FortEPiaNO technical notes

S. Gariazzo,[1,*] P.F. de Salas,[2,†] and S. Pastor[1,‡]

[1] *Instituto de Física Corpuscular (CSIC-Universitat de València), Valencia, Spain*
[2] *The Oskar Klein Centre for Cosmoparticle Physics, Department of Physics, Stockholm University, SE-106 91 Stockholm, Sweden*

We present here the main features of our code, `FORTran-Evolved PrimordIAl Neutrino Oscillations` (`FortEPiaNO`) [1]. The code is publicly available at the url `https://bitbucket.org/ahep_cosmo/fortepiano_public`.

I. EQUATIONS

`FortEPiaNO` can compute oscillations with up to six neutrinos in the early universe. Neutrinos, including the sterile ones, are always treated as ultra-relativistic particles, which is a good approximation if the neutrino masses do not exceed $\mathcal{O}$(a few keV), i.e. neutrinos are still fully relativistic at decoupling. For larger masses, neutrinos may start to become non-relativistic before decoupling, and in that case one should take into account the effect of the mass.

The code computes the evolution of the $N \times N$ neutrino density matrix [2–5]

$$\varrho(x,y) = \begin{pmatrix} \varrho_{ee} & \varrho_{e\mu} & \varrho_{e\tau} & \varrho_{es_1} & \cdots \\ \varrho_{\mu e} & \varrho_{\mu\mu} & \varrho_{\mu\tau} & \varrho_{\mu s_1} & \\ \varrho_{\tau e} & \varrho_{\tau\mu} & \varrho_{\tau\tau} & \varrho_{\tau s_1} & \\ \varrho_{s_1 e} & \varrho_{s_1\mu} & \varrho_{s_1\tau} & \varrho_{s_1 s_1} & \\ \vdots & & & & \ddots \end{pmatrix}, \tag{1}$$

which is the same for neutrinos and antineutrinos, in terms of the comoving coordinates $x \equiv m_e\, a$, $y \equiv p\, a$, $z \equiv T_\gamma\, a$ and $w \equiv T_\nu\, a$. The momentum dependence of the density matrix ϱ is taken into account using a discrete grid of momenta, as described in section V.

When using N neutrinos, the mixing matrix is defined as

$$U = R^{(N-1)N} \ldots R^{1N} R^{(N-2)(N-1)} \ldots R^{1(N-1)} \ldots R^{34} R^{24} R^{14} R^{23} R^{13} R^{12}, \tag{2}$$

following and extending the convention presented in Eq. (12) of [6], where each R^{ij} is a real rotation matrix described by the angle θ_{ij}, containing $\cos\theta_{ij}$ in the diagonal elements ii and jj, 1 in the remaining diagonal elements, $\sin\theta_{ij}$ ($-\sin\theta_{ij}$) in the off-diagonal element ij (ji) and zero otherwise:

$$[R^{ij}]_{rs} = \delta_{rs} + (\cos\theta_{ij} - 1)(\delta_{ri}\delta_{si} + \delta_{rj}\delta_{sj}) + \sin\theta_{ij}(\delta_{ri}\delta_{sj} - \delta_{rj}\delta_{si})\,. \tag{3}$$

It enters the calculation of the rotated mass matrix $\mathbb{M}_\mathrm{F} = U \mathbb{M} U^\dagger$, where the diagonal mass matrix is $\mathbb{M} = \mathrm{diag}(m_1^2, \ldots, m_N^2)$. Other matrices that we need to define are

$$\mathbb{E}_\ell = \mathrm{diag}(\rho_e, \rho_\mu, 0, \ldots)\,, \qquad \mathbb{P}_\ell = \mathrm{diag}(P_e, P_\mu, 0, \ldots)\,, \qquad \mathbb{E}_\nu = S_a \frac{1}{\pi^2}\left(\int \mathrm{d}y y^3 \varrho\right) S_a \quad \text{with } S_a = \mathrm{diag}(1,1,1,0,\ldots)\,, \tag{4}$$

while the interaction matrices used in the the collision terms, presented in section II, are

$$G^L = \mathrm{diag}(g_L, \tilde{g}_L, \tilde{g}_L, 0, \ldots)\,, \qquad G^R = \mathrm{diag}(g_R, g_R, g_R, 0, \ldots)\,, \tag{5}$$

where $g_L = \sin^2\theta_W + 1/2$, $\tilde{g}_L = \sin^2\theta_W - 1/2$, $g_R = \sin^2\theta_W$, and θ_W is the weak mixing angle. It is easy to see from the definitions of Eqs. (57) and (58) that the collision terms vanish when considering the interactions corresponding only to sterile neutrinos. When more than one sterile neutrino is considered, the damping terms between the different sterile neutrinos are therefore set to zero.

* gariazzo@ific.uv.es
† pablo.fernandez@fysik.su.se
‡ pastor@ific.uv.es

The definitions of comoving energy density and pressure, which can be combined to obtain the comoving entropy density, are written as:

$$\rho_i = g_i \int \frac{dp}{2\pi^2}\, p^2 E_p \frac{1}{e^{E_p/z} \pm 1}\,, \tag{6}$$

$$P = g_i \int \frac{dp}{2\pi^2}\, \frac{p^4}{3E_p} \frac{1}{e^{E_p/z} \pm 1}\,, \tag{7}$$

$$s = \frac{\rho + P}{z}\,, \tag{8}$$

where the + (-) applies for fermions (bosons), i denotes the species, which has g_i degrees of freedoms, p is the comoving momentum, $E_p = \sqrt{p^2 + m_i^2}$, being m_i the comoving mass of the particle, and z must be substituted with w in the case of neutrinos.

To take into account finite temperature QED (FTQED) corrections [5, 7, 8], we have to modify the total pressure and energy density of the fluid:

$$P = \sum_{i=\gamma,\nu_i,e,\mu} P_i + \delta P(x,z)\,, \tag{9}$$

$$\rho = \sum_{i=\gamma,\nu_i,e,\mu} \rho_i + \delta \rho(x,z)\,, \tag{10}$$

where δP and $\delta\rho$ are contributions that can be computed using FTQED. It is convenient to define them in terms of the following functions:

$$J_a(r) = \frac{1}{\pi^2}\int_0^\infty \mathrm{d}u\, u^a \frac{\exp(\sqrt{u^2+r^2})}{\left[\exp(\sqrt{u^2+r^2})+1\right]^2}\,, \tag{11}$$

$$K_a(r) = \frac{1}{\pi^2}\int_0^\infty \mathrm{d}u\, \frac{u^a}{\sqrt{u^2+r^2}} \frac{1}{\exp(\sqrt{u^2+r^2})+1}\,. \tag{12}$$

Let us also define for convenience:

$$\mathcal{N}_p = \frac{2}{e^{E_p/z}+1}\,, \tag{13}$$

$$\partial_x \mathcal{N}_p = -\frac{x\, e^{E_p/z}\, \mathcal{N}_p^2}{2z\, E_p}\,, \tag{14}$$

$$\partial_z \mathcal{N}_p = \frac{e^{E_p/z}\, E_p\, \mathcal{N}_p^2}{2z^2}\,, \tag{15}$$

$$\partial_x \partial_z \mathcal{N}_p = \frac{x\, e^{E_p/z}\, \mathcal{N}_p^2}{2z^3}\left(1 - e^{E_p/z}\,\mathcal{N}_p + \frac{z}{E_p}\right)\,. \tag{16}$$

