs that incorporate off-diagonal flavour oscillations as density matrix equations (DME). The density matrix equations solved can be found in Ref. [19] while in the resonant case we solve the equations of Ref. [20]. Finally, the model which includes scattering is based on Ref. [17]. We provide example parameter cards for each model. They are located in the `examples` folder of the source tree. A quick overview of the contents of this section can be found in Table (1). The information about currently available models is also accessible by invoking the command `uls-models` which is available after installation of ULYSSES. We note that for all of the preprovided BEs, we have assumed a standard cosmology. From this assumption, the Boltzmann equations can be written in terms of the scale factor a which can be converted to an evolution in time, t. The time variable can be exchanged for a more convenient evolution parameter $z = M/T$ where M is the mass of the light right-handed neutrino and T is the plasma temperature. If the Hubble expansion rate evolved according to standard cosmology this is a convenient approach. We provide one example (1BE1Fsf) where the Hubble expansion rate is explicit and the evolution parameter is the scale factor. This would be a starting point for the user interested in implementing their own non-standard cosmology.

- **1DME** provides the semi-classical density matrix equations (DME) for one decaying right-handed neutrino:

$$\begin{aligned}
\frac{dN_{N_1}}{dz} &= -D_1\left(N_{N_1} - N_{N_1}^{\rm eq}\right) \\
\frac{dN_{\alpha\beta}^{B-L}}{dz} &= \epsilon_{\alpha\beta}^{(1)} D_1\left(N_{N_1} - N_{N_1}^{\rm eq}\right) - \frac{1}{2} W_1 \left\{P^{0(1)}, N^{B-L}\right\}_{\alpha\beta} \\
&\quad - \frac{\Gamma_\tau}{2Hz}\left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, N^{B-L}\right]\right]_{\alpha\beta} \\
&\quad - \frac{\Gamma_\mu}{2Hz}\left[\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \left[\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, N^{B-L}\right]\right]_{\alpha\beta},
\end{aligned} \tag{15}$$

  where N_{B-L}, D_1 and W_1 are the (negative) lepton asymmetry number density, decay and washout respectively. This equation accounts for the transitions between the 1, 2 and 3-flavour regimes by promoting the lepton asymmetry number density to a

Model	example input file	Description
1DME	1N3F.dat	DME 1 RHN
2DME	2N3F.dat	DME 2 RHN
3DME	3N3F.dat	DME 3 RHN
1BE1F	1N1F.dat	one-flavoured BE 1 RHN
1BE2F	1N2F.dat	two-flavoured BE 1 RHN
1BE3F	1N3F.dat	three-flavoured BE 1 RHN
2BE1F	2N1F.dat	one-flavoured BE with 2 RHN
2BE2F	2N2F.dat	two-flavoured BE with 2 RHN
2BE3F	2N3F.dat	three-flavoured BE with 2 RHN
3DMEsct	3N3F.dat	DME 3 RHN including scattering effects
1BE1Fsf	1N1F.dat	1BE1F evolving in scale factor
2RES	Res.dat	2BE3F in the resonant regime
2RESsp	Res.dat	2RES including spectator processes

Table 1: Overview of built-in plugins. We abbreviate density matrix equations, Boltzmann equations and (decaying) right-handed neutrino as DME, BE and RHN respectively. The evolution variable is $z = M_1/T$ for all plugins other than 1BE1Fsf which evolves in the cosmological scale factor.

density matrix and adding the appropriate commutators for flavour effects involving the interaction widths, Γ, of the leptons. The initial conditions for RH neutrino and lepton asymmetry number densities are set to zero initial abundance; however, this can be easily modified by the user.

- **2DME** provides the DMEs for the decay of two heavy neutrinos. This is the same as Eq (15) but with subscript 1 replaced with a dummy index i that is summed over two heavy mass states.

- **3DME** provides the DMEs for the decay of three heavy neutrinos.

- **1BE1F** provides the semi-classical BE for one decaying right-handed neutrino, N_1, with number density N_{N_1} in the single flavour approximation. The BE is given by

$$\begin{aligned} \frac{dN_{N_1}}{dz} &= -D_1 \left(N_{N_1} - N_{N_1}^{\rm eq}\right) \\ \frac{dN_{B-L}}{dz} &= \epsilon^{(1)} D_1 \left(N_{N_1} - N_{N_1}^{\rm eq}\right) - W_1 N_{B-L}\,, \end{aligned} \tag{16}$$

This is the simplest possible Boltzmann equation for thermal leptogenesis.

- **1BE2F** provides the semi-classical two-flavoured BE for one decaying right-handed neutrino with flavour effects due to tau leptons. The kinetic equations solved are

$$\begin{aligned}\frac{dN_{N_1}}{dz} &= -D_1\left(N_{N_1} - N_{N_1}^{\rm eq}\right)\\ \frac{dN_{\alpha\alpha}}{dz} &= \sum_{\alpha=\beta\tau}\left(\epsilon^{(1)}_{\alpha\alpha}D_1\left(N_{N_1} - N_{N_1}^{\rm eq}\right) - p_{1\alpha}W_1N_{\alpha\alpha}\right)\,,\end{aligned} \tag{17}$$

  where $p_{1\alpha}$ are projection probabilities between the mass and flavour states. The state β is the coherent e/μ superposition that is left after τ decoheres.

