

PRIMAT (*PRImordial MATter*)

PRIMAT is a *Mathematica* code which computes the abundances of elements at the end of the big-bang nucleosynthesis (BBN).

It can be downloaded by registering at <http://www2.iap.fr/users/pitrou/primat.htm>

Version 0.2.2 (23/06/2023)

Short description

*The implementation follows the presentation of the companion paper Pitrou, Coc, Uzan, Vangioni, Physics Reports, 04, (2018) 005.

All equation numbers, when non specified, refer to this companion paper, in its arXiv version (arXiv:1801.08023).

*It is based on a previous Fortran code written by Alain Coc.

*The user can modify several parameters which are in the preamble of the code :

a) The nuclear reaction rates involved in the nuclear reaction network are tabulated in an external file which can be easily modified.

b) The corrections to the weak-interaction reactions ($n + \nu \leftrightarrow p^+ + e^-$ and its related reactions), can be turned on and off with booleans.

The details of these corrections are given in the companion paper.

c) Cosmological parameters, that is essentially the number of baryons (Ω_b) and the number of neutrino generations N_ν can also be modified.

*The numerical integration is performed in two steps. First the cosmology and the thermodynamics of the plasma are solved,

and then the nuclear reactions are computed, the backreaction of the latter on the former being negligible (see companion paper).

*The results are given as a set of interpolating functions of time for the abundances of individual species.

If one is interested in final abundances only, these are also directly accessible by evaluating these functions at final time.

*A very basic Monte-Carlo exploration of uncertainties is provided at the end of the code. It is used in associated example notebooks located in the 'Example' folder.

For each reaction, an uncertainty factor variance is provided and it is possible to run the code with these uncertainty parameters taking random values according to the variances (see [Coc et al.

2014]). A parallelization is possible for this Monte-Carlo exploration and the analysis of the results can be output in an external file and analyzed separately.

*Several examples and applications are gathered in the 'Examples' folder.

*Neutrino decoupling is handled using output files generated by the (non-public) Python code NEVO [Froustey et al. 2020].

*An option allows to use only a reduced network of equations (weak interactions and 12 nuclear reactions), resulting in much faster results, though slightly different for Li7 results.

Basic usage

*In the Evaluation menu, select 'Evaluate notebook'. If asked the question 'evaluate initialization cells first?', answer no.

Then Mathematica proceeds in evaluating all cells, in order, and it should reach the end of the notebook (with the plots and results) in less than one minute.

*If the user erases the precomputed weak rates which are precomputed and stored in the 'Interpolation' folder, or asks for these rates to be re-precomputed, this can take considerably longer, typically one hour.

*Furthermore, when PRIMAT-Main.nb (this notebook) is opened and saved in Mathematica, the cells which are 'initialization cells' are saved into the file PRIMAT-Main.m.

This *.m file contains all the important definitions, functions and variables which can be loaded from another notebook to perform BBN computations and analysis of results.

The 'Examples' Folder contains several typical applications which work exactly like that. First it loads all necessary definitions stored in PRIMAT-Main.m and then it performs a few useful computations for each selected example.

Preamble

Information

Authors

This code is written and maintained by Cyril Pitrou¹ in collaboration with Alain Coc², J.-P. Uzan³ and E. Vangioni⁴.

Neutrino decoupling computations have been performed by Julien Froustey⁵ with NEVO (non-public Python code).

^{1,3,4,5}*Institut d'Astrophysique de Paris (CNRS)*

98 bis Boulevard Arago

75014 Paris, France

²former CSNSM (CNRS, IN2P3)

Orsay, France

emails and homepages :

-pitrou@iap.fr, <http://www2.iap.fr/users/pitrou>

-uzan@iap.fr

-vangioni@iap.fr

-alain.coc@csnsm.in2p3.fr

-froustey@iap.fr

Dates and versions

In V0.2.0:

- New dpg rate (reaction $d + p \leftrightarrow \text{He3} + g$) in new reaction file BBNRatesAC2021.dat.
- Reaction $\text{Be7}(d,p)$ is now tabulated in this external file and not given analytically.
- Decoupling of neutrinos with oscillations (precomputed with NEVO, see [Froustey et al. 2020]).
- Constants reevaluated with Particle Data Group 2020 data. Main change is G_{Newt} .
- QED corrections (pair production) to some nuclear rates included (see [Pitrou&Pospelov 2020]).
- An option to use only reduced nuclear network has been added.
- QED at order (e^3) included in plasma thermodynamics (reduces N_{eff} by 0.001, see [Bennett et al. 2019]).
- Cosmological parameters are set to the last column of table 2 of [Planck 2018 VI].

In V0.2.1:

- New dpg rate (reaction $d + p \leftrightarrow \text{He3} + g$) in new reaction file BBNRatesAC2022.dat.
- Rate taken from Moscoso et al. 2021

In V0.2.2:

- small modification to make it compatible with Mathematica 13.

In[1]:=

Date[]
[date]

Out[1]= {2023, 6, 23, 23, 39, 17.259668}

Mathematica version used :

In[2]:=

\$Version
[version actuelle]

Out[2]= 13.2.1 for Mac OS X ARM (64-bit) (January 27, 2023)

GPL

```
In[3]:= (* Copyright (C) 2018- Cyril Pitrou, Alain Coc *)
          \[constante C\]

(* This program is free software; you can redistribute it and/or
   modify it under the terms of the GNU General Public License as
          \[général\]
   published by the Free Software Foundation; either version 2 of
   the License, or (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
   General Public License for more details.
   \[général\]

   You should have received a copy of the GNU General Public License
          \[général\]
   along with this program; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place-Suite 330, Boston, MA 02111-1307,
   USA.

*)
```

Bibliography

Hereafter we use the following shorthands for the references cited.

BBN review with PRIMAT description

[Pitrou et al. 2018] C. Pitrou, A. Coc, J.-P. Uzan, E. Vangioni, Physics Reports, 04, (2018) 005, 1801.08023.

[Pitrou et al. 2021] C. Pitrou, A. Coc, J.-P. Uzan, E. Vangioni, MNRAS, ... , 2011.11320.

BBN references

[Bennett et al. 2019] J. J. Bennett, G. Buldgen, M. Drewes, Y. Y. Y. Wong, JCAP **03** (2020) 003, 1911.04504 .

[Bernstein 1989] J. Bernstein, L. S. Brown, G. Feinberg, Rev. Mod. Phys **61** (1989).

[Brown&Sawyer] L. S. Brown, R. F, Sawyer, Phys. Rev. **D63**, 083503 (2001) astro-ph/0006370.

[Cambier et al.] J. L. Cambier, J. R. Primack, and M. Sher, Nucl. Phys. **B209**, 372 (1982).

[Coc et al. 2012] A. Coc, S. Goriely, Y. Xu, M. Saimpert and E. Vangioni, Astrophys. J. **744** (2012) 1107.1117.

[Coc et al. 2014] A. Coc, J.-P. Uzan and E. Vangioni JCAP 1410 (2014) 050 1403.6694.

[Czarnecki et al. 2004] Czarnecki, Marciano and Sirlin, Phys Rev. D **70**, 093006 hep-ph/0406324.

[Dicus et al.] D. A. Dicus, E. W. Kolb, A. M. Gleeson, E. C. G. Sudarshan, V. L. Teplitz, and M. S. Turner, Phys. Rev. **D26**, 2694 (1982).

[Esposito et al. 1999] S. Esposito, G. Mangano, G. Miele and O. Pisanti, Nucl. Phys. **B540**, 3 (1999)

astro-ph/9808196.

[Froustey&Pitrou 2019] J. Froustey, C. Pitrou, Phys. Rev. **D101** (2020) 043524, 1912.09378

[Grohs et al. 2012] E. Grohs, G. M. Fuller, C. T. Kishimoto, M. W. Paris, and A. Vlasenko, astro-ph/1512.02205.

[Horowitz] astro-ph/0109209 (for weak-magnetism definition and related computations).

[Horowitz&Li] C.J. Horowitz, G. Li, astro-ph/9908219

[Heckler 1994] A. F. Heckler, Phys. Rev. **D49** 611 (1994).

[Ivanov 2012] A. N. Ivanov, M. Pitschmann, N. I. Troitskaya Phys. Rev. **D88**, 073002 (2013) hep-ph/1212.0332.

[Kernan] P. J. Kernan, PhD thesis, *Two astroparticle physics problems; solar neutrinos, and primordial ^4He* . Ohio State University (1993).

[Lopez&Turner 1998] R. E. Lopez and M. S. Turner, Phys. Rev. **D59**, 103502 (1999) astro-ph/9807279.

[Lopez et al. 1997] R. E. Lopez, M. S. Turner and G. Gyuk, Phys. Rev. **D56**, 3191 (1997), astro-ph/9703065.

[Mangano et al. 2001] G. Mangano, G. Miele, S. Pastor, M. Peloso, Phys.Lett. **B534** (2002) 8-16, astro-ph/0111408.

[Mangano et al. 2005] G. Mangano, G. Miele, S. Pastor, T. Pinto, O. Pisanti, P. D. Serpico, Nucl.Phys. **B729** (2005) 221-234 hep-ph/0506164.

[ParthENoPE] O. Pisanti et al. Comp. Phys. Com **178** 956 (2008), 0705.0290. Update 2017: arXiv:1712.04378.

[Pitrou&Pospelov 2020] C. Pitrou and M. Pospelov, Phys Rev. C. **102** (2020) 015803, 1904.07795 .

[Seckel 1993] D. Seckel, hep-ph/9305311.

[Sirlin 1967] A. Sirlin Phys. Rev. 164, Vol **5**, p1767 (1967).

[Moscoso 2021] J. Moscoso, R. S. Souza, A. Coc, C. Iliadis, Astrophysical Journal, 923:15 (2021).

Other references. Incomplete neutrino decoupling :

[Froustey et. al. 2020] J. Froustey, C. Pitrou, C. Volpe, JCAP **12** (2020) 015, 2008.01074.

[Hannestad 1995], S. Hannestad, J. Madsen, Phys. Rev. **D52**, 1754 (1995), astro-ph/9506015.