The contributions δP and $\delta\rho$ can be expanded as a series of powers of the electron charge $e^2 = 4\pi\alpha$, where α is the fine structure constant. Taking into account the first orders of the expansion, and using $r = x/z$, for the pressure one has [8]

$$\delta P(x,z) = \delta P^{(2)}(x/z) + \delta P^{(2+\ln)}(x,z) + \delta P^{(3)}(x/z) + \dots\,, \tag{17}$$

$$\delta P^{(2)}(r) = -e^2 z^4\, K_2 \left(\frac{1}{6} + \frac{K_2}{2}\right)\,, \tag{18}$$

$$\delta P^{(2+\ln)}(x,z) = \frac{e^2 x^2}{16\pi^4} \iint_0^\infty \mathrm{d}y\, \mathrm{d}k\, \frac{y\,k}{E_y E_k} \ln\left|\frac{y+k}{y-k}\right| \mathcal{N}_y\, \mathcal{N}_k\,, \tag{19}$$

$$\delta P^{(3)}(r) = \frac{2e^3 z^4}{3\pi}\left(K_2 + \frac{r^2}{2} K_0\right)^{3/2}\,, \tag{20}$$

while for the energy density the various terms are

$$\delta\rho(x,z) = \delta\rho^{(2)}(x/z) + \delta\rho^{(2+\ln)}(x,z) + \delta\rho^{(3)}(x/z) + \dots\,, \tag{21}$$

$$\delta\rho^{(2)}(r) = e^2 z^4 \left(\frac{K_2^2}{2} - \frac{K_2+J_2}{6} - K_2 J_2\right)\,, \tag{22}$$

$$\delta\rho^{(2+\ln)}(x,z) = \frac{e^2x^2}{16\pi^4}\iint_0^\infty \mathrm{d}y\,\mathrm{d}k\,\frac{y\,k}{E_y E_k}\ln\left|\frac{y+k}{y-k}\right| \mathcal{N}_y\Big(2z\partial_z\mathcal{N}_k - \mathcal{N}_k\Big)\,, \tag{23}$$

$$\delta\rho^{(3)}(r) = \frac{e^3 z^4}{\pi}\left(K_2 + \frac{r^2}{2}K_0\right)^{1/2}\left(J_2 + \frac{r^2}{2}J_0\right)\,. \tag{24}$$

In order to compute the evolution of the neutrino density matrix (1), we have to compute both the derivative of ϱ (as a function of the momentum) and of the comoving photon temperature z with respect to our time parameter x. The differential equations which the code solves are the following [2–5]:

$$\frac{\mathrm{d}\varrho(y)}{\mathrm{d}x} = \sqrt{\frac{3m_{\mathrm{Pl}}^2}{8\pi\rho}}\left\{-i\frac{x^2}{m_e^3}\left[\frac{\mathbb{M}_{\mathrm{F}}}{2y} - \frac{2\sqrt{2}G_{\mathrm{F}} y m_e^6}{x^6}\left(\frac{\mathbb{E}_\ell+\mathbb{P}_\ell}{m_W^2} + \frac{4}{3}\frac{\mathbb{E}_\nu}{m_Z^2}\right), \varrho\right] + \frac{m_e^3}{x^4}\mathcal{I}(\varrho)\right\}\,, \tag{25}$$

$$\frac{\mathrm{d}z}{\mathrm{d}x} = \frac{\displaystyle\sum_{\ell=e,\mu}\left[\frac{r_\ell^2}{r}J_2(r_\ell)\right] + G_1(r) - \frac{1}{2\pi^2 z^3}\int_0^\infty \mathrm{d}y\, y^3 \sum_{\alpha=e}^{s_{N_s}}\frac{\mathrm{d}\varrho_{\alpha\alpha}}{\mathrm{d}x}}{\displaystyle\sum_{\ell=e,\mu}\left[r_\ell^2 J_2(r_\ell) + J_4(r_\ell)\right] + G_2(r) + \frac{2\pi^2}{15}}\,, \tag{26}$$

where $r_\ell = m_\ell/m_e\, r$. The expressions for the G_1 and G_2 functions, which again take into account the FTQED corrections, are written as [5, 8]:

$$G_{1,2}(x,z) = G_{1,2}^{(2)}(x/z) + G_{1,2}^{(2+\ln)}(x,z) + G_{1,2}^{(3)}(x/z) + \dots\,, \tag{27}$$

$$G_a(r) = \frac{K_2'}{6} - K_2K_2' + \frac{J_2'}{6} + K_2'J_2 + K_2J_2'\,, \tag{28}$$

$$G_1^{(2)}(r) = 2\pi\alpha\left[\frac{1}{r}\left(\frac{K_2}{3} + 2K_2^2 - \frac{J_2}{6} - K_2J_2\right) + G_a\right]\,, \tag{29}$$

$$G_2^{(2)}(r) = -8\pi\alpha\left(\frac{K_2}{6} + \frac{J_2}{6} - \frac{1}{2}K_2^2 + K_2J_2\right) + 2\pi\alpha r G_a\,, \tag{30}$$

$$\begin{aligned} G_1^{(2+\ln)}(x,z) = \frac{e^2 x}{16\pi^4 z^3}\iint_0^\infty \mathrm{d}y\,\mathrm{d}k\,\frac{y\,k}{E_yE_k}\ln\left|\frac{y+k}{y-k}\right| \Bigg\{ &-x\left[z\Big(\partial_x\mathcal{N}_y\partial_z\mathcal{N}_k + \mathcal{N}_y\partial_x\partial_z\mathcal{N}_k\Big) - \mathcal{N}_y\partial_x\mathcal{N}_k\right] \\ &-\mathcal{N}_y\mathcal{N}_k - z\mathcal{N}_y\partial_z\mathcal{N}_k + \frac{x^2(E_y^2+E_k^2)}{2E_y^2E_k^2}\Big(2z\mathcal{N}_y\partial_z\mathcal{N}_k - \mathcal{N}_y\mathcal{N}_k\Big)\Bigg\}\,, \end{aligned} \tag{31}$$

$$G_2^{(2+\ln)}(x,z) = \frac{e^2x^2}{16\pi^4 z^2}\iint_0^\infty \mathrm{d}y\,\mathrm{d}k\,\frac{y\,k}{E_yE_k}\ln\left|\frac{y+k}{y-k}\right|\partial_z\Big(\mathcal{N}_y\partial_z\mathcal{N}_k\Big)\,, \tag{32}$$

$$G_b(r) = \sqrt{K_2 + r^2\frac{K_0}{2}}\,, \tag{33}$$

$$G_c(r) = \frac{2J_2 + r^2J_0}{2\Big(2K_2 + r^2K_0\Big)}\,, \tag{34}$$

$$G_1^{(3)}(r) = \frac{e^3}{4\pi}G_b\left\{\frac{1}{r}\Big(2J_2 - 4K_2\Big) - 2J_2' - r^2J_0' - r\Big(2K_0 + J_0\Big) - G_c\Big[r(K_0-J_0) + K_2'\Big]\right\}\,, \tag{35}$$

$$G_2^{(3)}(r) = \frac{e^3}{4\pi}G_b\left[G_c\Big(2J_2 + r^2J_0\Big) - \frac{2}{r}J_4' - r\Big(3J_2' + r^2J_0'\Big)\right]\,, \tag{36}$$

where the prime denotes derivative with respect to r and we dropped the explicit dependence on r in the expressions for the G functions. For the sake of computational speed, we calculate and store lists for all the terms of Eq. (26) which do not depend on the neutrino density matrix at the initialisation stage, and compute their values through interpolation during the real calculation. The same happens for the energy densities of charged leptons, for which performing an interpolation is much faster than computing an integral. The interpolation, however, can be disabled during compilation (see the `README`), with a $\lesssim 20\%$ increase of the running time.

In order to estimate the effective comoving neutrino temperature w, which is not needed for the calculation but useful to understand the final results, we use an equation similar to Eq. (26), but considering only relativistic electrons, i.e. fixing $r_e = 0$ in the equation.

The finite-temperature electromagnetic corrections are also taken into account in the calculation of the electron mass, used in the collision terms. The contribution to the electron mass, in comoving coordinates, is obtained as [5, 7, 8]:

$$\delta m_e^2(x,y,z) = \frac{2\pi\alpha z^2}{3} + \frac{4\alpha}{\pi}\int_0^\infty \mathrm{d}k\,\frac{k^2}{E_k}\frac{1}{e^{E_k/z}+1} - \frac{x^2\alpha}{\pi y}\int_0^\infty \mathrm{d}k\,\frac{k}{E_k}\log\left|\frac{y+k}{y-k}\right|\mathcal{N}_k\,, \tag{37}$$

so that the comoving electron mass must be replaced using $x^2 \to x^2 + \delta m_e^2$. In the calculation of the collision integrals we ignore the log term that depends on y.

Finally, the effective number of degrees of freedom that the code returns in output is defined as:

$$N_{\mathrm{eff}}^e = \frac{8}{7}\frac{\sum_i \rho_{\nu_i}}{\rho_\gamma} \qquad \text{at early times,} \tag{38}$$

$$N_{\mathrm{eff}}^l = \frac{8}{7}\left(\frac{11}{4}\right)^{4/3}\frac{\sum_i \rho_{\nu_i}}{\rho_\gamma} \qquad \text{at late times,} \tag{39}$$

being ρ_γ the comoving energy density of photons and ρ_{ν_i} the one of the i-th neutrino.