- **1BE3F** provides the semi-classical three-flavoured BE for one decaying right-handed neutrino. This BE is accurate for $M_1 \lesssim 10^9$ GeV and the differential equations solved are

$$\begin{aligned}\frac{dN_{N_1}}{dz} &= -D_1\left(N_{N_1} - N_{N_1}^{\rm eq}\right)\\ \frac{dN_{\alpha\alpha}}{dz} &= \sum_{\alpha=e,\mu,\tau}\left(\epsilon^{(1)}_{\alpha\alpha}D_1\left(N_{N_1} - N_{N_1}^{\rm eq}\right) - p_{1\alpha}W_1N_{\alpha\alpha}\right)\,,\end{aligned} \tag{18}$$

  where $p_{1\alpha}$ are projection probabilities between the mass and flavour states, computed from the $c_{i\alpha}$ elements in lines 14 to 16.

- **2BE1F** provides the semi-classical BE for two decaying right-handed neutrinos in the single flavour approximation. The solved equations are

$$\begin{aligned}\frac{dN_{N_i}}{dz} &= -D_i\left(N_{N_i} - N_{N_i}^{\rm eq}\right)\\ \frac{dN_{B-L}}{dz} &= \sum_{i=1}^{2}\left(\epsilon^{(i)}D_i\left(N_{N_i} - N_{N_i}^{\rm eq}\right) - W_iN_{B-L}\right)\,,\end{aligned} \tag{19}$$

  where $i \in \{1, 2\}$.

- **2BE2F** provides the semi-classical BE for two decaying right-handed neutrinos in the two-flavour approximation.

- **2BE3F** provides the semi-classical three-flavoured BE for two decaying right-handed neutrinos.

- **3DMEsct** provides the three heavy neutrino density matrix equations including $\Delta L = 1$ scattering effects. These are the same as Eq (15) but with three heavy neutrinos and the replacement

$$D_1 \to D_1 + S_1\,, \tag{20}$$

  where S_1 incorporates the effects of $\Delta L = 1$ scatterings involving N_1.

- **1BE1Fsf** is based on the same set of Boltzmann equations as 1BE1F but rather than evolving in $z = M/T$ evolves in the scale factor. This BE is useful if the user wants to implement a non-standard cosmology which modifies the Hubble expansion rate which is given explicitly in this code (for examples see Refs. [21, 22]). We note that in this BE we use the normalisation convention of Section (2.2), namely the particle number density is normalised to a comoving volume which contains a single photon.

- **2RES** provides two heavy neutrino Boltzmann equations for the resonant case. These are the same equations as for 2BE3F, however the CP asymmetries are modified for accuracy in resonant scenarios in which $M_2 - M_1 \sim \Gamma_i$. The modified CP asymmetries used are [20]

$$-\epsilon^{(i)}_{\alpha\alpha} = \sum_{j\neq i} \frac{\mathrm{Im}\left[Y^\dagger_{i\alpha} Y_{\alpha j} \left(Y^\dagger Y\right)_{ij}\right] + \frac{M_i}{M_j}\mathrm{Im}\left[Y^\dagger_{i\alpha} Y_{\alpha j} \left(Y^\dagger Y\right)_{ji}\right]}{\left(Y^\dagger Y\right)_{ii}\left(Y^\dagger Y\right)_{jj}} \left(f^{\mathrm{mix}}_{ij} + f^{\mathrm{osc}}_{ij}\right),$$
$$f^{\mathrm{mix}}_{ij} = \frac{\left(M_i^2 - M_j^2\right) M_i \Gamma_j}{\left(M_i^2 - M_j^2\right)^2 + M_i^2 \Gamma_j^2}, \quad f^{\mathrm{osc}}_{ij} = \frac{\left(M_i^2 - M_j^2\right) M_i \Gamma_j}{\left(M_i^2 - M_j^2\right)^2 + \left(M_i\Gamma_i + M_j\Gamma_j\right)^2 \frac{\det\left[\mathrm{Re}\left(Y^\dagger Y\right)\right]}{\left(Y^\dagger Y\right)_{ii}\left(Y^\dagger Y\right)_{jj}}}. \tag{21}$$

- **2RESsp** provides the equations for resonant leptogenesis with the lowest temperature scale spectator effects included through the factors C^Φ and C^l [23] by promoting the washout terms to

$$-p_{1\alpha} W_1 \sum_\beta \left(C^l_{\alpha\beta} + C^\Phi_\beta\right) N_{\beta\beta}\,. \tag{22}$$

  The current implementation includes spectator effects accurate for $T \ll 10^8$ GeV.