[Gnedin 1997] N. Y. Gnedin, O. Y. Gnedin, AP. J. **509**, 11-15 (1998), astro-ph/9712199.

Reaction rates

They are listed in references of [Coc. et. al 2012] (see Table 4). We use the following acronyms :

NACRE (Angulo et al. 1999)

NACRE II (Xu et al. 2010, 2011)

DAACV04 (Descouvemont et al. 2004)

ILCCF10 (Iliadis et al. 2010)

CF88 (Caughlan& Fowler 1988)

MF89 (Malaney& Fowler 1989)

Boy93 (Boyd et al. 1993)
 Bal95 (Balbes et al. 1995)
 Hei98 (Heil et al. 1998)
 Rau94 (Rauscher et al. 1994)
 Des99 (Descouvemont 1999)
 Bea01 (Beaumel et al. 2001)
 Tan03 (Tang et al. 2003)
 Wan91 (Wang et al. 1991)
 Efr96 (Efros et al. 1996)
 Wie87 (Wiescher et al. 1987 Wiescher,Harms,Goerres,Thielemann& Rybarcyk ApJ 316 (1997) 162
 1001.2053)
 Bar97 (Bardayan& Smith 1997)
 Koe91 (Koehler& Graff 1991)
 And06 (Ando et al. 2006)
 Ser04 (Serpico et al. 2004)
 Wag69 (Wagoner 1969)
 Has09 (Hashimoto et al.2009)
 Wie89 (Wiescher et al.1989)
 FK90 (Fukugita& Kajino 1990)
 Bru91 (Brune et al.1991)
 Bec92 (Becchetti et al.1992)
 Iga95 (Igashira et al.1995)
 Cyb08 (Cyburt& Davids 2008)
 Miz00 (Mizoi et al. 2000)
 Nag06 (Nagai et al. PRC 74 (2006) 025804 AC2010)
 Has09c (Hashimoto et al.PLB 674 (2009) 27)
 FK90 (Fukugita& Kajino,PRD 42 (1990) 4251)
 Rau94 (C. Rauscher et al.ApJ 429 (1994) 499)
 Men12 (C Mendes et al.PRC 86 (2012) 064321)
 Tang03 (Tang et al.PRC 67 (2003) 015804)
 Bar97C (Bardayan& Smith PRC 56 (1997) 1647)
 Kaw91 (Kawano,Fowler,Kavanagh,Malaney ApJ 372 (1991) 1-7)
 Cam08 (Camargo et al.Phys.Rev.C 78,034605 (2008))
 Ili16 (Iliadis et al. 2016)

Numerical values

[Planck 2015 XIII] Ade et al. A.&A. 594, A13 (2016).
 [Planck 2018 VI] A.&A. (2020). We use TT+TE+EE+lowE+Lensing+BAO parameters (last column of table 2).
 [PDG17] Particle Data Group 2017.
 [PDG20] Particle Data group 2020.

Options

Directory set up

We set the directory. This cell is not an initialization cell, and is not save in the .m file. However for some systems, when evaluating examples which load PRIMAT-Main.m, it is necessary to turn this cell into an initialization cell so that it is part of the PRIMAT-Main.m file.

```
In[4]:= SetDirectory[NotebookDirectory[]]
      |alloue répertoire |répertoire de notebook
```

```
Out[4]= /Users/pitrou/Dropbox/iap/notebooks/PRIMAT2022
```

To check what is the directory of your notebook :

```
In[5]:= Print["The current Directory is ", Directory[]]
      |imprime |répertoire |répertoire
```

```
The current Directory is /Users/pitrou/Dropbox/iap/notebooks/PRIMAT2022
```

Numerical options

■ Miscellaneous options

```
In[6]:= $InterpolateAnalytics = True;
      |vrai
```

It is slightly faster to interpolate the analytic expressions of nuclear reaction rates. Setting \$InterpolateAnalytics to True is recommended.

```
In[7]:= $HistoryLength = 10;
      |longueur d'historique
```

This is to avoid *Mathematica* to store too many results in memory. Only the past \$HistoryLength results are kept in computer memory (this is standard Mathematica option).

```
In[8]:= $PaperPlots = False;
      |faux
$ResultsPlots = False;
      |faux
```

If \$PaperPlots is set to True, the most important plots are constructed and they are output in pdf in the 'Plots' subfolder.

Unless interested in reproducing the plots of the companion paper, this should be set to False to avoid any loss of time in the code.

If \$ResultsPlots is True the results for the abundances are plotted at the end and exported in pdf files. Similarly, to avoid loss of time this should be set to False.

■ Numerical precision

```
In[10]:= $CompileNDSolve = True;
      |vrai
```

If \$CompileNDSolve is set to True (advised), then the differential equation solver uses compiled

functions. This is slightly faster.

```
In[11]:= $BDFOrder = 2;
```

Order of numerical scheme for Backward Differentiation Scheme (BDS) integration.

-Order 1 works well but is slow.

-Order 2 (advised) is faster. Higher orders result in numerical crash.

```
In[12]:= PrecisionNDSolve = 2; (* Put 2 here is the best choice *)
      mets en fichier
```

PrecisionNDSolve is a precision parameter used in the differential equation solver. It is tuned such that 0 gives reasonable results, 1 gets very good results, and 2 gets excellent results and 3 gets super dupper precise results (10^{-5} precision).

```
In[13]:= AccuracyNDSolve := 15 + PrecisionNDSolve;
```

We slave AccuracyNDSolve to PrecisionNDSolve. This is roughly the number of digits of precision behind the dot, so increasing it will lead to always better results but this can crash if the accuracy required is too strong.

```
In[14]:= NTemperaturePoints = 1200; (*1000 is enough*)
```

Number of points in discretization of temperature between the highest temperature (10^{12} K) and the lowest ($\sim 10^{7.5}$ K). 1000 is enough.

Sampling is performed with NTemperaturePoints + 1 points. Spacing is performed logarithmically, with Log_{10} .

```
In[15]:= InterpOrder = 3;
```

Order of polynomials used in interpolations of reaction rates. Usually with Spline functions.

```
In[16]:= $FastPENRatesIntegrals = True;
      vrai
```

If \$FastPENRatesIntegrals is set to True, it uses a Simpson method for numerical integrals. Otherwise it uses the Mathematica function NIntegrate which is more accurate (adaptative refinement of integral) but much slower.

```
In[17]:= $PENRatesIntegralsPoints = 300; (*200 is enough *)
```

In case \$FastPENRatesIntegrals is set to True, \$PENRatesIntegralsPoints is the number of points used to perform the numerical integrals. 200 is enough. 400 is ultra precise.

```
In[18]:= SetOptions[SelectedNotebook[], PrintPrecision → 8]
      alloue options  [notebook sélectionné]  précision d'impression
```

We increase the number of digits which are displayed

BBN options

■ Nuclear rates

Most nuclear reactions rates are tabulated in a separate file. The rest of the rates are given as analytic fits.

The external file loaded with all reactions definitions and rates is given by the name `TabulatedReactionsFile`.

From version 0.1.2 onward, we remove the possibility of setting `NumberNuclearReactions`.

The set of nuclear reactions is given by the external files plus all the analytic forms inside the code.

These must be modified if one intends to use a different set or number of nuclear reactions.

```
In[19]:= TabulatedReactionsFile = "BBNRatesAC2022.dat";
```

We can decide to keep only a subset of the equations by specifying the maximum nuclear mass. The default value is `Infinity`, meaning that we do not cut the network.

```
In[20]:= MaximumNuclearMass = Infinity;  
          |  
          |infinity
```

We can also reduce the network to its 12 first reactions. This is enough for the first light elements. It consists in the first 12 reactions of the tabulated reactions (those in the file `TabulatedReactionsFile`).

Furthermore, if `$ReducedNetwork` is selected, no extra analytic reactions are added so that we really have only a total of 12+1 reactions (because we also have weak interactions).

```
In[21]:= $ReducedNetwork = False;  
          |  
          |faux
```

```
In[22]:= If[$ReducedNetwork, SmallNuclearNetworkSize = 12];  
          |  
          |si
```

■ Monte-Carlo uncertainty estimation options

```
In[23]:= $RandomNuclearRates = False;  
          |  
          |faux  
  
$MaxVariationRate = 1000;
```

If `$RandomNuclearRates` is `True`, then each time we build the equations of the nuclear network, we generate rates with a log normal distribution according to the 'f' specified (see Eq. 4.5 of [Coc et al. 2014] for definition of f).

We limit f to the values $1/\$MaxVariationRate < f < \$MaxVariationRate$.

■ Rescaling of some rates

It is possible to rescale some rates by choosing their rescaling factor below. First, the option `$RescaleSomeRates` must be set to `True` otherwise the rescaling factors are ignored.

```
In[25]:= $RescaleSomeRates = False;
          faux
```

For each reaction whose rate we want to rescale, we need to know the exact name of the reaction. The names of reactions are collected in tables below in the core of the notebook. For a reaction of the type $a + b \rightarrow c + d$, the name is “abTOcd”. However one needs to know the order of the initial particles and of the final particles, and to put correctly the photons if they appear. Hence it is recommended not to guess the name of the rate, but to properly read it from the Tables below located in the section “Collecting all reaction rates”.

For instance the reaction $\text{He3} + t \rightarrow \text{He4} + d$ (we use d for H2 and t for H3 and a for He4) has name He3tTOHe4d. The factor to modify it is therefore He3tTOHe4dFactor.

As an example, we show how to introduce a rescaling factor for the $d + p \rightarrow \text{He3} + \gamma$, $d + d \rightarrow \text{He3} + n$, and $d + d \rightarrow \text{H3} + p$ reactions hereafter.