II. COLLISION INTEGRALS

The full collision terms are defined by the sum of the contributions from neutrino–electron/positron scattering and $e^\pm$ annihilation into neutrinos, plus neutrino–neutrino interactions. We neglect other reactions, such as $\mu^\pm$ annihilation (which only affects at very early temperatures when everything is in equilibrium). We therefore have [2, 9]

$$\mathcal{I}[\varrho(y)] = \frac{G_F^2}{(2\pi)^3 y^2}\mathcal{I}^u\,, \tag{40}$$

$$\mathcal{I}^u \equiv \mathcal{I}^u_{\mathrm{sc}} + \mathcal{I}^u_{\mathrm{ann}} + \mathcal{I}^u_{\nu\nu} + \mathcal{I}^u_{\nu\bar\nu}\,, \tag{41}$$

$$\begin{aligned}\mathcal{I}^u_{\mathrm{sc}} = \int \mathrm{d}y_2\mathrm{d}y_3\,\frac{y_2}{E_2}\frac{y_4}{E_4}\\ &\Big\{\left(\Pi_2^s(y,y_4)+\Pi_2^s(y,y_2)\right)\left[F_{\mathrm{sc}}^{LL}\left(\varrho^{(1)},f_e^{(2)},\varrho^{(3)},f_e^{(4)}\right)+F_{\mathrm{sc}}^{RR}\left(\varrho^{(1)},f_e^{(2)},\varrho^{(3)},f_e^{(4)}\right)\right]\\ &-2(x^2+\delta m_e^2)\Pi_1^s(y,y_3)\left[F_{\mathrm{sc}}^{RL}\left(\varrho^{(1)},f_e^{(2)},\varrho^{(3)},f_e^{(4)}\right)+F_{\mathrm{sc}}^{LR}\left(\varrho^{(1)},f_e^{(2)},\varrho^{(3)},f_e^{(4)}\right)\right]\Big\}\,,\end{aligned} \tag{42}$$

$$\begin{aligned}\mathcal{I}^u_{\mathrm{ann}} = \int \mathrm{d}y_2\mathrm{d}y_4\,\frac{y_3}{E_3}\frac{y_4}{E_4}\\ &\Big\{\Pi_2^a(y,y_4)F_{\mathrm{ann}}^{LL}\left(\varrho^{(1)},\varrho^{(2)},f_e^{(3)},f_e^{(4)}\right)+\Pi_2^a(y,y_3)F_{\mathrm{ann}}^{RR}\left(\varrho^{(1)},\varrho^{(2)},f_e^{(3)},f_e^{(4)}\right)\\ &+(x^2+\delta m_e^2)\Pi_1^a(y,y_2)\left[F_{\mathrm{ann}}^{RL}\left(\varrho^{(1)},\varrho^{(2)},f_e^{(3)},f_e^{(4)}\right)+F_{\mathrm{ann}}^{LR}\left(\varrho^{(1)},\varrho^{(2)},f_e^{(3)},f_{e^+}^{(4)}\right)\right]\Big\}\,,\end{aligned} \tag{43}$$

$$\mathcal{I}^u_{\nu\nu} = \frac{1}{4}\int \mathrm{d}y_2\mathrm{d}y_3\,\Pi_2^\nu(y,y_2)F_{\nu\nu}\left(\varrho^{(1)},\varrho^{(2)},\varrho^{(3)},\varrho^{(4)}\right)\,, \tag{44}$$

$$\mathcal{I}^u_{\nu\bar\nu} = \frac{1}{4}\int \mathrm{d}y_2\mathrm{d}y_3\,\Pi_2^\nu(y,y_4)F_{\nu\bar\nu}\left(\varrho^{(1)},\varrho^{(2)},\varrho^{(3)},\varrho^{(4)}\right)\,, \tag{45}$$

where $E_i^2 = \sqrt{x^2 + y_i^2 + \delta m_e^2}$ and

$$\Pi_1^s(y, y_3) = y\, y_3\, D_1 + D_2(y, y_3, y_2, y_4), \quad (46)$$

$$\Pi_1^a(y, y_2) = y\, y_2\, D_1 - D_2(y, y_2, y_3, y_4), \quad (47)$$

$$\Pi_2^s(y, y_2)/2 = y\, E_2\, y_3\, E_4\, D_1 + D_3 - y\, E_2 D_2(y_3, y_4, y, y_2) - y_3\, E_4 D_2(y, y_2, y_3, y_4), \quad (48)$$

$$\Pi_2^s(y, y_4)/2 = y\, E_2\, y_3\, E_4\, D_1 + D_3 + E_2\, y_3 D_2(y, y_4, y_2, y_3) + y\, E_4 D_2(y_2, y_3, y, y_4), \quad (49)$$

$$\Pi_2^a(y, y_3)/2 = y\, y_2\, E_3\, E_4\, D_1 + D_3 + y\, E_3 D_2(y_2, y_4, y, y_3) + y_2\, E_4 D_2(y, y_3, y_2, y_4), \quad (50)$$

$$\Pi_2^a(y, y_4)/2 = y\, y_2\, E_3\, E_4\, D_1 + D_3 + y_2\, E_3 D_2(y, y_4, y_2, y_3) + y\, E_4 D_2(y_2, y_3, y, y_4), \quad (51)$$

$$\Pi_2^\nu(y, y_2)/2 = y\, y_2\, y_3\, y_4\, D_1 + D_3 - y\, y_2 D_2(y_3, y_4, y, y_2) - y_3\, y_4 D_2(y, y_2, y_3, y_4), \quad (52)$$

$$\Pi_2^\nu(y, y_4)/2 = y\, y_2\, y_3\, y_4\, D_1 + D_3 + y_2\, y_3 D_2(y, y_4, y_2, y_3) + y\, y_4 D_2(y_2, y_3, y, y_4), \quad (53)$$

where the functions D_i have the following definitions [10]:

$$D_1(a, b, c, d) = \frac{16}{\pi} \int_0^\infty \frac{\mathrm{d}\lambda}{\lambda^2} \prod_{i=a,b,c,d} \sin(\lambda i)\,, \quad (54)$$

$$D_2(a, b, c, d) = -\frac{16}{\pi} \int_0^\infty \frac{\mathrm{d}\lambda}{\lambda^4} \prod_{i=a,b} \Big[\lambda i \cos(\lambda i) - \sin(\lambda i)\Big] \prod_{j=c,d} \sin(\lambda j)\,, \quad (55)$$

$$D_3(a, b, c, d) = \frac{16}{\pi} \int_0^\infty \frac{\mathrm{d}\lambda}{\lambda^6} \prod_{i=a,b,c,d} \Big[\lambda i \cos(\lambda i) - \sin(\lambda i)\Big]\,. \quad (56)$$

The three functions can be written in a more efficient way for the calculation, since they can be solved analytically, see e.g. [11] for the complete expressions.

Finally, the functions that define the phase space factors in the collision terms are [2, 9]:

$$\begin{aligned} F_{\rm sc}^{ab}\left(\varrho^{(1)}, f_e^{(2)}, \varrho^{(3)}, f_e^{(4)}\right) = {} & f_e^{(4)}(1 - f_e^{(2)}) \Big[G^a \varrho^{(3)} G^b (1 - \varrho^{(1)}) + (1 - \varrho^{(1)}) G^b \varrho^{(3)} G^a\Big] \\ & - f_e^{(2)}(1 - f_e^{(4)}) \Big[\varrho^{(1)} G^b (1 - \varrho^{(3)}) G^a + G^a (1 - \varrho^{(3)}) G^b \varrho^{(1)}\Big]\,, \end{aligned} \quad (57)$$

$$\begin{aligned} F_{\rm ann}^{ab}\left(\varrho^{(1)}, \varrho^{(2)}, f_e^{(3)}, f_e^{(4)}\right) = {} & f_e^{(3)} f_e^{(4)} \Big[G^a (1 - \varrho^{(2)}) G^b (1 - \varrho^{(1)}) + (1 - \varrho^{(1)}) G^b (1 - \varrho^{(2)}) G^a\Big] \\ & - (1 - f_e^{(3)})(1 - f_e^{(4)}) \Big[G^a \varrho^{(2)} G^b \varrho^{(1)} + \varrho^{(1)} G^b \varrho^{(2)} G^a\Big]\,, \end{aligned} \quad (58)$$