4. Installation

The code is hosted on `https://github.com/earlyuniverse/ulysses`. Once the git repository is pulled, the basic installation steps are shown in Listing (1). In addition, releases are packaged and available to install with PIP from `pypi.org`.

Listing 1: Minimal installation steps.

```
# Installation from within the source tree
git clone https://github.com/earlyuniverse/ulysses.git
cd ulysses
pip install . --user

# Installation with pip or pip3 from pypi.org
pip install ulysses --user
```

4.1. Core dependencies

The code is written in python3 and heavily uses the widely available modules NumPy [24, 25] and SciPy [26]. We accelerate the computation with the just in time compiler provided by Numba [27] where meaningful. At its core, ULYSSES solves a set of coupled differential equations. To undertake this task we use odeintw [28] which provides a wrapper of scipy.integrate.odeint that allows it to handle complex and matrix differential equations. The latter is redistributed with ULYSSES and does not need to be downloaded separately. These dependencies for ULYSSES are automatically resolved during the install process with pip. They provide the minimal functionality for solving Boltzmann equations for a given point in the model parameter space.

4.2. Additional requirements for multidimensional scans

Listing 2: Installation of libMultiNest

```
git clone https://github.com/JohannesBuchner/MultiNest
cd MultiNest/build
cmake ..
make
cd ..
export LD_LIBRARY_PATH=$PWD/lib:$LD_LIBRARY_PATH
```

For multidimensional parameter space exploration with the aim of finding regions compatible with the experimentally measured η_B we provide a script, uls-nest, which invokes MultiNest [29, 30, 4]. MultiNest efficiently scans a parameter space to find regions of max-

imum likelihood. ULS-NEST implements a simple log-likelihood for that purpose:

$$\log \mathcal{L}(x|\vec{p}) = -0.5 \cdot \left(\frac{\eta_B(\vec{p}) - x}{\Delta x} \right)^2 , \tag{23}$$

where $x \pm \Delta x$ are the experimentally measured values $\eta_{B\,\mathrm{CMB}} = (6.10 \pm 0.04) \times 10^{-10}$. We denote the baryon asymmetry parameter as calculated by ULYSSES for a point $\vec{p}$ of the currently loaded model as $\eta_B(\vec{p})$.

MultiNest is a code written in C and FORTRAN that can optionally be compiled with support for message passing interface (MPI) to enable parallel computing on a single workstation or potentially many network connected computers. The usage of MultiNest in python is made possible with the additional pip installable package PYMULTINEST (https://github.com/JohannesBuchner/MultiNest). PYMULTINEST requires a shared library of MultiNest to be be available in the users environment. MPI parallelism is available through MPI4PY [31, 32, 33]. It should be noted that MPI4PY and PYMULTINEST are automatically installed when using pip to install ULYSSES. The compilation of the MultiNest library cannot be automated in that fashion. An example of how to obtain the source code and how to compile the shared library using CMAKE is given in Listing (2). Furthermore, CMAKE detects if MPI is available on the system and triggers the compilation of the library LIBMULTINEST_MPI in addition to the serial LIBMULTINEST.

5. Computing model

We designed ULYSSES to be easily extensible in such a way that users can focus on the physics. The module contains a single base class, ULSBase, which has all the infrastructure needed to solve the problem at hand. This includes machineries to set global constants, parameters of the physics models and the ODE solver as well as commonly used computations, such as the calculation of the PMNS matrix in the Casas-Ibarra parametrisation. The base class itself is devoid of any concrete physics but contains a dummy function, `EtaB`, which must be overwritten in classes which are derived from ULSBase that implement the actual Boltzmann equations. We further provide a plugin mechanism that allows the seamless usage of user developed models with the run-time scripts of ULYSSES — as long as the new model also derives from `ULSBase` and implements its own version of `EtaB`. An example of the code structure can be seen in Listing (3).

Listing 3: A skeleton for an externally provided plugin model for the calculation of η_B

```
# Content of myplugin.py
import ulysses
class EtaB_plugin(ulysses.ULSBase):
    """
    My new plugin
    """
    def RHS(self):
        """
        Right hand side of ODE system goes here
        """
        rhs = ...
        return rhs

    @property
    def EtaB(self):
        """
        Invoke e.g. odeintw, calculate and return etab.
        """
        y0      = np.array([0+0j,0+0j], dtype=np.complex128)
        ys, _   = odeintw(self.RHS, y0, self.zs)
        nb      = self.normfact*(ys[-1,1]+ys[-1,2]+ys[-1,3])
        return np.real(nb)
```

Parameter	variable name	default	unit
Higgs VEV, v	`vev`	174.0	[GeV]
Higgs mass, M_H	`mhiggs`	125.0	[GeV]
Z boson mass, M_Z	`mz`	91.1876	[GeV]
Planck mass, $M_{\rm PL}$	`mplanck`	1.22×10^{19}	[GeV]
Neutrino cosmological mass, m^*	`mstar`	10^{-12}	[GeV]
Degrees of freedom, $g*$	`gstar`	106.75	
Normalisation factor	`normfact`	0.013	
Solar mass square splitting, $m^2_{\rm SOL}$	`m2solar`	7.4×10^{-23}	[GeV^2]
Atm. mass squared splitting (normal), $m^2_{\rm ATM}$	`m2atm`	2.515×10^{-21}	[GeV^2]
Atm. mass squared splitting (inverted), $m^2_{\rm ATM,inv}$	`m2atminv`	2.483×10^{-21}	[GeV^2]

Table 2: Overview of global parameters and their defaults values.