If we want to vary other rates, one needs only to set the numerical value of their associated factors .

```
In[26]:= If[$RescaleSomeRates,
          si
          dpT0He3gFactor = 1;
          ddT0He3nFactor = 1;
          ddT0tpFactor = 1;
          ];
```

Note that this procedure is only valid for variation of rates which are given as tables in the external file. If the user wants to vary a rate whose form is analytic, it is sufficient to add a factor “Reaction-Factor” in the definition of the analytic forward rate (forward[T9_]:=). It is then advised to write here (so that all modifications can be turned on and off without forgetting that some rates have been modified)

```
If[$RescaleSomeRates,
  ReactionFactor=100;,
  ReactionFactor=1;
]
```

We recall that the syntax for If/then/else is

```
If[Condition,
  commandtrue;,
  commandfalse;
]
```

■ QED corrections (pair production) to nuclear rates

If this options is set to True, then we rescale some rates due to pair productions in the final state. See [Pitrou&Pospelov 2020].

```
In[27]:= $NuclearRatesQEDCorrections = True; (* recommended value is True *)
          |vrai                               |vrai
```

Weak rates options

Since the weak interaction rates, that is the rates of weak reactions of the type $n + \nu \leftrightarrow p + e$ and associated reactions, depend only on temperature, these can be computed once and for all so as to explore the dependence in cosmological parameters or other parameters.

If `$RecomputeWeakRates` is set to `True`, the code recomputes the weak rates. Otherwise it reads them from files previously stored. If the file does not exist, it recomputes the rates.

If `$ParallelWeakRates=True` this is done using parallelization over the various CPUs that Mathematica can detect.

```
In[28]:= $RecomputeWeakRates = False;
          |faux
          $ParallelWeakRates = False;
          |faux
```

There are several booleans corresponding to the different types of corrections which can be considered in these weak rates.

The name of the file used for storing the rates is built out of these booleans.

Since the corrections do not add linearly, there are in principle many choices of corrections, but the only meaningful choices are those without any corrections and those with all corrections included. Or maybe those with just one correction added, so as to check its amplitude.

```
In[30]:= $RadiativeCorrections = True; (*Recommended True*)
          |vrai                               |vrai

          $ResummedLogsRadiativeCorrections = True; (*Recommended True*)
          |vrai                               |vrai

          $RelativisticFermiFunction = True; (*Recommended True*)
          |vrai                               |vrai
```

1) If `$RadiativeCorrections` is set to `True`, we use Coulomb and radiative corrections (see section III.E of companion paper. The corrections are implemented in Eqs. 101 and 104.).

If `$ResummedLogsRadiativeCorrections` is set to `True` we use Eq. 15 of [Czarnecki et al 2004] which amounts to resumming some higher order radiative corrections, and this is also Eq. B35 of the companion paper.

If `$ResummedLogsRadiativeCorrections` is `False` we use simply Eq. 7 of the same reference, which corresponds to Eq. 103 and B30 of the companion paper.

If `$RelativisticFermiFunction` is `True` we use the relativistic Fermi function (Eq. 5 in [Ivanov 2012]), which corresponds to Eq. 100 of the companion paper. Otherwise we use the standard non-relativistic Fermi function, given by Eq. 99 of the companion paper.

```
In[33]:= $RadiativeThermal = True; (*Recommended True*)
          _vrai                      _vrai
$CorrectionBremsstrahlung = True; (*Recommended True*)
          _vrai                      _vrai
```

2) If `$RadiativeThermal` set to `True`, the thermal radiative corrections are taken into account. The first expressions date back from [Dicus et al.]. However other expressions were derived subsequently in [Lopez&Turner 1998], [Cambier et al.] and [Esposito et al. 1999] among other references. [Kernan] pointed typos and compared the differing results of [Cambier et al.] and [Dicus et al.]. These were correctly computed in [Brown&Sawyer].

In the companion paper, the thermal radiative corrections are given by Eq. 108 with the various contributions given by Eqs. 109-113. However, it is easier to compute the first contribution of 108 using Eqs. 109 (with definition B41), but to compute the second and third contributions of 108 using Eqs. B50-B51.

We showed in companion paper that Bremsstrahlung corrections need also to be added to be fully consistent with [Brown&Sawyer]. This is controlled by the boolean `$CorrectionBremsstrahlung`. If `$RadiativeThermal` is set to `True`, then these bremsstrahlung corrections (corresponding to Eqs. 107a and 107b in companion paper with the definitions B48-B49) are also incorporated if `$CorrectionBremsstrahlung` is `True`.

```
In[35]:= $FiniteNucleonMass = True; (*Recommended True*)
          _vrai                      _vrai
```

3) If `$FiniteNucleonMass` is set to `True`, we take into account the finite mass of nucleons by keeping corrections in $1/M_n$ in the collision integrals of the weak rates. Our method is described in the companion paper and differs from earlier literature. There is a suboption for these finite mass corrections which is

```
In[36]:= $CoupledFMandRC = True; (*Recommended True*)
          _vrai                      _vrai
```

If `$CoupledFMandRC` is `False`, finite mass corrections are computed from the Born results with no radiative null temperature corrections. If this is set to `True`, the finite mass corrections are applied to the rates on which the Coulomb and radiative corrections are taken into account. `True` is the advised value.

The expressions implemented are Eqs. 114 when `$CoupledFMandRC` is `False`, with the definition B23. If `$CoupledFMandRC` is `True`, then we use Eqs. 115 with the Fermi and radiative corrections corresponding to the above choices for radiative corrections.

4) Mass shifts due to QED plasma effects are **ALREADY** taken into account in thermal radiative corrections.

However it is possible to turn the option `$QEDMassShift` to `true` to check how this affects the rate

when taken individually.

Apart to satisfy this curiosity, **this option should be set to False** in all cases.

```
In[37]:= $QEDMassShift = False;
          _vrai
```

Plasma physics options

```
In[38]:= $QEDPlasmaCorrections = True; (*Recommended True*)
          _vrai
          _vrai
          $CompleteQEDPressure = True; (*Recommended True*)
          _vrai
          _vrai
          $QEDOe3 = True; (*Recommended True*)
          _vrai
          _vrai
```

If \$QEDPlasmaCorrections is set to True, the QED corrections to the pressure and the energy density are taken into account. This affects for instance the expansion rate via the Friedmann equation. Expressions can be found in [Lopez&Turner 1998], [Mangano et al. 2001] or in [Heckler 1994]. See also the companion paper where Eqs. 48-52 are used inside Eq. 55 to modify the entropy and Eq. 58 to modify the energy density.

If \$CompleteQEDPressure is False, the subdominant term is ignored. Otherwise it is included. We checked it is so subdominant that it does not change the results.

If \$QEDOe3 is True, then order e^3 corrections to the Plasma (to P and ρ) are included, Following [Bennett et. al. 2019].

```
In[41]:= $RecomputePlasmaCorrections = False;
          _faux
```

Since the QED effects depend on temperature, they need to be computed only once for all and they are stored on a file. But we can force the recomputation of these corrections by setting \$RecomputePlasmaCorrections to True;

Neutrino incomplete decoupling options

```
In[42]:= $IncompleteNeutrinoDecoupling = True; (*Recommended True*)
          _vrai
          _vrai
```

If \$IncompleteNeutrinoDecoupling is set to True, the entropy transfer from e^\pm annihilations to neutrinos is taken into account. Indeed if we consider the details of the decoupling of neutrinos, it is found that decoupling is incomplete by the time electrons and positrons annihilate into photons. This results in a slight overheating of neutrinos and cooling of the electrons/photons plasma. The advised value for \$IncompleteNeutrinoDecoupling is True.

```
In[43]:= $NEVO = True; (* Recommended True *)
          _vrai
          _vrai
          $TrueTve = True; (* Recommended True *)
          _vrai
          _vrai
          $SpectralDistortions = True; (* Recommended True *)
          _vrai
          _vrai
```


NEVO's results would then be inconsistent because they were computed without chemical potential. μ_{OverTv} is also noted ξ_v in the companion paper. See section VI.C.

Check Incompatible options

If Plasma QED corrections are not used, then the full expression for the Pressure QED correction cannot be used, nor the order e^3 correction.

```
In[49]:= If[Not@$QEDPlasmaCorrections && $CompleteQEDPressure,
  [si [négation]
    $CompleteQEDPressure = False];
  [faux]

If[Not@$QEDPlasmaCorrections && $QED0e3, $QED0e3 = False];
[si [négation] [faux]
```

If NEVO results are not used for neutrino decoupling, then we cannot have the true v_e temperature, nor spectral distortions

```
In[51]:= If[Not@$NEVO && $TrueTve, $TrueTve = False];
[si [négation] [faux]

If[Not@$NEVO && $SpectralDistortions, $SpectralDistortions = False];
[si [négation] [faux]
```

Initial definitions

Temperature eras

We choose to split the BBN numerical calculations in three eras.

- 1) First the high temperature era between $T_{\text{start}} = 10^{11} \text{ K}$ and T_{middle} . Only neutrons and protons abundances are tracked and this is ruled by weak interactions.
- 2) The intermediary era, between T_{middle} and T_{18} , where only a small network of reactions (17 nuclear reactions plus weak interactions or less if the reduced network is used) is used.
- 3) A low temperature region, between T_{18} and T_{end} , where T_{end} is usually slightly lower than 10^8 K , typically $T_{\text{end}} = 6 \times 10^7 \text{ K}$, and where all nuclides and reactions are considered.

So we have $T_{\text{start}} > T_{\text{middle}} > T_{18} > T_{\text{f}}$.

```
In[53]:= Kelvin = 1;
Tstart = 1011 Kelvin;
Tmiddle := 0.9999 * 1010 Kelvin;
T18 := 1.25 * 109 Kelvin;
Tend = 6. * 107 Kelvin;
```

Temperature sampling

We choose to sample temperature starting from 10^{12} K . This is interesting to check high T behaviour of some effects.

```
In[58]:= Ti = 1012 Kelvin;
Tf = 107 Kelvin;
LogTi = 1. Log10[Ti];
           |logarithme en ba
LogTf = 1. Log10[Tf];
           |logarithme en ba
```

We first build the list of LogT points (ListLogT) and then the list of T points (ListT).

```
In[62]:= ListLogT = Sort@DeleteDuplicates@Join[{10.},
           |trie |supprime répétitions |joins
           Table[i, {i, LogTf, LogTi, (LogTi - LogTf) / NTemperaturePoints}]];
           |table
ListT = 1. × 10ListLogT;
```

```
In[64]:= ListTRange[T1_, T2_] := Module[
           |module
           {len = Length@ListT, imindown, imaxup, Tmin = Min[T1, T2], Tmax = Max[T1, T2]},
           |longueur |minimum |maximum
           imindown = Max[1, -1 + Position[ListT, SelectFirst[ListT, # > Tmin &]]][1, 1]];
           |maximum |position |sélectionne premier
           imaxup = Min[len, Position[ListT, SelectFirst[ListT, # ≥ Tmax &]]][1, 1]];
           |minimum |position |sélectionne premier
           ListT[[imindown ;; imaxup]]
           ]
```

ListTRange is a function to select a sublist in this list of temperature, according to a range of temperature which makes sure to have either the points on the boundary or at least one point beyond (to avoid problems with interpolating functions).