$$\begin{aligned} F_{\nu\nu}\left(\varrho^{(1)}, \varrho^{(2)}, \varrho^{(3)}, \varrho^{(4)}\right) = {} & \left(1 - \varrho^{(1)}\right) \varrho^{(3)} \Big[\left(1 - \varrho^{(2)}\right) \varrho^{(4)} + \mathrm{Tr}(\cdots)\Big] \\ & - \varrho^{(1)} \left(1 - \varrho^{(3)}\right) \Big[\varrho^{(2)} \left(1 - \varrho^{(4)}\right) + \mathrm{Tr}(\cdots)\Big] + \text{h.c.} \end{aligned} \quad (59)$$

$$\begin{aligned} F_{\nu\bar{\nu}}\left(\varrho^{(1)}, \varrho^{(2)}, \varrho^{(3)}, \varrho^{(4)}\right) = {} & \left(1 - \varrho^{(1)}\right) \left(1 - \varrho^{(2)}\right) \Big[\varrho^{(4)} \varrho^{(3)} + \mathrm{Tr}(\cdots)\Big] \\ & - \varrho^{(1)} \varrho^{(2)} \Big[\left(1 - \varrho^{(4)}\right) \left(1 - \varrho^{(3)}\right) + \mathrm{Tr}(\cdots)\Big] \\ & + \left(1 - \varrho^{(1)}\right) \varrho^{(3)} \Big[\varrho^{(4)} \left(1 - \varrho^{(2)}\right) + \mathrm{Tr}(\cdots)\Big] \\ & - \varrho^{(1)} \left(1 - \varrho^{(3)}\right) \Big[\left(1 - \varrho^{(4)}\right) \varrho^{(2)} + \mathrm{Tr}(\cdots)\Big] + \text{h.c.}, \end{aligned} \quad (60)$$

where $\varrho^{(i)} = \varrho(y_i)$ and $f_e^{(i)} = f_{\rm FD}(y_i, z)$ represent the momentum distribution function of the various particles, and $\mathrm{Tr}(\cdots)$ denotes the trace of the term immediately before it. The full expression for these functions should take into account the lepton asymmetry and distinguish the momentum distributions of leptons/neutrinos from those of antilepton/antineutrinos. Since we do not include lepton asymmetry, we just report the expressions without the heavier notation required to distinguish the various terms.

Concerning the neutrino–neutrino collision terms [9], there are few more points to be discussed. We can see from Eqs. (59)–(60) that the expressions include up to four products of the neutrino density matrices. With a smart implementation of the integration methods (see sections V and VI), one can use for three of the terms (namely, the ones containing y_1, y_2 and y_3) the values obtained on the momentum grid. For the fourth occurrence, $\varrho(y_4)$, an interpolation is needed. We tested several possibilities, and implemented a scheme that computes the linear interpolation of $\varrho_{\alpha\alpha}/f_{\rm FD}$ on the diagonal, and the linear interpolation of $\varrho_{\alpha\beta}$ ($\alpha \neq \beta$) for the off-diagonal, as it guarantees a better numerical stability (see [9]). Nevertheless, the selection of the momentum grid may have a strong impact on the results obtained using neutrino–neutrino collision terms, as we will discuss in section VII. We have also verified that the numerical values of $N_{\rm eff}$ obtained including the integrals in Eqs. (44)–(45) does not vary significantly if we consider only the diagonal components of ϱ, or we instead consider the full matrix. The reason is that the off-diagonal terms are typically much smaller than the diagonal ones, so that their combinations are suppressed and give a small contribution to the collision terms. Ignoring the off-diagonal terms, therefore, gives a very precise result with a slightly smaller computational cost. Notice also that the implementation of the neutrino–neutrino collision terms as described here is the first that allows to compute off-diagonal collision terms taking into account the full neutrino density matrix [9].

Damping approximation

The code can compute the collision terms according to Eqs. (42)–(45), but the integrals are very expensive. For the non-diagonal terms of the collision matrix we therefore allow the possibility to use damping approximations, in the form

$$\mathcal{I}^u_{\alpha\beta}(\varrho) = -D^u_{\alpha\beta}\varrho_{\alpha\beta}, \tag{61}$$

for $\alpha \neq \beta$. Notice that $\mathcal{I}^u_{\alpha\beta}(\varrho)$ is the unnormalized term defined in Eq. (41). The expressions for the coefficients $D^u_{\alpha\beta}$ depend on the elements considered. In case more than one sterile state is considered, the terms $D^u_{s_is_j}$ are always zero. For the rest of the terms, two possibilities are implemented in the code.

a. The first possibility arises from the calculations presented in the Appendix B of [9]. Notice that the appendix proposes also a damping approximation for diagonal collision terms, which however is not implemented in `FortEPiaNO`. According to [9], we have a contribution from neutrino–neutrino interactions that is flavour-blind (as expected, since in equilibrium there are the same number of neutrinos and antineutrinos of each flavour), and one from neutrino–electron collisions. We can write the damping terms in the following forms:

$$\begin{aligned}
\big\{D^u(y)\big\}_{\alpha\beta} &= \frac{1}{2}\Big[\big\{R^u(y)\big\}_\alpha + \big\{R^u(y)\big\}_\beta\Big]\,, &(62)\\
\big\{R^u_{\nu\nu}(y)\big\}_\alpha &= \int \mathrm{d}y_2\mathrm{d}y_3\,\left[\Pi_2^\nu(y,y_2)+2\Pi_2^\nu(y,y_4)\right]\\
&\qquad\times\Big([1-f_{\rm eq}(y_2)]f_{\rm eq}(y_3)f_{\rm eq}(y_4)+f_{\rm eq}(y_2)[1-f_{\rm eq}(y_3)][1-f_{\rm eq}(y_4)]\Big)\\
&\equiv \mathcal{D}^u(y), &(63)\\
\{R^u_{\nu e}(y)\}_\alpha &= \frac{1}{2}\,\left[(2\sin^2\theta_W\pm1)^2_\alpha+4\sin^4\theta_W\right]\int \mathrm{d}y_2\mathrm{d}y_3\,\left[\Pi_2^\nu(y,y_2)+2\Pi_2^\nu(y,y_4)\right]\\
&\qquad\times\Big([1-f_{\rm eq}(y_2)]f_{\rm eq}(y_3)f_{\rm eq}(y_4)+f_{\rm eq}(y_2)[1-f_{\rm eq}(y_3)][1-f_{\rm eq}(y_4)]\Big)\\
&= \frac{1}{4}\left[(2\sin^2\theta_W\pm1)^2_\alpha+4\sin^4\theta_W\right]\mathcal{D}^u(y), &(64)
\end{aligned}$$

where in the prefactor $(2\sin^2\theta_W \pm 1)_\alpha$ the plus sign "+" is understood to apply to $\alpha = e$ and "−" to $\alpha = \mu, \tau$.

For relativistic Fermi–Dirac distributions, the function $\mathcal{D}^u(y)$ evaluates to

$$\mathcal{D}^u(y) = 2y^3z^4 d(y/z)\,, \tag{65}$$

where $d(s)$ is a number around 100 which can be obtained as a double momentum integral, see Eq. (B.9) in [9]. For computational ease, $d(s)$ can be fitted in the interval $s \in [10^{-4}, 10^3]$ to better than 0.1% accuracy by the curve

$$d_{\rm fit}(s) = d_0 e^{-1.01s} + d_\infty(1-e^{-0.01s}) + (e^{-0.01s}-e^{-1.01s})\left[\frac{a_0+a_1\ln(s)+a_2\ln^2(s)}{1+b_1\ln(s)+b_2\ln^2(s)}\right]\,, \tag{66}$$

where $d_0 = 129.875$ and $d_\infty = 100.999$ are the asymptotic values of the function as $s \to 0$ and $s \to \infty$ respectively, and the fitting coefficients are $a_0 = 90.7332$, $a_1 = -48.4473$, $a_2 = 20.1219$, $b_1 = -0.529157$, and $b_2 = 0.20649$.

b. The second possibility is to implement the coefficients derived in [12], see also [13, 14], under the assumption

$$\varrho(y) = \frac{f_{\rm eq}(y)}{f_{\rm eq}(\langle y\rangle)}\varrho(\langle y\rangle)\,, \tag{67}$$

such that (y) at all modes y evolve in phase, where $\langle y \rangle$ denotes a representative momentum. Upon integration in y, the procedure yields a thermally-averaged collision term, which can be written in the form

$$D^u_{e\mu}/F = D^u_{e\tau}/F = 15 + 8\sin^4\theta_W\,, \tag{68}$$

$$D^u_{\mu\tau}/F = 7 - 4\sin^2\theta_W + 8\sin^4\theta_W\,, \tag{69}$$

$$D^u_{es}/F = 29 + 12\sin^2\theta_W + 24\sin^4\theta_W\,, \tag{70}$$

$$D^u_{\mu s}/F = D^u_{\tau s}/F = 29 - 12\sin^2\theta_W + 24\sin^4\theta_W\,, \tag{71}$$

where $F = 7\pi^4 z^4 y^3/135$ is a common normalisation coefficient.