5.1. Setting parameters

Listing 4: Example input for uls-calc.

```
m       -1.10
M1      12.10
M2      12.60
M3      13.00
delta   213.70
a21     81.60
a31     476.70
x1      90.00
x2      87.00
x3      180.00
y1      -120.00
y2      0.00
y3      -120.00
t12     33.63
t13     8.52
t23     49.58
```

Listing 5: Example input for uls-scan.

```
m       -1.10
M1      6.00    12.00
M2      12.60
M3      13.00
delta   213.70
a21     81.60
a31     476.70
x1      90.00
x2      87.00
x3      180.00
y1      -120.00
y2      0.00
y3      -120.00
t12     33.63
t13     8.52
t23     49.58
```

Listing 6: Example parameter card for uls-nest.

```
m       -4.00   -1.00
M1      6.00
M2      7.00
M3      7.50
delta   0.00    360.00
a21     0.00    720.00
a31     0.00    720.00
x1      0.00    180.00
x2      0.00    180.00
x3      0.00    180.00
y1      -180.00 180.00
y2      -180.00 180.00
y3      -180.00 180.00
t12     33.63
t13     8.52
t23     49.58
```

Listing 7: Example input for uls-calc.

```
Y11_mag   0.01
Y12_mag   0.01
Y13_mag   0.01
Y21_mag   0.01
Y22_mag   0.03
Y23_mag   0.05
Y31_mag   0.01
Y32_mag   0.03
Y33_mag   0.05
Y11_phs  -1.11
Y12_phs   2.89
Y13_phs   1.32
Y21_phs   2.88
Y22_phs  -0.23
Y23_phs  -1.80
Y31_phs  -1.72
Y32_phs   2.96
Y33_phs   1.39
M1        12.0
M2        12.5
M3        13.0
```

Listing 8: Example input for uls-scan.

```
Y11_mag   0.01
Y12_mag   0.01
Y13_mag   0.01
Y21_mag   0.01
Y22_mag   0.03
Y23_mag   0.05
Y31_mag   0.01
Y32_mag   0.03
Y33_mag   0.05
Y11_phs   0.00 3.14
Y12_phs   2.89
Y13_phs   1.32
Y21_phs   2.88
Y22_phs  -0.23
Y23_phs  -1.80
Y31_phs  -1.72
Y32_phs   2.96
Y33_phs   1.39
M1       12.00
M2       12.50
M3       13.00
```

Listing 9: Example parameter card for uls-nest.

```
Y11_mag   0.01
Y12_mag   0.01
Y13_mag   0.01
Y21_mag   0.01
Y22_mag   0.03
Y23_mag   0.05
Y31_mag   0.01
Y32_mag   0.03
Y33_mag   0.05
Y11_phs  -3.14 3.14
Y12_phs  -3.14 3.14
Y13_phs  -3.14 3.14
Y21_phs  -3.14 3.14
Y22_phs  -3.14 3.14
Y23_phs  -3.14 3.14
Y31_phs  -3.14 3.14
Y32_phs  -3.14 3.14
Y33_phs  -3.14 3.14
M1        12.0
M2        12.5
M3        13.0
```

All global constants are defined in the __init__ function of the base class. We allow the user to set their values per the standard python keyword argument formalism using the variable names shown in the second column of Table (2). The required input from the user is the model parameters which derive from the Casas-Ibarra parametrisation of the Yukawa matrix, Y, as shown in Eq (3). The parameters which may be explored by the user are shown in Table (3). The lightest neutrino mass (m) is fixed by the user and the two heavier neutrino masses are fixed at the best-fit values from global fit data [34] which can be changed in ulsbase.py. In the example shown in Listing (4), the lightest active neutrino mass is $m_1 = 10^{-1.1}$ eV and the right-handed neutrino masses are $N_{1,2,3} = 10^{12.1,12.6,13}$ GeV respectively. We note that the masses of both the light and heavy neutrinos are provided by the exponent to base 10.