If $T = 10^{10}$ is not in the list, we add it to avoid problems with interpolations of reactions rates which all start at 10^{10} and below.

Constants of Physics

■ cgs system

We use cgs system with Kelvin. (For instance an erg is $1 \text{ g cm}^2 \text{ s}^{-2}$).

By definition these are set to one in these units. Any change of system of units can be made by modifying these variables only.

For instance if we want to use the m/kg/s system we need only put $\text{cm} = 0.01$ and $\text{gram} = 0.001$ below.

As a check, final results should not depend on these conventions since abundance rates are dimensionless.

```
In[65]:= second = 1;
cm = 1;
gram = 1;
```

When taking values from the kg/m/s system, we use the factors


```
In[68]:= kg = 103 gram;
meter = 102 cm;
km = 103 meter;
Joule = kg meter2 / second2; (* This gives 107 ergs *)
DensityUnit = gram / cm3;
Hz = 1 / second;
```

```
In[74]:= Giga = 109;
Mega = 106;
Kilo = 103;
```

■ Fundamental constants

```
In[77]:= kB = 1.3806488 × 10-23 Joule / Kelvin; (* Boltzmann constant in J/K *)
cLight = 2.99792458 × 108 * meter / second; (* speed of light in cm/s *)
hbar =  $\frac{6.62607015}{2 \pi}$  10-34 Joule second;
Avogadro = 6.02214076 × 1023;
```

When using masses of particles, we use eV and MeV that we convert in the cgs system

```
In[81]:= eV = 1.602176634 × 10-19 Joule;
keV = Kilo eV;
MeV = Mega eV;
GeV = Giga eV;
```

Interactions constants

```
In[85]:= GN = 6.67430 × 10-11 meter3 / kg / second2; (* Gravitation constant PDG2020 *)
GF = 1.1663787 × 10-5 / (GeV)2; (* Fermi Constant PDG2020*)
gA = 1.2756;
(* Axial current constant of structure of
the nucleons. It was 1.2723(+23) in PDG2016 *)
(* However post 2002 data suggest
1.2755(11) as advised by William Marciano*)
(* We now use PDG2020 value 1.2756(13) which
is much more consistent with the neutron lifetime *)
```

```
In[88]:= fWM = 3.7058 / 2 (*1.853*); (* Weak magnetism see 1212.0332*)
radiusproton = 0.841 × 10-15 meter (*arXiv:1212.0332*)
```

```
Out[89]= 8.41 × 10-14
```

```
In[90]:= fWM
```

```
Out[90]= 1.8529
```

f_{WM} is the weak magnetism constant. See Eq. [Horowitz&Li] for definition with the value given by its Table 1.

Note that all expressions in [Seckel 1993] seem to have a factor 2 difference. That is all interaction rates involving the weak magnetism in [Seckel 1993] are underestimated by a factor 2. Expressions in [Lopez et al. 1997] seem correct however.

```
In[91]:=  $\alpha_{FS} = 1 / 137.03599911; (* \text{ Fine structure constant } = e^2 / (4\pi) *)$ 
```

■ Particle masses

Throughout, masses stand always for mc^2 so that they are in fact energies. This avoids putting unnecessary c^2 factors

```
In[92]:= me = 0.510998950 MeV; (*PDG2020*)
mn = 939.56542052 MeV; (*PDG2020*)
mp = 938.27208816 MeV; (*Idem *)
Q = mn - mp; (* Mass difference between neutrons and protons *)
mNucleon = mn;

mW = 80.379 GeV; (* Mass of the W Boson. *)
mZ = 91.1876 GeV;
```

The energy difference between neutron and proton in MeV gives 1.29333205(51) [PDG 2020].

```
In[99]:= Q / MeV
```

```
Out[99]= 1.2933324
```

■ Cosmology constants

h is the Hubble rate in units of 100 km/s/Mpc.

```
In[100]:= pc = 3.0856777807  $\times 10^{16}$  meter; (* The parsec *)
Mpc = Mega pc;
H0 = 100 h km / second / Mpc; (* Hubble constant today *)
H100 = 100 km / second / Mpc;
(*Fake Hubble rate given by 100 km/s/Mpc so that h = H0/H100 *)
```

We define two critical densities. One for the actual Hubble rate, and one for the rate at 100km/s/Mpc.

In[104]:=

$$\rho_{\text{crit}} = \frac{3.}{8 \pi G N} (H0)^2 (* \text{ in g cm}^{-3} \text{ by construction} *)$$

$$\rho_{\text{crit100}} = \frac{3.}{8 \pi G N} (H100)^2 (* \text{ in g cm}^{-3} \text{ by construction} *)$$

Out[104]=

$$1.8783414 \times 10^{-29} \text{ h}^2$$

Out[105]=

$$1.8783414 \times 10^{-29}$$

Neutron life time (Particle Data Group 2020).

In[106]:=

```
(* 880.2+-1.1s was previous value from PDG2017 *)
(* We then used 1712.05663 Section 11 which
   includes recent 2017 measurements and gave 879.5+-0.8*)
(* Finally PDG2020 gave 879.4+-0.6 and we now take this value*)
Meanτneutron := 879.4 second;
στneutron := 0.6 second;
τneutron = Meanτneutron;
```

Cosmological Parameters

Neutrinos generations and possible neutrino degeneracy (neutrino chemical potential).

In[109]:=

```
NeutrinosGenerations := 3.;
ξν := If[$DegenerateNeutrinos, μ0verTν, 0];
```

In[111]:=

$$\rho_{\text{FD}}[c_]=\frac{1}{2\pi^2}\int_0^{\text{Infinity}}\frac{y^3}{(e^{y-c}+1)}dy;$$

$$n_{\text{FD}}[c_]=\frac{1}{2\pi^2}\int_0^{\text{Infinity}}\frac{y^2}{(e^{y-c}+1)}dy;$$

$$\rho_{\text{FDNonDegenerate}}=\rho_{\text{FD}}[0];$$

Information

In[114]:=

$$\text{Series}\left[\frac{\rho_{\text{FD}}[c]+\rho_{\text{FD}}[-c]}{2\rho_{\text{FDNonDegenerate}}},\{c,0,4\}\right];$$

In[115]:=

$$\eta_{\nu}[c_]=\frac{(n_{\text{FD}}[c]-n_{\text{FD}}[-c])}{(2\text{Zeta}[3]/\pi^2)};$$

$$\text{Series}[\eta_{\nu}[c],\{c,0,3\}];$$

développement en série entière

Effective number of neutrinos generation due to chemical potential

(this is different from N_{eff} which takes into account also QED and incomplete neutrino decoupling)

In[117]:=

$$\text{Nneu} := \text{NeutrinosGenerations} * \frac{\rho_{\text{FD}}[\xi \nu] + \rho_{\text{FD}}[-\xi \nu]}{2 \rho_{\text{FDNonDegenerate}}}$$

CMB temperature today

In[118]:=

$$\begin{aligned} \text{TCMB0} &:= 2.7255 \text{ Kelvin}; \\ \sigma \text{TCMB0} &:= 0.0006 \text{ Kelvin}; (* [\text{Planck 2015 XIII}] *) \end{aligned}$$

Temperature of CMB today in Kelvin. We consider the case where QED effects are ignored or taken into account. This is the implementation of (the inverse of) Eq. 56 in companion paper.

In[120]:=

$$\begin{aligned} \text{FourOverElevenQED} &:= 1 / \left(\frac{11}{4} - \frac{25 \alpha_{\text{FS}}}{8 \pi} + \text{If} \left[\text{\$QED0e3}, \frac{10 \alpha_{\text{FS}}^{(3/2)}}{\pi^2} \text{Sqrt} \left[\frac{\pi}{3} \right], 0 \right] \right); \\ \text{FourOverElevenNoQED} &:= \frac{4}{11}; \\ \text{FourOverEleven} &:= \\ &\quad \text{If}[\text{\$QEDPlasmaCorrections}, \text{FourOverElevenQED}, \text{FourOverElevenNoQED}]; \\ \text{Tv0} &= (\text{FourOverEleven})^{1/3} \text{TCMB0}; \end{aligned}$$

In[124]:=

$$\left(\frac{1}{\text{FourOverEleven}} \right)^{(1/3)}$$

Out[124]=

1.3998958

Temperature of neutrinos today is lower than photons because they decoupled earlier and electron/positron annihilation has only reheated photons. This leads to the famous ratio of 4/11 between the T^3 of the neutrinos and photons. However, since decoupling is slightly incomplete, this in principle should be corrected like in [Mangano et al. 2005] or [Grohs et al. 2012].

Additionally, there is another source of modification to this 4/11 ratio which comes from the QED corrections to the plasma thermodynamic quantities (modification of pressure and energy density and thus of entropy). Taking the high temperature modification leads to the correction added above in the variable FourOverEleven. See e.g. Eq. 41 of [Lopez&Turner 1998] and/or the companion paper (Eq. 56). The effect of incomplete neutrino decoupling is considered further below.

Hubble rate in units of 100 km/s/Mpc.

In[125]:=

$$\text{h} := 0.6766; (* -0.0042 *) (* [\text{Planck 2018 VI TT+TE+EE+lowE+Lensing+BAO}] *)$$

Baryons and Cold Dark Matter density fraction. This is $\Omega_b h^2$ and $\Omega_c h^2$.