III. LOW REHEATING

The code can also be executed taking into account a simple low-reheating model, such as the one considered in [15**?**]. In such model, the Universe is initially in a matter-domination stage, where almost the totality of the matter energy density is provided by a massive scalar field, which can be the inflaton or another new particle. Photons are almost absent at the beginning of the evolution, but the decay of the scalar field into standard-model relativistic particles other than neutrinos (photons, electrons, muons if any) populates the radiation components. Neutrinos are populated via weak interactions with charged leptons and may reach equilibrium with the rest of the thermal plasma if the decay occurs early enough. In case the decay starts too late, the neutrino fluid cannot reach equilibrium with the rest of the relativistic particles, so that the neutrino energy density is smaller than in the standard case and the final $N_{\rm eff}$ is smaller than expected.

In order to compute the evolution of the Universe within a low-reheating scenario, the equations discussed in Sec. I must be modified. First of all, one should define a quantity that describes the decay of the scalar field. One simple way is to define a reheating temperature $T_{\rm RH}$, which at the end of the day is a different way of referring to the decay rate of the massive particle: one can write

$$\Gamma_\phi = 3H(T_{\rm RH})\,, \tag{72}$$

where H is the Hubble factor as a function of the reheating temperature. Assuming

$$H\,, \tag{73}$$

one obtains that the reheating temperature can be rewritten as

$$T_{\rm RH} = \,. \tag{74}$$

The evolution of the energy density of the scalar field, in physical coordinates, can be written as a function of its decay width:

$$\rho_\phi = \,, \tag{75}$$

while in comoving coordinates the same equation reads:

$$\rho_\phi = \,. \tag{76}$$

The initial condition on ρ_ϕ is computed using:

$$\rho_\phi^{\rm in} = \,. \tag{77}$$

The other equation we need to modify proceeds from the continuity equation of the total energy-momentum in the expanding Universe, which in physical coordinates is

$$\rho_t\,. \tag{78}$$

From this expression and using Eq. (76), one can derive a modified version of Eq. (26) which describes the evolution of the photon temperature, which is the only variable we need to describe the relativistic plasma, apart for the neutrino component. The comoving photon temperature, in a low-reheating scenario, evolves according to

$$dz/dx\,, \tag{79}$$

which is very similar to Eq. (26). The evolution of the neutrino fluid, instead, is still the one given by Eq. (25).

In order to solve the equations in the low-reheating case, therefore, we have to compute the evolution of the same variables as in the standard scenario, plus the scalar field energy density. For convenience, together with the comoving variable x, the code also saves the physical time t, which can be obtained using:

$$t = \,. \tag{80}$$

IV. SOLVER AND INITIAL CONDITIONS

We solve the differential equations with the `DLSODA` routine from the `ODEPACK`[16] Fortran package [17, 18]. `ODEPACK` is a collection of solvers for the initial value problem for systems of ordinary differential equations. It includes methods to deal with stiff and non-stiff systems, and some of the provided subroutines automatically recognise which type of problem they are facing.

The specific solver we use, `DLSODA`, is a modification of the Double-precision Livermore Solver for Ordinary Differential Equations (`DLSODE`) which includes an automatic switching between stiff and non-stiff problems of the form $\mathrm{d}y/\mathrm{d}t = f(t, y)$. In the stiff case, it treats the Jacobian matrix $\mathrm{d}f/\mathrm{d}y$ as either a dense (full) or a banded matrix, and as either user-supplied or internally approximated by difference quotients. It uses Adams methods (predictor–corrector) in the non-stiff case, and Backward Differentiation Formula (BDF) methods (the Gear methods) in the stiff case. The linear systems that arise are solved by direct methods (LU factor/solve). For more details, see the original publications [17, 18].

The initial conditions for `DLSODA` are defined as follows. The initial time $x_{\rm in}$ is an input parameter of the code, and reasonable values would correspond to temperatures between a few hundreds and a few tens of MeV. The initial comoving photon temperature is computed evolving Eq. (26) from even earlier times ($z_0 = 1$ at $T_0 = 10\, m_\mu$, $x_0 = m_e/T_0$) until $x_{\rm in}$. The obtained value $z_{\rm in}$ is then considered as the temperature of equilibrium of the entire plasma. Concerning the neutrino density matrix at $x_{\rm in}$, all off-diagonal elements and the diagonal ones for sterile neutrinos are fixed to zero, while the diagonal elements corresponding to active neutrinos are Fermi–Dirac distributions with a temperature $z_{\rm in}$. For typical values that we use in the code, we have $z_{\rm in} - 1 = 2.9 \times 10^{-4}$ for $x_{\rm in} = 0.001$ (which we use for the 3+1 cases) or $z_{\rm in} = 1.098$ for $x_{\rm in} = 0.05$ (suitable for the three-neutrino case, see [2]).

V. MOMENTUM GRID

In order to follow the evolution of Eq. (1), we discretise its dependence on y and evolve each of the momentum in x. One of the most interesting ways to make the code more precise and faster is related to the choice of the y_i. Discretising the momenta with a linear or logarithmic spacing works[19], but it is not the most efficient way to generate the grid. Inspired by one of the methods used in `CLASS` (see [20]), we deeply tested and finally considered a spacing based on the Gauss–Laguerre (GL) integration method. The crucial point of the calculation is to compute the energy density of neutrinos, given by

$$\rho_\alpha = \frac{1}{\pi^2} \int_0^\infty \mathrm{d}y\, y^3\, \varrho_{\alpha\alpha}(y), \tag{81}$$

where $\varrho_{\alpha\alpha}(y)$ will be close to a Fermi–Dirac distribution and in any case always exponentially suppressed. The Gauss–Laguerre quadrature (see e.g. [21]) is a method that is designed to optimise the solution of integrals of the type

$$I = \int_0^\infty \mathrm{d}x\, y^\alpha\, e^{-y}\, f(y) \simeq \sum_i^N w_i^{(\alpha)}\, f(y_i)\,, \tag{82}$$

where $f(y)$ is a generic function, y_i are the N roots of the Laguerre polynomial L_N of order N, and w_i are relative weights, which are obtained using

$$w_i^{(\alpha)} = \frac{y_i}{(N+1)^2 \left[L_{N+1}^{(\alpha)}(y_i)\right]^2}. \tag{83}$$

The weights can be computed for example using the `gaulag` routine from [21]. Since our momentum distribution function is not directly proportional to e^{-y}, we consider $f(x) = e^y\, \varrho_{\alpha\alpha}(y)$, in order to rescale the weights appropriately.

For the simple purpose of integrating the Fermi–Dirac distribution, very few points are typically required. `CLASS`, for example, uses order of ten points for integrating the neutrino distribution. In our case the non-thermal distortions must be computed accurately, and in particular when evolving the thermalisation of a sterile neutrino we need more precision on the small momenta. On the other hand, we do not want to compute the momentum distribution function at very high y, which gives a very small contribution to the total integral. We therefore use a truncated list of nodes y_i over which to compute the evolution of ϱ, selecting only the $N_y \le N$ nodes for which $y_i < 20$. In this way we can increase the number of points at small y and the resolution on the thermalisation processes without having to compute a large number of points at high momentum. The number of points we can use is limited by the accuracy of the algorithm that computes the w_i. For the `gaulag` routine [21], our setup allows to reach $N_y \sim 50$ when $N \sim 350$, when $y_i < 20$. This number of momentum nodes is already large enough to reach a precision much better than one per mille on $N_{\rm eff}$, which is the same we could obtain with a linear spacing of the points and $N_y = 100$ [2]. Since the evaluation of the collision integrals scales as N_y^2 and the number of derivatives in Eq. (25) scales with N_y, this ensures a significant gain. Unfortunately, since the method used to compute the GL nodes does not allow to increase N_y arbitrarily, the GL method has limitations when the neutrino–neutrino collision terms are considered. We further comment on these points in the next sections.