Parameter	Unit	Code input example	
δ	[°]	delta	270
α_{21}	[°]	a21	0
α_{31}	[°]	a31	0
θ_{23}	[°]	t23	48.7
θ_{12}	[°]	t12	33.63
θ_{13}	[°]	t13	8.52
x_1	[°]	x1	45
y_1	[°]	y1	45
x	[°]	x2	45
y_2	[°]	y2	45
x	[°]	x3	45
y_3	[°]	y3	45
$\log_{10}\left(m_{1/3}\right)$	[eV]	m	-0.606206
$\log_{10}\left(M_1\right)$	[GeV]	M1	11
$\log_{10}\left(M_2\right)$	[GeV]	M2	12
$\log_{10}\left(M_3\right)$	[GeV]	M3	15

Table 3: Overview of input parameters in the Casas-Ibarra parametrisation.

As discussed before, the method of Casas and Ibarra is one popular way of parametrising the Yukawa matrix. However, ULYSSES also allows the user to provide their own Yukawa matrix, in polar coordinates, and calculate the resultant baryon asymmetry. We note that the user will need to independently ensure that oscillation data is satisfied. The input logic is such that each element of the Yukawa matrix, Y_{ij}, is determined by two independent parameters `Yij_mag` and `Yij_phs`:

$$Y_{ij} = \texttt{Yij\_mag} \cdot \exp\left(i\ \texttt{Yij\_phs}\right) \tag{24}$$

An example parameter card is shown in Listing (7).

6. Run time scripts and examples

To display the preprovided BEs, as detailed in Section (3), and the strings needed to load them from the command line the user can call:

```
# display list of available models
```

The output is similar to Table (1); the shorthand for the models will be printed to screen in the leftmost column.

For convenience, we ship three runtime scripts which use the ULYSSES module for the evaluation of η_B at a single point as well as in one-dimensional and in multi-dimensional parameter space explorations:

- `uls-calc`
- `uls-scan`
- `uls-nest`

which are discussed in Sections (6.2-6.4) respectively. The only mandatory argument to all of these scripts is a parameter input card. We allow the user to apply the Casas-Ibarra parametrisation as well as specifying the Yukawa matrix explicitly. The former has a total of 16 free parameters, while the latter has 21. The structure of the parameter files is slightly different for each script and is explained below. It should be noted that we decided against setting the physics parameters to default values. This means that in all scripts, the full set of 16 (21) input parameters must be provided. The physics and computational setup can further be steered with a set of command line options and switches.

6.1. Common options

We first describe the command line options that are common to all three scripts. All scripts allow the user to set the global constants given in Table (2) on the command line. The syntax is always `key:value`. For example, to set the normalisation factor to 0.015, the user would input to the command line:

```
# Use one of the built—in plugin
uls—calc —m 1DME examples/1N3F.dat  normfact:0.015
```

Boltzmann equation selection, -m The command line argument "-m" is used to select a Boltzmann equation. For the built-in BEs this can be any string as given in Table (1). For the plugin system the syntax is slightly different. The absolute or relative path to the file containing the plugin implementation needs to be specified, together with the name of the class. Both are separated by a colon:

```
# Use one of the built-in plugins
uls-calc -m 1DME examples/1N3F.dat

# Use an externally provided plugins
uls-calc -m myplugin.py:EtaB_plugin  examples/1N3F.dat
```

Inverted mass ordering, loop corrections By default, the normal mass ordering is applied in the calculations. To explore the parameter space in the context of an inverted mass ordering, the command line switch "–inv" must be added. Similarly, to implement loop corrections which by default are off, as detailed in Section (2.1), can be enabled by adding the switch "–loop" to the command line.

```
uls-calc -m 1DME examples/1N3F.dat --inv
uls-calc -m 1DME examples/1N3F.dat --loop
uls-calc -m 1DME examples/1N3F.dat --inv --loop
```

Integration range, --zrange To set up the integration range and steps, we use the following syntax:

```
uls-calc -m 1DME examples/1N3F.dat --zrange 0.1,50,300
```

This example sets the integration range to be between 0.1 and 50, using 300 steps as opposed to the default of 1000 steps between 0.1 and 1000.

6.2. uls-calc

This code calculates and prints the baryon asymmetry parameter for a given point and selected BE:

```
uls-calc -m 3DME  point.txt
```

The required positional argument is the parameter point in question in a simple text file with parameter name value pairs. An example parameter card is given in Listing (4) for the Casas-Ibarra parametrisation and the free format in Listing (7). For convenience, we provide the functionality to write out the evolution of the lepton asymmetry number densities if the command line option "-o" is provided. Depending on the ending of the file