In[126]:=

```

Meanh2Ωb0Planck = 0.022425; (* [Planck 2018 VI, Last column of
                                dernier
                                table 2 (and checked from the Planck Legacy Markov chains)] *)
σh2Ωb0Planck = 0.000136; (* Standard deviation *)
(* Planck 2018 VI states 0.00014 but from the
   Legacy Markov chains we infer the value 0.000136 *)

```

In[128]:=

```

Meanh2Ωb0 = Meanh2Ωb0Planck;
σh2Ωb0 = σh2Ωb0Planck;
h2Ωb0 = Meanh2Ωb0;

```

In[131]:=

```

ReSetCosmology := (
  Meanh2Ωb0 = Meanh2Ωb0Planck;
  NeutrinosGenerations = 3;
);

```

In[132]:=

```

Meanh2Ωc0 = 0.11933; (* [Planck 2018 VI, last column of Fig. 2] *)
σh2Ωc0 = 0.00091;
h2Ωc0 = Meanh2Ωc0;

```

Cosmological constant fraction Ω_Λ . Obtained by summing baryons and cold dark matter, given that radiation is negligible today.

This is just for information and it is not used since the cosmological constant is totally negligible for its influence in the expansion rate during BBN.

In[135]:=

$$1 - (h2\Omega b0 + h2\Omega c0) / h^2$$

Out[135]=

0.69034764

Density of photons and neutrinos

The Black Body constant is defined as

In[136]:=

$$a_{BB} = \frac{\pi^2}{15 \hbar^3 (c \text{light})^5}$$

Out[136]=

 2.3167357×10^{28}

Energy density and number density of CMB today (See appendix A1 in companion paper)

In[137]:=

```

ρCMB0 := aBB (kB TCMB0)4 ; (* in g cm-3 *)

nCMB0 :=  $\frac{2 \text{Zeta}[3]}{\pi^2 \hbar^3 (c \text{light})^3}$  (kB TCMB0)3

```

We recover the number of photons per cubic centimeter (410) :

In[139]:=

```
nCMB0
```

Out[139]=

```
410.72667
```

The fraction of energy content due to photons is simply

In[140]:=

```
Ωγ0 := ρCMB0 / ρcrit;
```

For neutrinos, we must take into account the temperature of neutrinos today, the number of neutrinos, and the fact that they are fermions.

See companion paper for details.

In[141]:=

```
Ων0 := Nneu *  $\frac{7}{8}$  * (FourOverEleven)4/3 Ωγ0;
```

The contribution to the energy content is obtained by the ratio between energy densities and critical density. We check that today it is around 0.1%.

In[142]:=

```

 $\frac{\Omega_{\gamma 0} h^2}{h^2 \Omega_{b 0}}$ 

```

Out[142]=

```
0.0011027756
```

Density of baryons

In[143]:=

```

ma = 931.494061 MeV; (* Audi2012 *)
He4overma = 4.0026032541; (* Audi2012 *)
H10verma = 1.00782503223; (* Audi2012 *)

```

The atomic mass, the Helium4 mass (in units of atomic mass) and Hydrogen mass (in units of atomic mass).

In[146]:=

```

xHe4 = 0.2471 ; (* Chemical composition at the end of BBN. In
                                dans
                                principle one should account for He4 produced by stars...*)
xH1 = 1 - xHe4;
mbaryon0 = (xH1 H10verma + xHe4 He4overma / 4) ma;

```

In[149]:=

ma

Out[149]=

0.001492418

This is the (average) mass of baryons today (that is of nucleons), taking into account that part is in Hydrogen and part in Helium. We use the current chemical composition with 24.75 % of Helium but this is subject to controversy. Indeed the abundance of baryons is measured with CMB, and thus refers to an epoch ($z \sim 1100$) where the composition was the same as the one at the end of BBN. See Eqs. C5 C6 in companion paper.

In[150]:=

mbaryon0 / ma**ma / mbaryon0**

Out[150]=

1.0060523

Out[151]=

0.99398413

In[152]:=

$$\frac{\left(\frac{\text{He40verma}}{4} - \text{H10verma}\right)}{\text{H10verma}}$$

% * X_{He4}

Out[152]=

-0.0071185161

Out[153]=

-0.0017589853

The number density of baryons, that is of nucleons is then given by the baryons mass density divided by the average mass of baryons.

In[154]:=

$$\rho_{B0} := h^2 \Omega_b^0 * \rho_{crit100};$$

In[155]:=

$$nbaryons0 := \frac{\rho_{B0}}{(mbaryon0 / (c_{light})^2)}$$

The ratio between baryons number and photons number is by definition the η parameter and its value for the parameters chosen is

In[156]:=

$$\frac{nbaryons0}{n_{CMB0}}$$

1 / %

Out[156]=

 $6.1388136 \times 10^{-10}$

Out[157]=

 1.6289793×10^9

It is convenient to define the ratio between $\Omega_b h^2$ and this η parameter.

In[158]:=

$$\Omega_{bh20ver\eta} := \frac{n_{CMB0}}{\rho_{crit100}} \frac{mbaryon0}{(c_{light})^2}$$

In[159]:=

$\Omega_{bh20ver\eta}$
% / Meanh2**Ω**b0Planck

Out[159]=

$$3.652986 \times 10^7$$

Out[160]=

$$1.6289793 \times 10^9$$

In[161]:=

1 / $\Omega_{bh20ver\eta}$

Out[161]=

$$2.7374865 \times 10^{-8}$$

The η parameter is then obtained from the baryon density fraction as

In[162]:=

$$\eta_{factor} := \frac{h^2 \Omega_{b0}}{\Omega_{bh20ver\eta}}$$

Baryons density is obtained from its valued today scaled by dilution (no thermal effects, so it is only the energy density due to rest mass of baryons).

In[163]:=

$$\begin{aligned} \rho_B[av_] &:= \frac{\rho_{B0}}{av^3}; \\ n_B[av_] &:= \frac{n_{baryons0}}{av^3}; \end{aligned}$$

For nuclear reactions, the mass density of baryons is in fact a number density of species multiplied by the atomic mass (see appendix C1 of companion paper for a detailed discussion). This differs slightly from the mass density of baryons and we take this into account.

If `$CorrectBaryonsEnergyDensityinBBNRRates` is set to `False`, then we use the baryons density naively in nuclear rates.

Otherwise we take into account that the baryons density is in fact the number density times the atomic mass as explained in App. C1 of the companion paper.

In[165]:=

```
$CorrectBaryonsEnergyDensityinBBNRRates = True;
ρBForBBN[av_] :=
  ρB[av] × If[$CorrectBaryonsEnergyDensityinBBNRRates, ma / mbaryon0, 1];
(* This is Eq. C8 of the companion paper *)
```

Distribution functions

Basic Fermi - Dirac (FD) and Bose - Einstein (BE) functions. x here $1/(k_B T)$.

In[167]:=

```

FD[EoverT_] =  $\frac{1}{(\text{Exp}[EoverT] + 1)}$ ; (* Fermi Dirac Distribution *)

FD[Energy_, x_] =  $\frac{1}{(\text{Exp}[x \text{ Energy}] + 1)}$ ;

BE[EoverT_] =  $\frac{1}{(\text{Exp}[EoverT] - 1)}$ ; (* Bose Einstein Distribution *)

BE[Energy_, x_] =  $\frac{1}{(\text{Exp}[x \text{ Energy}] - 1)}$ ;

(* For neutrinos with a chemical potential *)
[boucle for]
FDv[Energy_,  $\phi$ _, x_] =  $\frac{1}{(\text{Exp}[x \text{ Energy} - \phi] + 1)}$ ;

```

Derivatives of FD wrt to energy.

In[172]:=

```

FDp[Energy_, x_] = [derivative]  $D\left[\frac{1}{(\text{Exp}[x \text{ Energy}] + 1)}, \text{Energy}\right]$ ;

```

Customized Mathematica tools

This function NP displays a certain number of digits for a given real number

In[173]:=

```

NP[number_] := NumberForm[number, 8]
[forme de nombres]

```

This function displays a table in grid form, that is with lines between the entries

In[174]:=

```

MyGrid[Table_List] := Grid[Table, Frame → All]
[liste] [grille] [table] [cadre] [tout]

```

This function performs interpolation on a list of points (x, f(x)) to the required order.

In[175]:=

```

MyInterpolation[Tab_List] :=
  Interpolation[Tab, InterpolationOrder → InterpOrder];

(* Does not work to interpolate the log
   of rates because it fails when rates vanish !!!*)
MyInterpolationLog[Tab_List] :=
  Function[{x}, Exp[Interpolation[{#[[1]], Log[#[[2]]]} & /@ Tab,
    InterpolationOrder → InterpOrder][x]]];

$InterpolateLogRate = False;

MyInterpolationRate[Tab_List] :=
  If[$InterpolateLogRate, MyInterpolationLog[Tab], MyInterpolation[Tab]]

```

Tools to avoid too small numbers in numerics.

MyChop chops small numbers and replaces them by 0.

In[179]:=

```

MyChop[el_?NumericQ] := (Chop[el, $MinMachineNumber]);
SetAttributes[MyChop, Listable];

```

Redefinition of Set to allow to set values to quantities already set

In[181]:=

```

MySet[Hold[expr_], value_] := (expr = value);
MySetDelayed[Hold[expr_], value_] := (expr := value);

```

Personal simple integral with second order polynomial interpolation (Simpson method).

In[183]:=

```

TableSimpsonC = Compile[
  {{a, _Real}, {b, _Real}, {Np, _Integer}}, With[{h = 1. (b - a) / Np, n2 = Np / 2},
    With[{h3 = h / 3.}, Join[{a, h3}, Table[{a + 2. j h, 2 h3}, {j, 1, n2 - 1}],
      Table[{a + (2. j - 1) h, 4 h3}, {j, 1, n2}], {{b, h3}}]]],
  CompilationTarget → "C", "RuntimeOptions" → "Speed"];

```

Generic compilation of a function and of its integration.