VI. NUMERICAL CALCULATION OF 1D AND 2D INTEGRALS

Most of the processing time is spent to compute the collision integrals discussed in section II, which are two-dimensional integrals in the momentum. We compute the integrals using a two-dimensional version of the Gauss–Laguerre method, which has been tested to be precise enough,

$$\int_{x_1}^{x_N}\int_{y_1}^{y_M} \mathrm{d}x\,\mathrm{d}y\, f(x,y) = \sum_{i=1}^{N}\sum_{j=1}^{M} w_i\, w_j\, f_{ij}\,. \tag{84}$$

This works under the assumption that $f(x,y)$ is exponentially suppressed both in x and y. Such assumption is valid in our case, as the functions F_{ab} always contain products of momentum distribution functions, which are typically very close to the Fermi–Dirac. The only exception is the case of the additional neutrino, for which the distribution can be very different from the Fermi–Dirac, but in any case it is always exponentially suppressed, since the lowest momenta are always populated first and its momentum distribution can never exceed the one of standard neutrinos.

When using a linear/logarithmic spacing of points, instead we perform the integrals using a composite two-dimensional Newton–Cotes (NC) formula of order 1 [22]:

$$\int_{x_1}^{x_N}\int_{y_1}^{y_M} \mathrm{d}x\,\mathrm{d}y\, f(x,y) = \sum_{i=1}^{N-1}\sum_{j=1}^{M-1} (x_j - x_i)(y_j - y_i)\left[\frac{f_{ij} + f_{i+1,j} + f_{i,j+1} + f_{i+1,j+1}}{4}\right]\,, \tag{85}$$

where we used the short notation $f_{i,j} = f(x_i, y_j)$, while i and j run over the grid of momenta we are using, which contains $N = M = N_y$ points for each dimension. This avoids us the need to interpolate the density matrix in points outside the momentum grid.

The integrals therefore require N_y^2 evaluations of the integrands at each evaluation: this means that reducing the value of N_y by a factor of two gives a factor four faster calculation of the integrals. The actual gain in the code is even larger, since the `DLSODA` algorithm needs to explore less combinations of variations in the $\varrho_{\alpha\beta}(y_l)$ for the different y_l in the momentum grid. Our goal is therefore to obtain with a coarse grid a result that is in reasonable agreement with the one obtained using a fine grid.

In order to obtain the maximum speed, we study the accuracy of each function that enters the code in comparison with the analytical results, were they can be obtained. The number of points and the integration methods adopted in all the integrals, for example, have been carefully studied to achive a reasonable precision with a short computation time. For the two-dimensional integrals, the selected momentum grid fully defines the integration procedure, and the precision is always good when using a reasonable number of points. Depending on the function, we may adopt the Gauss–Laguerre, Newton–Cotes or Romberg integration [23] methods for the one-dimensional integrals. In particular, for the electron and muon energy densities and for most of the funcions that enter the calculation of Eq. (26) we use a Gauss–Laguerre method on a dedicated grid of up to 110 points for the most complicated functions. In one single case, the $K'(r)$ function derived from Eq. (12), the result obtained with the Gauss–Laguerre method did not reach the requested precision and we decided to use a Romberg integration instead. Although this requires a longer computation time, it only affects the initialisation stage, as in the code we interpolate over the pre-computed values. The number of points and the interpolation range have also been studied in order to obtain sufficiently precise results for all the computations required in the code.

VII. PRECISION OF THE FINAL RESULTS

We have tested our code with the results available in the literature and verified the robustness of our findings against changes in the settings used in the calculations. In particular, we refer to the high-precision results in the three-neutrino case of [2], from which we have adopted most of the equations. Most of the results summarized here are discussed more in details in [1, 9].

Concerning the value of $N_{\rm eff}$ that we obtain using only active neutrinos, if we ignore neutrino–neutrino collision terms, we verified that we can reach much better than per mille stability on $N_{\rm eff} = 3.044$ using $N_y \geq 20$ points spaced with the Gauss–Laguerre method, if the tolerance for `DLSODA` [24] is 10^{-6}. This means that using $N_y = 50$ instead of $N_y = 20$ does not significantly alter the result. If we want to consider a linear or logarithmic spacing for the momentum grid, a minimum of 40 grid points must be employed in order to reach the same level of stability. Another possible setting that can give us a faster execution of the code is the precision used for `DLSODA`. We verified that once the tolerance for `DLSODA` is smaller than 10^{-5}, the results are already stable at a level much better than per mille (actually closer to the 0.1 per mille) with respect to the most precise case considered here ($N_y = 50$, tolerance 10^{-6}). Using a tolerance of 10^{-4} gives a value of $N_{\rm eff}$ which is stable at the level of few per mille, and still better than 1%.

A full calculation of $N_{\rm eff}$, in any case, must be performed taking into account also the neutrino–neutrino collision terms. Including them raises $N_{\rm eff}$ by approximately 0.002, depending on the values of the oscillation parameters and the considered momentum grid. Mostly because of the interpolation required to compute $\varrho(y_4)$ in Eqs. (59)–(60), the numerical uncertainties grow significantly: when a coarse grid is considered, the interpolation is much less precise and the final results are much more instable. As expected, however, the instability decreases when the number of momentum nodes is increased. It is worth noticing that a GL grid tends to give a slightly higher estimation of $N_{\rm eff}$ than a grid with linearly spaced momentum nodes, with a difference of ~ 0.001 between the GL case with $N_y = 50$ and a mixed linear/logarithmic spacing with $N_y = 100$, when requiring all the nodes to satisfy $y_i < 20$. Since our GL method does not allow to increase arbitrarily the number of nodes when imposing the upper limit on their value, we cannot have a more precise estimate of the effect of the momentum grid, nor to determine at which N_y the two different momentum grid schemes converge to the same value of $N_{\rm eff}$ when requiring a higher precision.

Considering the implementation of the off-diagonal collision terms, we find that the $N_{\rm eff}$ output does not change significantly if we use the full integrals or the damping terms, both for neutrino–neutrino and neutrino–electron contributions. The damping formulas are sufficiently good to obtain a very precise result and allow to save a lot of computation time.

If we repeat the same exercise in the 3+1 scheme, using $\Delta m^2_{41} = 1.29$ eV^2, $|U_{e4}|^2 = 0.012$ [25] and $|U_{\mu4}|^2 = |U_{\tau4}|^2 = 0$, we find similar conclusions. A tolerance of 10^{-5} gives results very close to those obtained with 10^{-6}, while any larger tolerance gives larger fluctuations depending on N_y. With 10^{-4}, the precision remains of the order of 0.5%, so it is still safe to compute the value of $N_{\rm eff}$ on a grid of active-sterile mixing parameters using this level of precision. With $N_y = 20$, a single run takes a few minutes on four cores, and the running time is not significantly affected by changes in the `DLSODA` tolerance. When more precision is required, however, the algorithm may have troubles in resolving some of the resonances, and in that case the run can take much longer because of the adaptive nature of the solver.

Another parameter that we tested is the initial value of x, $x_{\rm in}$. Apart for fluctuations which are compatible with those obtained varying N_y, the result is stable against variations in $5 \times 10^{-4} \leq x_{\rm in} \leq 5 \times 10^{-2}$. The largest values of $x_{\rm in}$ may be inappropriate for high values of Δm^2_{41}, as it is important for the solver to start the evolution before the sterile state starts to oscillate significantly with the active ones. Smaller values, on the contrary, may create numerical problems in `DLSODA` due to the very small initial value $z_{\rm in} - 1$, and are never really required for our purposes.

More details on the precision of numerical calculations and the dependence of $N_{\rm eff}$ on physical parameters (including Fermi constant, Weinberg angle mass and neutrino mixing parameters) are discussed in [9], considering the three-neutrino case only. In order to study the accuracy of the numerical calculations as functions of the physical parameters, the tolerance for `DLSODA` has been set to 10^{-7}. Considering variations for the physical parameters in the allowed 3σ range (from [26] for the neutrino mixing parameters, from [27] for the weak interaction parameters), we see that the stability of $N_{\rm eff}$ is at the 10^{-4} level or better. A similar precision is obtained when changing the way the off-diagonal terms. Altering $x_{\rm in}$, which controls the initial time in the code, only affects $N_{\rm eff}$ at the 10^{-5} level. More critical is the dependence on N_y, which controls the number and the position of the nodes of the momentum grid. Varying N_y between 25 and 50 only alters the final result at the 10^{-4} level, if neutrino–neutrino collision terms are ignored, but has a much bigger impact if they are considered. In general, variations up to 0.001 can be expected if one includes neutrino–neutrino collisions and changes the momentum grid (using a linear spacing or a GL one for the y nodes), assuming that a sufficiently high number of nodes is considered ($N_y \geq 40$ for GL spacing of the nodes, $N_y \geq 80$ for a linear one).