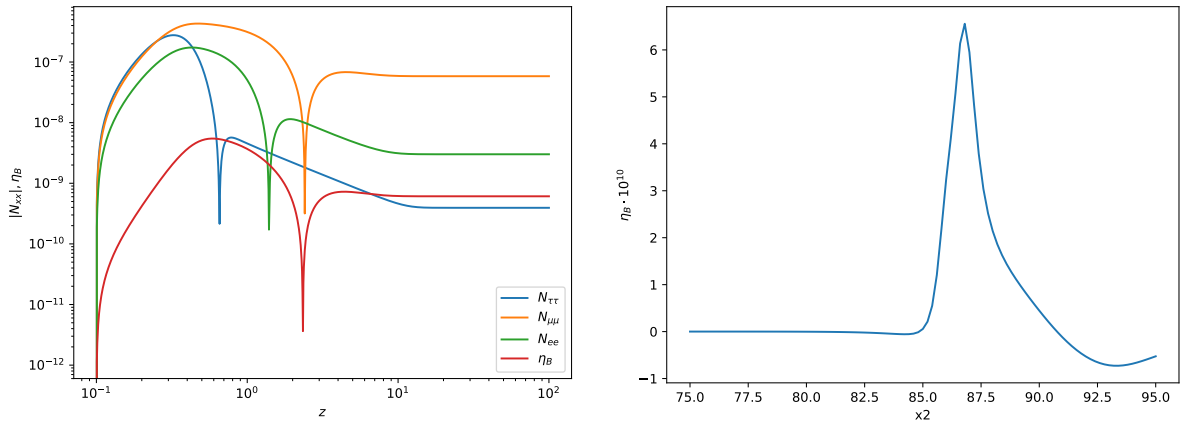

Figure 6.1: Example output of uls-calc (left) and uls-scan (right).

name this is either in the form of a plot (see left plot of Fig (6.1)) or as an array of numbers stored in a text file.

```
# Produce a plot of the evolution
uls-calc -m 3DME  examples/3N3NF.dat -o evolution.pdf

# Write evolution data to a text file
uls-calc -m 3DME  examples/3N3NF.dat -o evolution.txt
```

6.3. uls-scan

To perform a one-dimensional scan of η_B for a certain model, uls-scan can be used. We again use the command line option "-o" to specify the output file name. An example plot of the output from uls-scan is shown in the right plot of Fig (6.1). The range of the parameter to be scanned is taken from the input file (see Listing (5) and (8) for an example). The number of points to run the scan for can be selected with "-n":

```
uls-scan -m 3DME  scan_x2.txt -o scan_x2.pdf  -n 40
```

6.4. uls-nest

uls-nest is a multidimensional likelihood sampler using MultiNest.

The output of uls-nest is the standard output of MultiNest which is a text file that contains the sampled points and corresponding likelihood and posterior values. Visualisation of

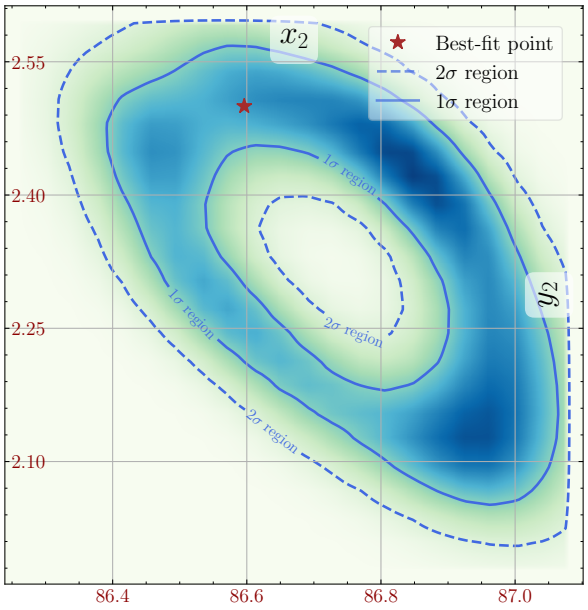

Figure 6.2: Visualisation of uls-nest output with SuperPlot.

the output can, for instance, be undertaken with SuperPlot [35] (see Fig (6.2)) or the plotting tools that are provided by PYMULTINEST. The parameter space to scan can be defined by supplying a simple text file with key value pairs. We use the following logic: A parameter name followed by two numbers is interpreted as boundaries on that particular parameter's subspace while a single number is interpreted as fixing the corresponding parameter to the supplied value. An example can be found in Listing (6) and (9).

The command line to run the code on a single CPU may look like this:

```
# Single core run
uls-nest -m 3DME scan_x2_y2.ranges -o 2Dscan
```

As the computational cost increases with the number of free parameters in the scan the run-time may become quite large. If MultiNest is compiled with MPI enabled and `mpi4py` is installed, uls-nest can be executed in parallel. We note that the parallel computation is already beneficial on a workstation or laptop.

```
# The same physics setup but distributed over 256 CPUs
mpiexec -np 256 uls-nest -m 3DME scan_x2_y2.ranges -o 2Dscan
```

Option	Default	Parameter name in pymultinest
`--mn-points`	400	`n_live_points`
`--mn-tol`	0.5	`evidence_tolerance`
`--mn-eff`	0.8	`sampling_efficiency`
`--mn-imax`	0	`max_iter`
`--mn-resume`	False	`resume`
`--mn-multimodal`	False	`multimodal`
`--mn-no-importance`	False	`not importance_nested_sampling`
`--mn-seed`	-1	`seed`
`--mn-update`	1000	`n_iter_before_update`

Table 4: MultiNest specific parameters and their defaults available in ULYSSES. The third column identifies the parameter name as used in pymultinest.