In[184]:=

```
MyCompile[LV_List, Body_] := Compile[LV, Evaluate[Body],
  _liste _compile _évalue
  "RuntimeOptions" → "Speed", CompilationTarget → "C",
  _options de durée d'exécution _cible de compilation _constante C
  CompilationOptions → {"InlineExternalDefinitions" → True},
  _options de compilation _vrai
  RuntimeAttributes → {Listable}]
  _attributs de durée d'exécution _listable
```

Compilation of a scalar product.

In[185]:=

```
V1dotV2 = Compile[{{V1, _Real, 1}, {V2, _Real, 1}},
  _compile _nombre réel _nombre réel
  V1.V2, CompilationTarget → "C", "RuntimeOptions" → "Speed"];
  _cible de compilation _con... _options de durée d'exécution
```

Compiled version of the Simpson integral

In[186]:=

```
IntegrateFunction[fun_, pemin_, pemax_, Np_] :=
  With[{interv = (pemax - pemin) / (Np), tab = TableSimpsonC[pemin, pemax, Np]},
  _avec
    V1dotV2[tab[[All, 2]], MyChop[fun[tab[[All, 1]]]]];
    _tout _tout
```

A function to import an external file and which returns an error and quits if the file does not exist.

In[187]:=

```
SafeImport[args__] := Module[{out}, out = Catch[Check[Import[args],
  _module _renvoi... _vérifie _importe
  Print["File ", {args}[[1]], " not found. Quitting Kernel."];
  _imprime _fichier
  Throw[$Failed];, Import::nffil]];
  _jette _échoué _importe
  If[out === $Failed,
  _si _échoué
    Beep[];
    _bip
    MessageDialog["Impossible to load a file. Kernel has been aborted."];
    _dialogue avec message
    Quit[]];
    _quitte noyau de système
  out]
```

Tools for plots. Some useful grid of ticks

In[188]:=

```

MyFrameTicksLog = {{Automatic, Automatic},
  {{Log[10^8], "10^8"}, {Log[10^8.5], "10^8.5"}, {Log[10^9], "10^9"},
  {Log[10^9.5], "10^9.5"}, {Log[10^10], "10^10"}, {Log[10^10.5], "10^10.5"},
  {Log[10^11], "10^11"}, {Log[10^11.5], "10^11.5"}}, Automatic}};

MyFrameTicks =
  {{Automatic, Automatic}, {{10^8, "10^8"}, {10^8.5, "10^8.5"}, {10^9, "10^9"},
  {10^9.5, "10^9.5"}, {10^10, "10^10"}, {10^10.5, "10^10.5"},
  {10^11, "10^11"}, {10^11.5, "10^11.5"}}, Automatic}};

```

Loading neutrino decoupling results from NEVO

We load the file which store the output of NEVO, and perform needed interpolations.

In[190]:=

```

If[$NEVO,
  FileNEVO = If[$QEDPlasmaCorrections,
    If[$QED0e3, "NEVOPRIMAT", "NEVOPRIMAT_QED2"], "NEVOPRIMAT_NoQED"];
  TableNEVO = Import["Interpolations/" <> FileNEVO <> ".csv"];
  FileGrid = "Interpolations/NEVOGrid.csv";
  TableGrid = Flatten@Import[FileGrid];
]

```

If QED effects are taken into account, we import results from a NEVO simulation incorporating the same QED corrections to the pressure and the energy density (but not to the weak rates of the reactions relevant from neutrino decoupling, as it results in a higher order effect).

In NEVO, the “time” variable (proxy for the scale factor) is $x_{\text{NEVO}} \propto m_e a$, while in PRIMAT we have used the notation $x = 1/(k_B T)$.

To avoid possible confusion we explicitly use x_{NEVO} below.

See e.g. [Froustey et al. 2020] for historical definitions of x, y, z in the context of neutrino decoupling ($y_{\text{NEVO}} = a p$ and $z_{\text{NEVO}} = a T_\nu$), but this notation is standard in neutrino decoupling literature.

We perform interpolations :

In[191]:=

```

If[$NEVO,

```

```

(*First column of NEVO.cvs file is xNEVO*)
  _premier
xNEVO = TableNEVO[All, 1];
  _tout

(* Second column is zNEVO*)
zNEVO = TableNEVO[All, 2];
  _tout

(* By definition the temperature of photons is related by : *)
TγNEVO = me * zNEVO / xNEVO / kB;

Tγmax = First@TγNEVO;
  _premier
Tγmin = Last@TγNEVO;
  _dernier

(*Columns 3, 4,
and 5 are the effective temperatures of nu_e, nu_mu and nu_tau *)
Tve = TableNEVO[All, 3] * me / xNEVO / kB;
  _tout
Tνμ = TableNEVO[All, 4] * me / xNEVO / kB;
  _tout
Tντ = TableNEVO[All, 5] * me / xNEVO / kB;
  _tout

(* We build interpolation functions of these neutrino temperatures,
as function of photon/plasma temperature *)
TveOFTγNEVO = Interpolation@Transpose[{TγNEVO, Tve}];
  _interpolation _transpose
TνμOFTγNEVO = Interpolation@Transpose[{TγNEVO, Tνμ}];
  _interpolation _transpose
TντOFTγNEVO = Interpolation@Transpose[{TγNEVO, Tντ}];
  _interpolation _transpose

(*We build the ratio of the nu_e temperature with the photon temperature,
as a function of the photon temperature *)
TveOverTγOFTγNEVO = Interpolation@Transpose[{TγNEVO, Tve / TγNEVO}];
  _interpolation _transpose

(* We avoid issues when outside of the interpolation range *)
TveOverTγOFTγNEVOSafe[T_] := If[T ≥ Tγmax, 1,
  _si
  If[T ≥ Tγmin, TveOverTγOFTγNEVO[T], TveOverTγOFTγNEVO[Tγmin]] ];
  _si

(* We also build the same type of ratio,
but with the average temperature of neutrinos *)
TvAverageOverTγOFTγNEVO = Interpolation@
  _interpolation
  Transpose[{TγNEVO, (1 / 3 * (Tve^4 + Tνμ^4 + Tντ^4)) ^ (1 / 4) / TγNEVO}];
  _transpose
TvAverageOverTγOFTγNEVOSafe[T_?NumericQ] := If[T ≥ Tγmax, 1, If[T ≥ Tγmin,
  _expression num... _si _si

```

```

TvAverageOverTγ0FTγNEVO[T], TvAverageOverTγ0FTγNEVO[Tγmin]]];

(* xNEVO as a function of the inverse neutrino temperature. *)
xNEVOofβ = Interpolation@Transpose[{me / (kB Tve), xNEVO}];
           |interpolation |transpose

(* xNEVOofT=Interpolation@Transpose[{TγNEVO,xNEVO}];*)
           |interpolation |transpose

zNEVOofT = Interpolation@Transpose[{TγNEVO, zNEVO}];
           |interpolation |transpose

(* The heating function  $\mathcal{N}$ , is the 6th column *)
HeatingNNEVO = TableNEVO[All, 6];
               |tout

 $\mathcal{N}$ NEVO = Interpolation@Transpose[{me / (kB TγNEVO), HeatingNNEVO}];
           |interpolation |transpose

]

```

We also interpolate (in 2D) the spectrum (in xNEVO and in comoving momentum $y = a p$). Only the spectrum of ν_e is given in the NEVO.cvs file, and it starts from the 7 – th column.

The grid in comoving momentum (variable $y = a p$), is stored in the file NEVOGrid which has been loaded in FileGrid

In[192]:=

```

If[$NEVO && $IncompleteNeutrinoDecoupling && $SpectralDistortions,
  |si

  ymin = First@TableGrid;
        |premier
  ymax = Last@TableGrid;
        |dernier
  GS = Length[TableGrid];
        |longueur
  Ny = GS;
  LogxNEVO = Log@xNEVO;
            |logarithme
  Logxmin = First@LogxNEVO;
            |premier
  Logxmax = Last@LogxNEVO;
            |dernier
  LinearGrid = DeleteDuplicates[
                |supprime répétitions
    Join[Table[y, {y, ymin, ymax, (ymax - ymin) / (Ny - 1)}], {ymax}]];
        |joins |table
  Lengthxrange = Dimensions[TableNEVO][[1]];
                |dimensions

  (* Here the 7 is because the
    |ici
    interpolation data starts at the 8-th column *)
  (* Since the grid in x is logarithmic,
    and the grid in y is irregular (Gauss-Laguerre Quadrature),

```

```

Mathematica cannot perform a 2D interpolation. To bypass this problem,
we first interpolate the spectra for each xNEVO*)
LinearTable2DNue = Table[Interpolation[
  Transpose[{TableGrid, TableNEVO[[xi, Table[6 + i, {i, 1, GS}]]}]]][
  LinearGrid], {xi, 1, Lengthxrange}];
(* And we then use these interpolation to resample
the spectrum at each xNEVO, on a linear grid of y.*)
(* The Log(xNEVO) grid is linearly spaced, and we now have a 2D
linearly spaced grid that we can interpolate in two dimensions *)
dfNuelogxy = Interpolation@Flatten[
  Table[{Re@LogxNEVO[[lxi]], LinearGrid[[yi]], LinearTable2DNue[[lxi, yi]],
    {yi, 1, Ny}, {lxi, 1, Lengthxrange}}, 1];
(* If points are outside interpolation we put 0 *)
dfNuelogxySafey[Lx_, y_] := If[y ≥ ymin && y ≤ ymax, dfNuelogxy[Lx, y], 0];
dfNuelogxySafexSafey[Lx_, y_] := If[Lx ≥ Logxmin,
  If[Lx ≤ Logxmax, dfNuelogxySafey[Lx, y], dfNuelogxySafey[Logxmax, y]], 0];
(* Finally, after all this, we obtain a function which depends
on xNEVO (the scale factor) and the comoving momentum y*)
dfNuexy[x_, y_] := dfNuelogxySafexSafey[Log@x, y];
]

```

We check the final and initial spectrum. We recognize the Fig. 2 of [Mangano et al. 2005] (with oscillations, and only the v_e)