As a summary, a reasonable estimate of the theoretical and numerical uncertainty on the recommended value obtained considering three active neutrinos, $N_{\rm eff} = 3.045$, is of order 10^{-3}.

VIII. OUTPUT FILES

The successful execution of `FortEPiaNO` will always store a limited number of files in the output folder:

- `ini.log`: a summary of the input parameters as they are read by the Fortran code.

- `messages.log`: summary of important messages from the code execution.
- `time.log`: a time log which indicates at what time `FortEPiaNO` started and finished the calculation of each x step.
- `rho_final.dat` and `rho_final_mass.dat`: the diagonal of the final neutrino density matrix, in flavor or mass basis, respectively. The first column of each file contains the grid of momenta y_i, while the following ones contain $\varrho_{jj}(y_i)$, where j is the column index minus one and i is the row index. From these two files, others can be created a posteriori by the included python scripts (see Section XI), storing the final neutrino density matrix content using different normalizations. The `rho_final*_var.dat` file stores the grid of momenta y_i and $\varrho_{jj}(y_i)/f_{\rm FD}(y_i/w_{\rm fin})$, for each flavor/mass eigenstate. This is a multiplicative factor with respect to the equilibrium distribution that the neutrinos would have if considering instantaneous decoupling. The `rho_final*_norm.dat` file, instead, contains a normalized $\varrho_{jj}(y'_i)/f_{\rm FD}(y'_i)$, which gives the correct contribution from the neutrinos to the effective number of relativistic species when computing the photon density using the standard temperature $z/w = (11/4)^{1/3}$. These values are prepared to be used in cosmological codes such as CLASS or CAMB. check
- `resume.dat`: it reports the final values of the most important quantities: the photon and effective neutrino temperatures, z and w, the variation in neutrino energy density with respect to the expected value at equilibrium, the final value of $N_{\rm eff}$ and the considered value of $T_{\rm RH}$ (only if low-reheating is considered).

In addition to the minimal output files that we discussed above, `FortEPiaNO` can save many more useful quantities, controlled by specific flags in the configuration file:

- `save_BBN`: store a set of files that contain quantities employed in Big Bang Nucleosynthesis calculations. Although the files stored by the Fortran code are incomplete, the associated python utilities (see Section XI) can be used to complete the information, for later use of the results with the `PArthENoPE` code [28**?** , 29]. In particular, the relevant quantities are z, w, ρ_ν, $d\rho_\nu/dx$, $\rho_{\rm rad}$, $\rho_{\rm tot}$ and $\varrho_{ee}(y_i)$ as functions of x, plus the grid of momenta, y_i. The derivative $d\rho_\nu/dx$ is computed by the python code using `numpy.gradient` and smoothed, while the rest of the quantities are saved directly by the Fortran code, but filtered by the python scripts in order to consider only the values of x for which radiation dominates the energy density, i.e. $\rho_{\rm rad} > 0.99\rho_{\rm tot}$.
- `save_energy_entropy_evolution`: save the evolution of the energy density and entropy density of all fluids as a function of x.
- `save_fd`: save the grid of momenta y_i and the initial Fermi-Dirac distribution $f_{\rm FD}(y_i)$.
- `save_Neff`: save the evolution of the effective number $N_{\rm eff}$ as a function of x.
- `save_nuDens_evolution`: save the evolution of each component of the neutrino density matrix as a function of x. Diagonal and off-diagonal components (real and imaginary part) will be saved, if available.
- `save_number_evolution`: save the evolution of the number density of all fluids as a function of x.
- `save_w_evolution`, `save_z_evolution`: save the photon and effective neutrino temperatures, z and w. The latter is not saved if `save_z_evolution` is disabled.

IX. DEFAULTS PARAMETERS

The numerical constants in the code (defined in `sources/const.f90`) are mostly taken from [27]. The default configuration of the code (`ini/explanatory.ini`) is the following:

- **Neutrino parameters**: three active neutrinos (`flavorNumber = 3`), with mixing parameters from [27]: `givesinsq = T` (the following parameters `thetaij` provide the value of $\sin^2\theta_{ij}$), `theta12 = 0.297`, `theta13 = 0.0215`, `theta23 = 0.425`, `dm21 = 7.37e-05` (eV^2), `dm31 = 0.00256` (eV^2).
- **Collision integrals**: non-zero (`collint_diagonal_zero = F`) diagonal terms computed taking into account only the contribution from neutrino–electron interactions (`collint_d_no_nue = F`, `collint_d_no_nunu = T`), off-diagonal terms computed using damping approximations (`collint_offdiag_damping = T`) from Eqs. (62)–(66) (`collint_damping_type = 1`), also in this case considering only the neutrino–electron interactions (`collint_od_no_nue = F`, `collint_od_no_nunu = T`).

- **FTQED**: finite temperature corrections are enabled (`ftqed_temperature_corr = T`), using third order corrections (`ftqed_ord3 = T`) but no logarithmic ones (`ftqed_log_term = F`), and including corrections to the electron mass in the matter potentials (`ftqed_e_mth_leptondens = T`).
- **Grid settings**: the x range is considered between `x_in = 0.01` and `x_fin = 35`, and the quantities saved in `Nx = 500` bins. For the momentum grid, the default configuration considers `Ny = 30` momentum nodes spaced according to the Gauss–Laguerre method (`use_gauss_laguerre = T`), as neutrino–neutrino collision terms are disabled. Remember that a correct run with neutrino–neutrino collisions is better performed using a much denser momentum grid, that will require to disable the Gauss–Laguerre nodes selection.
- **Output settings**: the default output folder is `outputFolder = output`, checkpoint are active (`checkpoint = T`), if a `resume.dat` file is present in the output folder it will be removed and the run repeated (`force_replace = T`), all the optional outputs are enabled (`save_BBN = T`, `save_energy_entropy_evolution = T`, `save_fd = T`, `save_Neff = T`, `save_nuDens_evolution = T`, `save_number_evolution = T`, `save_w_evolution = T`, `save_z_evolution = T`).
- **Precision**: `dlsoda_rtol = 1.d-6`, `dlsoda_atol_z = 1.d-6`, `dlsoda_atol_d = 1.d-6`, `dlsoda_atol_o = 1.d-6`.
- **Low-reheating**: considered only when the code is compiled with `LOW_REHEATING=1`. The only additional parameter is the low-reheating temperature, by default $T_{\rm RH} = 25$ MeV, which is high enough to mimic a scenario without low-reheating.

For more available parameters and the description of all of them, see `ini/explanatory.ini`.

X. PRECOMPILER OPTIONS

The compilation of `FortEPiaNO` may be performed including a number of precompiler options that alter the behaviour of the code. The advantage of precompiler options is that the related features are enabled or disabled before the code is compiled, and their existence in the source code does not affect the execution speed when they are disabled.

Available precompiler options currently include:

- `GLR_ZERO_MOMENTUM`: use g_L, $\tilde{g}_L$ and g_R values computed at zero momentum transfer [30] in Eq. (5).
- `FULL_F_AB`: full matrix multiplication when computing neutrino–electron collision integrals (Eqs. (57) and (58)). No effect on the code execution unless the diagonal matrices $G_{L,R}$ are modified to be off-diagonal.
- `FULL_F_NU`: take into account the full neutrino density matrix when computing neutrino–neutrino collision integrals (Eqs. (59) and (60)). By default, the code only considers the diagonal elements of the neutrino density matrix, as off-diagonal ones are typically much smaller.
- `LOW_REHEATING`: enable calculations in a low-reheating model (see Sec. III).
- `NO_INTERPOLATION`: do not interpolate electron/muon energy densities and FTQED corrections (enabled by default to save time). Incompatible with the use of logarithmic FTQED corrections.
- `NO_MUONS`: disable muon contributions to total energy density and matter potentials.
- `NO_NUE_ANNIHILATION`: disable contribution from $\nu\bar{\nu} \leftrightarrow e^+e^-$ processes to collision integrals.
- `TESTSPEED`: compute a speed test with a timing of the first 1000 derivatives.