MultiNest parameters We provide access to all commonly used MultiNest parameters through command line options. To separate them from the rest of the options, we use the pattern `--mn-OPTION`. Table (4) gives an overview of various switches and their defaults. For a thorough discussion of their meaning we direct the reader to the official documentation at https://johannesbuchner.github.io/PyMultiNest/

7. Summary and Discussion

ULYSSES is the first publicly available code to calculate the baryon asymmetry in the framework of a type-I seesaw mechanism. Currently the code provides momentum-averaged Boltzmann equations for the out-of-equilibrium decays and resonant leptogenesis with examples on how to incorporate lepton flavour, scatterings and spectator effects. The ULYSSES code structure also allows the user to calculate the baryon asymmetry from their own externally defined plugin. Additional effects, which would refine the baryon asymmetry calculation, are of interest for future code development. These include thermal production rates at finite temperature [36, 37], next-to-leading-order corrections for the source term [38, 39, 40] and inclusion of partially equilibrated spectator processes [41, 42]. Furthermore, inclusion of a plugin for leptogenesis via oscillation [43] is of interest given its close connection with a number of experimental probes. Finally, we view this as a community project and invite users to add their own plugins to share with others. This is implemented via issues and pull requests on our GitHub repository.

Acknowledgements

We are deeply grateful to Serguey T. Petcov for useful discussions and suggestions. It is a pleasure to thank Marco Drewes for helpful discussions on this code. This research was supported by the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, HEP User Facility. K.M. acknowledges the (partial) support from the European Research Council under the European Union Seventh Framework Programme (FP/2007-2013) / ERC Grant NuMass agreement n. [617143]. Fermilab is managed by Fermi Research Alliance, LLC (FRA), acting under Contract No. DE–AC02–07CH11359. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program, grant HEP Data Analytics on HPC, No. 1013935. It was supported by the U.S. Department of Energy under contracts DE-AC02-76SF00515.

[1] M. Fukugita and T. Yanagida. Baryogenesis Without Grand Unification. *Phys. Lett.*, B174:45–47, 1986.

[2] M. E. Shaposhnikov. Baryon Asymmetry of the Universe in Standard Electroweak Theory. *Nucl. Phys.*, B287:757–775, 1987.

[3] Andrew G. Cohen, David B. Kaplan, and Ann E. Nelson. Baryogenesis at the weak phase transition. *Nucl. Phys.*, B349:727–742, 1991.

[4] F. Feroz, M. P. Hobson, and M. Bridges. MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Mon. Not. Roy. Astron. Soc.*, 398:1601–1614, 2009.

[5] C. Patrignani et al. Review of Particle Physics. *Chin. Phys.*, C40(10):100001, 2016.

[6] P. A. R. Ade et al. Planck 2015 results. XIII. Cosmological parameters. *Astron. Astrophys.*, 594:A13, 2016.

[7] C. Hagedorn, R. N. Mohapatra, E. Molinaro, C. C. Nishi, and S. T. Petcov. CP Violation in the Lepton Sector and Implications for Leptogenesis. *Int. J. Mod. Phys.*, A33(05n06):1842006, 2018.

[8] P. S. Bhupal Dev, Pasquale Di Bari, Bjorn Garbrecht, Stephane Lavignac, Peter Millington, and Daniele Teresi. Flavor effects in leptogenesis. *Int. J. Mod. Phys.*, A33:1842001, 2018.

[9] Bhupal Dev, Mathias Garny, Juraj Klaric, Peter Millington, and Daniele Teresi. Resonant enhancement in leptogenesis. *Int. J. Mod. Phys.*, A33:1842003, 2018.

[10] Peter Minkowski. $\mu \to e\gamma$ at a Rate of One Out of 10^9 Muon Decays? *Phys. Lett.*, B67:421–428, 1977.

[11] Tsutomu Yanagida. HORIZONTAL SYMMETRY AND MASSES OF NEUTRINOS. *Conf. Proc.*, C7902131:95–99, 1979.

[12] Murray Gell-Mann, Pierre Ramond, and Richard Slansky. Complex Spinors and Unified Theories. *Conf. Proc.*, C790927:315–321, 1979.

[13] J. A. Casas and A. Ibarra. Oscillating neutrinos and muon —¿ e, gamma. *Nucl. Phys.*, B618:171–204, 2001.

[14] J. Lopez-Pavon, S. Pascoli, and Chan-fai Wong. Can heavy neutrinos dominate neutrinoless double beta decay? *Phys. Rev.*, D87(9):093007, 2013.

[15] K. Moffat, S. Pascoli, S. T. Petcov, H. Schulz, and J. Turner. Three-flavored nonresonant leptogenesis at intermediate scales. *Phys. Rev.*, D98(1):015036, 2018.