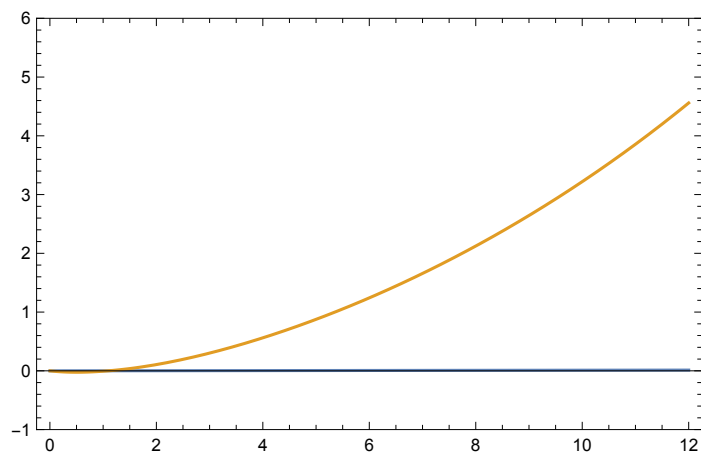
In[193]:=

```

If[$NEVO && $IncompleteNeutrinoDecoupling,
  Plot[{100 * dfNuexy[0.03, y], 100 * dfNuexy[20, y]}, {y, ymin, 12},
  PlotRange → {-1, 6}, Frame → True], FrameLabel → {"y = a p", "100  $\delta f_e$ "}]

```

Out[193]=



What is stored by NEVO was the deviation with respect to a pure Fermi-Dirac distribution and the neutrino temperature. (Slightly inconsistent to put a chemical potential in neutrinos here, since decoupling was computed without. Similarly it was computed without extra relativistic degrees of freedom).

In the following, remember that the energy is in units of the electron mass.

In[194]:=

```

If[{$NEVO && $IncompleteNeutrinoDecoupling,
  _si
  Tvemax = First@Tve;
    _premier
  Tvemin = Last@Tve;
    _dernier
   $\beta_{vemin} = m_e / (k_B T_{vemax});$ 
   $\beta_{vemax} = m_e / (k_B T_{vemin});$ 

  (* This is the difference between the distorted spectrum,
  and the Fermi-Dirac spectrum at the  $\nu_e$  temperature. *)
  (* We recall that the  $\nu_e$  temperature is defined as the temperature
  of the fermion gas which has the same energy density. *)
   $\delta FDv_{Raw}[en_, \phi_, \beta_{ve\_}] := With[{\mathbf{x} = x_{NEVO} of \beta[\beta_{ve}]},
    _avec
    With[{\mathbf{y} = en * \mathbf{x}}, \frac{(1 + dfNuexy[\mathbf{x}, \mathbf{y}])}{Exp[\mathbf{y} - \phi] + 1} - \frac{1}{Exp[\beta_{ve} en - \phi] + 1}]];$ 
    _avec

  (* The negative energy corresponds to Pauli Blocking factors,
  hence the extra minus sign*)
  (* We also treat the cases which are outside the grid of
  interpolations using that spectra are frozen once decoupled *)
  Clear[ $\delta FDv$ ];
  _efface
   $\delta FDv[en\_?NumericQ, \phi_, \beta_{ve\_}] :=$ 
    _expression numérique ?
     $\delta FDv_{Raw}[en, \phi, \beta_{ve}] /;$   $en \geq 0 \ \&\& \ \beta_{ve} \geq \beta_{vemin} \ \&\& \ \beta_{ve} \leq \beta_{vemax};$ 
   $\delta FDv[en\_?NumericQ, \phi_, \beta_{ve\_}] :=$ 
    _expression numérique ?
     $-\delta FDv_{Raw}[-en, \phi, \beta_{ve}] /;$   $en < 0 \ \&\& \ \beta_{ve} \geq \beta_{vemin} \ \&\& \ \beta_{ve} \leq \beta_{vemax};$ 
   $\delta FDv[en\_?NumericQ, \phi_, \beta_{ve\_}] := 0 /;$   $\beta_{ve} < \beta_{vemin} \ || \ \beta_{ve} > \beta_{vemax};$ 
    _expression numérique ?
  SetAttributes[ $\delta FDv$ , Listable];,
  _alloue attributs _listable
  (* If $NEVO is False, then we fall back to these
  _si _faux
  definitions. We set spectral distortion to be zero in that case.*)
  Clear[ $\delta FDv$ ];
  _efface
   $\delta FDv[en\_?NumericQ, \phi_, \beta_{ve\_}] := 0;$ 
    _expression numérique ?
  SetAttributes[ $\delta FDv$ , Listable];
  _alloue attributs _listable
]

```

It is this function δFDv that we use later to take into account the effect of distortions in n-p conversions.

Plasma thermodynamics

Thermodynamic integrals

Defined in appendix A1 of companion paper. These are the integrals needed to obtain the thermodynamic quantities of FD or BE distributions and correspond to Eqs. A5 in companion paper.

In[195]:=

```
Clear[Imn]
Efface

Imn[sgn_][m_, n_][x_] := NIntegrate[Intègre numérique  $\frac{(pe^2 + x^2)^{((m-1)/2)} pe^{(n+1)}}{(\text{Exp}[\sqrt{pe^2 + x^2}] + \text{sgn})}$ ,
{pe, 0, Infinity}, Method → {Automatic, "SymbolicProcessing" → 0}]
infini méthode automatique

ImnT[sgn_][m_, n_][T_] := Imn[sgn][m, n][ $\frac{me}{kB T}$ ]

(* Interpolations *)
ImnI[sgn_][m_, n_] := ImnI[sgn][m, n] =
Interpolation@Table[{ $\frac{me}{kB Tv}$ , Imn[sgn][m, n][ $\frac{me}{kB Tv}$ ]}, {Tv, ListT}]
interpolation table

ImnIT[sgn_][m_, n_][T_] := ImnI[sgn][m, n][ $\frac{me}{kB T}$ ]
```

QED corrections to plasma thermodynamics

QED mass corrections

From this, using Eq. 12 and 13 of [Mangano.et.al 2001] (or Eq. 35 of [Lopez & Turner 1998] for the mass of the electron), we get the modification to the mass of the electron and of the photon. For this we ignore the last term in Eq. 12 of [Mangano.et.al 2001] or Eq. 35 of [Lopez&Turner 1998]. See also companion paper (Eqs. 44 and 46).

The mass shift is expressed in units of the electron mass so as to be dimensionless. So what we define as dme2 is really $\delta(m_e)^2 / (m_e)^2$ and similarly for dmy2 it is $\delta(m_\gamma)^2 / (m_e)^2$

In[200]:=

```

dme2[T_] :=  $\left(\frac{k_B T}{m_e}\right)^2 \left(\frac{2 \pi \alpha F_S}{3} + \frac{4 \alpha F_S}{\pi} \text{ImnT}[1][0, 1][T]\right)$ 
(* Only main part of mass shift *)
dmγ2[T_] :=  $\frac{8 \alpha F_S}{\pi} \text{ImnT}[1][0, 1][T] \left(\frac{k_B T}{m_e}\right)^2$ 

```

We perform interpolations of these mass shifts over the relevant range of temperatures. We store it on disk in the files dme2.dat and dmγ2.dat if they have never been computed.

If they have already been computed we just load the results (unless the Boolean option \$RecomputePlasmaCorrections is set to True).

In[202]:=

```

dme2Tab = Check[Import["Interpolations/dme2.dat", "TSV"],
  |vérifie |importe
  Print["Precomputed data not found. We recompute and store the data."];
  |imprime
  $Failed, Import::nffil];
  |échoué |importe

dmγ2Tab = Check[Import["Interpolations/dmγ2.dat", "TSV"],
  |vérifie |importe
  Print["Precomputed data not found. We recompute and store the data."];
  |imprime
  $Failed, Import::nffil];
  |échoué |importe

```

In[204]:=

```

Timing[
  |chronométrage
  If[dme2Tab == $Failed || dmγ2Tab == $Failed || $RecomputePlasmaCorrections,
    |si |échoué |échoué

    dme2Tab = Table[{T, dme2[T]}, {T, ListT}];
    |table

    dmγ2Tab = Table[{T, dmγ2[T]}, {T, ListT}];
    |table

    Export["Interpolations/dme2.dat", dme2Tab, "TSV"];
    |exporte
    Export["Interpolations/dmγ2.dat", dmγ2Tab, "TSV"];
    |exporte
  ];]

```

Out[204]:=

```
{0.000011, Null}
```

Once having the table of values for the mass shift as a function of temperature, we perform an interpolation

In[205]:=

```
dme2I = MyInterpolation@ToExpression@dme2Tab;
                                     |en expression
dmy2I = MyInterpolation@ToExpression@dmg2Tab;
                                     |en expression
```

We define a function which gives a value for all temperature and not just in the range of the interpolation so as to avoid any numerical problem.

In[207]:=

```
dme2N[T_?NumericQ] := Which[T < Tf, 0, T ≤ Ti, dme2I[T], T > Ti, dme2I[Ti]];
                        |expression num... |quel
dmy2N[T_?NumericQ] := Which[T < Tf, 0, T ≤ Ti, dmy2I[T], T > Ti, dme2I[Ti]];
                        |expression num... |quel
```

We also define these interpolations in terms of the inverse temperature (in units of electron mass, that is the quantity $x = m_e/(k_B T)$)

In[209]:=

```
dme2x[x_] := dme2N[me / (kB x)];
```

We plot the result for illustration purposes (only if option \$PaperPlots is True).

In[210]:=

```
If[$PaperPlots, LogLogPlot[Abs@dme2N[Tv] / Tv^2, {Tv, 10^8, 10^12}, Frame → True,
|si |tracé log-log |valeur absolue |cadre |vrai
    FrameLabel → {"T (K)", "δm_e^2/T^2"}, PlotStyle → {Black, Thickness[0.0035]}]]
|étiquette de cadre |style de tracé |noir |épaisseur
If[$PaperPlots,
|si
    Export["Plots/Plotdme2.pdf", Style[%, Magnification → 1], "PDF"];]
|exporte |style |agrandissement |fonction de densité de probab
```

QED pressure corrections

Pressure corrections are obtained from Eq. 13 of [Heckler 1994] when including only electron mass shift, or Eq. 16 of [Mangano et al. 2001] for both electron mass and photon mass shifts. See also companion paper (around Eqs. 48 and 49) It is made of the dominant term dPa, and the subdominant terms dPb which are the two contributions of Eq. 48 in companion paper.