XI. PYTHON SCRIPTS

Together with the Fortran code, `FortEPiaNO` includes some python tools that can be used to facilitate the preparation of input files for the Fortran code, or to read the outputs of the code and produce plots. In order to use them, you only have to make sure that your command line executes the commands from the main `FortEPiaNO` directory or that the `python/` folder is inside your `PYTHONPATH`.

a. prepareIni.py

This script, as the name suggests, helps to create a `.ini` file, that you can use as an argument for the Fortran executable. You can use the script via command line (use e.g. "`python python/prepareIni.py -h`" from the main `FortEPiaNO` folder, it will show the syntax and all the available options) or from a different python script, importing the relevant functions. In such case you will have to import and use some of the internal functions, such as:

```
from prepareIni import setParser, getIniValues, writeIni
parser = setParser()
pargs = parser.parse_args(
    [
        "myfile.ini",
        "myOutputFolder/",
        "3nu",
        "--default_active=VLC",
    ]
)
values = getIniValues(pargs)
writeIni(pargs.inifile, values)
```

b. fortepianoOutput.py

This script contains useful functions to read the output files, process them and produce plots. Depending on the content of the selected output folder and on the passed options, the script will generate additional files in teh same output folder (for example, those for running `PArthENoPE`) and prepare some standard plots. As in the previous case, you can use the functions in two ways: via command line [31] (see "`python python/fortepianoOutput.py -h`") or importing them from other python scripts, for example you can plot the energy density evolution from the output in `myOutputFolder/` using:

```
import matplotlib.pyplot as plt
from fortepianoOutput import FortEPiaNORun
run = FortEPiaNORun("myOutputFolder/")
try:
    print("I got: %s" % run.Neff)
except AttributeError:
    print("The run failed. Cannot read Neff")
else:
    run.plotEnergyDensity()
    plt.savefig("energyDensity.pdf")
```

c. tests.py

Utility to verify that all the python functions implemented in the other two files work properly. Run `python python/tests.py` to test all the functions and methods independently. Some tests will fail if the Fortran code has not been executed with the default `ini/explanatory.ini`.

ACKNOWLEDGMENTS

We thank J.J. Bennett and G. Buldgen for helping spotting some issue in the treatment of finite temperature corrections and for support in implementing the higher order correction terms, and M. Escudero Abenza for useful comments and suggestions.

This software is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme, under the Marie Skłodowska Curie grant agreements No 796941 (ENCORE, until March 2020) and 754496 (FELLINI, starting October 2020).

[1] S. Gariazzo, P. F. de Salas, and S. Pastor, *Thermalisation of sterile neutrinos in the early Universe in the 3+1 scheme with full mixing matrix*, *JCAP* **07** (2019) 014, [`arXiv:1905.11290`].

[2] P. F. de Salas and S. Pastor, *Relic neutrino decoupling with flavour oscillations revisited*, *JCAP* **07** (2016) 051, [`arXiv:1606.06986`].

[3] A. Mirizzi, N. Saviano, G. Miele, and P. D. Serpico, *Light sterile neutrino production in the early universe with dynamical neutrino asymmetries*, *Phys.Rev.D* **86** (2012) 053009, [arXiv:1206.1046].

[4] N. Saviano, A. Mirizzi, O. Pisanti, P. D. Serpico, G. Mangano, and G. Miele, *Multi-momentum and multi-flavour active-sterile neutrino oscillations in the early universe: role of neutrino asymmetries and effects on nucleosynthesis*, *Phys.Rev.D* **87** (2013) 073006, [arXiv:1302.1200].

[5] G. Mangano, G. Miele, S. Pastor, and M. Peloso, *A Precision calculation of the effective number of cosmological neutrinos*, *Phys.Lett.B* **534** (2002) 8–16, [astro-ph/0111408].

[6] S. Gariazzo, C. Giunti, M. Laveder, Y. F. Li, and E. M. Zavanin, *Light sterile neutrinos*, *J.Phys.G* **43** (2016) 033001, [arXiv:1507.08204].

[7] N. Fornengo, C. Kim, and J. Song, *Finite temperature effects on the neutrino decoupling in the early universe*, *Phys.Rev.D* **56** (1997) 5123–5134, [hep-ph/9702324].

[8] J. J. Bennett, G. Buldgen, M. Drewes, and Y. Y. Wong, *Towards a precision calculation of the effective number of neutrinos $N_{\rm eff}$ in the Standard Model I: The QED equation of state*, *JCAP* **03** (2020) 003, [arXiv:1911.04504].

[9] J. J. Bennett et al., *in preparation*, .

[10] A. Dolgov, S. Hansen, and D. Semikoz, *Nonequilibrium corrections to the spectra of massless neutrinos in the early universe*, *Nucl.Phys.B* **503** (1997) 426–444.

[11] D. N. Blaschke and V. Cirigliano, *Neutrino Quantum Kinetic Equations: The Collision Term*, *Phys.Rev.D* **94** (2016), no. 3 033009, [arXiv:1605.09383].

[12] B. H. McKellar and M. J. Thomson, *Oscillating doublet neutrinos in the early universe*, *Phys.Rev.D* **49** (1994) 2710–2728.

[13] K. Enqvist, K. Kainulainen, and M. J. Thomson, *Stringent cosmological bounds on inert neutrino mixing*, *Nucl.Phys.B* **373** (1992) 498–528.

[14] N. F. Bell, R. R. Volkas, and Y. Y. Y. Wong, *Relic neutrino asymmetry evolution from first principles*, *Phys.Rev.D* **59** (1999) 113001, [hep-ph/9809363].

[15] P. F. de Salas, M. Lattanzi, G. Mangano, G. Miele, S. Pastor, and O. Pisanti, *Bounds on very low reheating scenarios after Planck*, *Phys.Rev.D* **92** (2015), no. 12 123534, [arXiv:1511.00672].

[16] https://computation.llnl.gov/casc/odepack/odepack_home.html.

[17] A. Hindmarsh and L. L. Laboratory, *ODEPACK, a Systematized Collection of ODE Solvers.* Lawrence Livermore National Laboratory, 1982.

[18] K. Radhakrishnan and A. C. Hindmarsh, *Description and use of LSODE, the Livermore solver for ordinary differential equations*, .

[19] Note that a logarithmic spacing will introduce biases in the numerical calculation and that the final numbers of the neutrino energy density and effective number may be artificially increased.

[20] J. Lesgourgues and T. Tram, *The Cosmic Linear Anisotropy Solving System (CLASS) IV: efficient implementation of non-cold relics*, *JCAP* **09** (2011) 032, [arXiv:1104.2935].

[21] N. Kaiser, *Clustering in real space and in redshift space*, *Monthly Notices of the Royal Astronomical Society* **227** (jul, 1987) 1–21.

[22] "Newton-Cotes quadrature formula." Encyclopedia of Mathematics. http://www.encyclopediaofmath.org/index.php?title=Newton%E2%80%93Cotes_quadrature_formula&oldid=22842.

[23] W. Romberg, *Vereinfachte numerische integration*, *Norske Vid. Selsk. Forh.* **28** (1955) 30–36.

[24] For simplicity, we assume the same numerical value for both the relative and absolute tolerance. The algorithm will always match the most stringent of the two requirements.

[25] S. Gariazzo, C. Giunti, M. Laveder, and Y. F. Li, *Model-Independent $\bar{\nu}_e$ Short-Baseline Oscillations from Reactor Spectral Ratios*, *Phys.Lett.B* **782** (2018) 13–21, [arXiv:1801.06467].

[26] P. F. de Salas, D. V. Forero, C. A. Ternes, M. Tórtola, and J. W. F. Valle, *Status of neutrino oscillations 2018: 3 σ hint for normal mass ordering and improved CP sensitivity*, *Phys. Lett. B* **782** (jul, 2018) 633–640, [arXiv:1708.01186].

[27] M. Tanabashi et al., *Review of Particle Physics*, *Phys.Rev.D* **98** (2018) 030001.

[28] O. Pisanti et al., *PArthENoPE: Public Algorithm Evaluating the Nucleosynthesis of Primordial Elements*, *Comput.Phys.Commun.* **178** (2008) 956–971, [arXiv:0705.0290].

[29] R. Consiglio, P. F. de Salas, G. Mangano, G. Miele, S. Pastor, and O. Pisanti, *PArthENoPE reloaded*, *Comput.Phys.Commun.* **233** (2018) 237–242, [arXiv:1712.04378].

[30] J. Erler and S. Su, *The Weak Neutral Current*, *Prog.Part.Nucl.Phys.* **71** (2013) 119–149, [arXiv:1303.5522].

[31] Note that this method limits significantly the possible applications because it does not allow to perform further actions.