[16] J. Lopez-Pavon, E. Molinaro, and S. T. Petcov. Radiative Corrections to Light Neutrino Masses in Low Scale Type I Seesaw Scenarios and Neutrinoless Double Beta Decay. *JHEP*, 11:030, 2015.

[17] W. Buchmuller, P. Di Bari, and M. Plumacher. Leptogenesis for pedestrians. *Annals Phys.*, 315:305–351, 2005.

[18] Luca Marzola. *On leptogenesis, flavour effects and the low energy neutrino parameters*. PhD thesis, Southampton U., 2012.

[19] Steve Blanchet, Pasquale Di Bari, David A. Jones, and Luca Marzola. Leptogenesis with heavy neutrino flavours: from density matrix to Boltzmann equations. *JCAP*, 1301:041, 2013.

[20] Andrea De Simone and Antonio Riotto. On Resonant Leptogenesis. *JCAP*, 0708:013, 2007.

[21] Bhaskar Dutta, Chee Sheng Fong, Esteban Jimenez, and Enrico Nardi. A cosmological pathway to testable leptogenesis. *JCAP*, 1810:025, 2018.

[22] W. Buchmuller, K. Schmitz, and G. Vertongen. Entropy, Baryon Asymmetry and Dark Matter from Heavy Neutrino Decays. *Nucl. Phys.*, B851:481–532, 2011.

[23] Enrico Nardi, Yosef Nir, Esteban Roulet, and Juan Racker. The Importance of flavor in leptogenesis. *JHEP*, 01:164, 2006.

[24] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[25] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, 2011.

[26] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[27] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM 15, New York, NY, USA, 2015. Association for Computing Machinery.

[28] Warren Weckesser. odeintw: Complex and matrix differential equations. https://github.com/WarrenWeckesser/odeintw, 2014.

[29] J. Buchner, A. Georgakakis, K. Nandra, L. Hsu, C. Rangel, M. Brightman, A. Merloni, M. Salvato, J. Donley, and D. Kocevski. X-ray spectral modelling of the AGN obscuring region in the CDFS: Bayesian model selection and catalogue. *aap*, 564:A125, April 2014.

[30] F. Feroz, M. P. Hobson, E. Cameron, and A. N. Pettitt. Importance Nested Sampling and the MultiNest Algorithm. 2013.

[31] Lisandro Dalcn, Rodrigo Paz, and Mario Storti. Mpi for python. *Journal of Parallel and Distributed Computing*, 65(9):1108 – 1115, 2005.

[32] Lisandro Dalcn, Rodrigo Paz, Mario Storti, and Jorge DEla. Mpi for python: Performance improvements and mpi-2 extensions. *Journal of Parallel and Distributed Computing*, 68(5):655 – 662, 2008.

[33] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124 – 1139, 2011. New Computational Methods and Software Tools.

[34] Ivan Esteban, M. C. Gonzalez-Garcia, Alvaro Hernandez-Cabezudo, Michele Maltoni, and Thomas

Schwetz. Global analysis of three-flavour neutrino oscillations: synergies and tensions in the determination of $\theta_2 3, \delta_C P$, and the mass ordering. *JHEP*, 01:106, 2019.

[35] Andrew Fowlie and Michael Hugh Bardsley. Superplot: a graphical interface for plotting and analysing MultiNest output. *Eur. Phys. J. Plus*, 131(11):391, 2016.

[36] Bjorn Garbrecht, Frank Glowna, and Matti Herranen. Right-Handed Neutrino Production at Finite Temperature: Radiative Corrections, Soft and Collinear Divergences. *JHEP*, 04:099, 2013.

[37] I. Ghisoiu and M. Laine. Right-handed neutrino production rate at T ¿ 160 GeV. *JCAP*, 1412:032, 2014.

[38] Dietrich Bodeker and Marc Sangel. Lepton asymmetry rate from quantum field theory: NLO in the hierarchical limit. *JCAP*, 1706:052, 2017.

[39] Simone Biondini, Nora Brambilla, and Antonio Vairo. CP asymmetry in heavy Majorana neutrino decays at finite temperature: the hierarchical case. *JHEP*, 09:126, 2016.

[40] Simone Biondini, Nora Brambilla, Miguel Angel Escobedo, and Antonio Vairo. CP asymmetry in heavy Majorana neutrino decays at finite temperature: the nearly degenerate case. *JHEP*, 03:191, 2016. [Erratum: JHEP08,072(2016)].

[41] Bjorn Garbrecht and Pedro Schwaller. Spectator Effects during Leptogenesis in the Strong Washout Regime. *JCAP*, 1410:012, 2014.

[42] Bjorn Garbrecht, Philipp Klose, and Carlos Tamarit. Relativistic and spectator effects in leptogenesis with heavy sterile neutrinos. *JHEP*, 02:117, 2020. [JHEP20,117(2020)].

[43] Evgeny K. Akhmedov, V. A. Rubakov, and A. Yu. Smirnov. Baryogenesis via neutrino oscillations. *Phys. Rev. Lett.*, 81:1359–1362, 1998.