In[212]:=

$$dPa[T_] := dPa[T] = \frac{\alpha FS}{\pi} (kB T)^4 \left(-\frac{2}{3} \text{ImnT}[1][0, 1][T] - \frac{2}{\pi^2} (\text{ImnT}[1][0, 1][T])^2 \right);$$

For the subdominant contribution we use reduced variables. But contrary to the rest of the code where p stands for p/me here it stands for p/T.

In[213]:=

```

Fdp1dp2 = Compile[{{p1, _Real}, {p2, _Real}, {x, _Real}}, Evaluate[With[
  {e1 =  $\sqrt{p1^2 + x^2}$ , e2 =  $\sqrt{p2^2 + x^2}$ },
   $\frac{\alpha_{FS}}{\pi^3} \frac{x^2 p1^2 p2^2}{p1 p2 e1 e2} \text{Log}\left[\text{Abs}\left[\frac{(p1 + p2)}{(p1 - p2)}\right]\right] \frac{1}{(\text{Exp}[e1] + 1) (\text{Exp}[e2] + 1)}$ 
]], "RuntimeOptions" → "Speed", CompilationTarget → "C"];

Fdp1dp2N[p1_?NumericQ, p2_?NumericQ, x_] := Fdp1dp2[p1, p2, x];

Clear[dPb]
dPb[Tv_] := dPb[Tv] = (kB Tv)^4 With[{x = me / (kB Tv)},
  0.5 NIntegrate[
    Fdp1dp2N[(p1pp2 + p1mp2) / 2, (p1pp2 - p1mp2) / 2, x]
    + Fdp1dp2N[(p1pp2 - p1mp2) / 2, (p1pp2 + p1mp2) / 2, x],
    {p1mp2, 0.0001, Max[20, 20 * x]}, {p1pp2,
    0.0001 + Abs[p1mp2], Max[20, 20 * x] + Abs[p1mp2]}, PrecisionGoal → 4]
];

```

In[217]:=

```

If[$PaperPlots, PlotdPadPb = ListLogLogPlot[
  _si                                     _tracé log-log de liste
  {Table[{Tv, Abs@dPa[Tv] / (kB Tv)^4}, {Tv, ListTRange[10^8.5, 10^11]}],
    _table                               _valeur absolue
    Table[{Tv, Abs@dPb[Tv] / (kB Tv)^4}, {Tv, ListTRange[10^8.5, 10^11]}]},
  _table                               _valeur absolue
  FrameLabel → {"T (K)", "δP / (kB T)^4"}, LabelStyle → {FontSize → 12},
  _étiquette de cadre                  _style d'étiquette    _taille de police de caractères
  FrameTicks → MyFrameTicksLog,
  _graduations de cadre
  PlotStyle → {{Red, Thickness[0.0035]}, {Blue, Dashed, Thickness[0.0035]}},
  _style de tracé                     _rouge _épaisseur      _bleu _en tirets _épaisseur
  Frame → True, FrameStyle → Thickness[0.004],
  _cadre _vrai _style de cadre _épaisseur
  Joined → True, PlotRange → {10^-10, 10^-2}]]
  _joint _vrai _zone de tracé

If[$PaperPlots,
  _si
  Export["Plots/PlotdPadPb.pdf", Style[PlotdPadPb, Magnification → 1], "PDF"];]
  _exporte                               _style                _agrandissement      _fonction de

```

QED of order e^3 . See appendix of [Froustey et al 2020]. Initial formulas from [Bennett et al. 2019].

In[219]:=

```

dPe3[T_] := dPe3[T] = (αFS) ^ (3 / 2) * (4 / 3) Sqrt[2 Pi]
                                     _racine ... _nombre pi
  (kB T)^4 ((ImnT[1][0, 1][T] + ImnT[1][2, -1][T]) / Pi^2) ^ (3 / 2);
                                     _nombre pi

```

The pressure is then obtained (restoring the correct dimensions)

In[220]:=

```

dP[T_] :=
  dP[T] = dPa[T] + If[$CompleteQEDPressure, dPb[T], 0] + If[$QED0e3, dPe3[T], 0]
                                     _si                                     _si

dPI := dPI = Interpolation@Table[{Tv, dP[Tv]}, {Tv, ListT}]
                                     _interpolation    _table

```

We check the high temperature limit, which is given in Eq. 30 of [Lopez&Turner 1998] or Eq. 1 of [Heckler 1994]. See also companion paper.

```

In[222]:=
dPa[10^12] / (kB 10^12)^4
dPb[10^12] / (kB 10^12)^4
dP[10^12] / (kB 10^12)^4
- (5/288) 4 π αFS

Out[222]=
-0.0015919089

Out[223]=
4.1712464 × 10^-9

Out[224]=
-0.0014501471

Out[225]=
-0.0015920354

In[226]:=
dPe3[10^12] / (kB 10^12)^4
αFS^(3/2) 2/9 Sqrt[Pi/3]
      |         |
      |racine|nombre p

Out[226]=
0.00014175759

Out[227]=
0.00014175872

```

QED energy density corrections

Energy density corrections are obtained from the thermodynamic identity $\rho = -P + T dP/dT$. See Eq. 50 of companion paper.

```

In[228]:=
Clear[dρ]
|efface
dρ[T_] := dρ[T] = -dP[T] + T dP[T]

```

In[230]:=

```

If[$PaperPlots, Plotdrho = ListLogLogPlot[
  [si] [tracé log-log de liste]
  {Table[{Tv, Abs@dρ[Tv] / (kB Tv)4}, {Tv, ListTRange[108.5, 1011]}]},
  [table] [valeur absolue]
  FrameLabel → {"T (K)", "δρ / (kB T)4"},
  [étiquette de cadre]
  LabelStyle → {FontSize → 12}, FrameTicks → MyFrameTicksLog,
  [style d'étiquette] [taille de police de ca...] [graduations de cadre]
  PlotStyle → {{Red, Thickness[0.0035]}, {Blue, Dashed, Thickness[0.0035]}},
  [style de tracé] [rouge] [épaisseur] [bleu] [en tirets] [épaisseur]
  Frame → True, FrameStyle → Thickness[0.004],
  [cadre] [vrai] [style de cadre] [épaisseur]
  Joined → True, PlotRange → {10-10, 10-2}]
  [joint] [vrai] [zone de tracé]

If[$PaperPlots,
  [si]
  Export["Plots/Plotdrho.pdf", Style[Plotdrho, Magnification → 1], "PDF"];]
  [exporte] [style] [agrandissement] [fonction de den:

```

QED modified relativistic degrees of freedom

The modified relativistic degrees of freedom (see [Lopez & Turner 1998] for definition) are [see also Eq. 52 of companion paper]

In[232]:=

$$\begin{aligned}
 \text{dgP}[T_] &:= \text{dP}[T] \frac{90}{\pi^2 (kB T)^4}; \\
 \text{dg}\rho[T_] &:= \text{d}\rho[T] \frac{30}{\pi^2 (kB T)^4};
 \end{aligned}$$

We check the high temperature limits (Eq. 54 of companion paper)

In[234]:=

```

dgP[10^12]
(*3dgρ[10^12]*)
-25 αFS
4 π [si] + If[$QED0e3, 20 Sqrt[Pi / 3] αFS3/2 / π2, 0]
  [racin...] [nombre pi]

```

Out[234]=

-0.013223756

Out[235]=

-0.013224937

We interpolate these relativistic degrees of freedom and we store them in a file 'dg.dat'. If this file is already present we do not recompute unless the option \$RecomputePlasmaCorrections is set to True.

In[236]:=

```

dgρdgP = Check[Import["Interpolations/dg.dat", "TSV"],
  _vérifie _importe
  Print["Precomputed data not found. We recompute and store the data."];
  _imprime
  $Failed, Import::nffil];
  _échoué _importe

Timing[If[dgρdgP == $Failed || $RecomputePlasmaCorrections,
  _chrono... _si _échoué

  dgρTab = Table[{T, dgρ[T]}, {T, ListT}];
  _table
  dgPTab = Table[{T, dgP[T]}, {T, ListT}];
  _table

  dgρdgP = {dgρTab, dgPTab};
  Export["Interpolations/dg.dat", dgρdgP, "TSV"];
  _exporte
];]

```

Out[237]=

 $\{7. \times 10^{-6}, \text{Null}\}$

We perform an interpolation in time of the modified relativistic degrees of freedom

In[238]:=

```

dgρI = MyInterpolation@ToExpression[dgρdgP[[1]]];
  _en expression
dgPI = MyInterpolation@ToExpression[dgρdgP[[2]]];
  _en expression

```

We also define functions which are valid everywhere so as to avoid numerical problems (indeed at very low and very large temperature δg_ρ and δg_p are constants).

In[240]:=

```

dgρN[T_?NumericQ] := Which[T < Tf, 0, T ≤ Ti, dgρI[T], T > Ti, dgρI[Ti]];
  _expression num... _quel
dgPN[T_?NumericQ] := Which[T < Tf, 0, T ≤ Ti, dgPI[T], T > Ti, dgPI[Ti]];
  _expression num... _quel

```

We define the relativistic degrees of freedom in function of inverse temperature (in units of electron mass), that is a functions of $x = m_e/(k_B T)$.

In[242]:=

```

dgρx[x_] := dgρN[ $\frac{m_e}{(k_B x)}$ ];
dgPx[x_] := dgPN[ $\frac{m_e}{(k_B x)}$ ];

```

We reproduce Fig. 14 of [Lopez & Turner 1998]

In[244]:=

```

If[$PaperPlots, PlotdPdrho =
  LogLinearPlot[{Abs@dgPN[Tv], Abs@dgPN[Tv], 25 αFS / (4 π)}, {Tv, 10^8.5, 10^11},
    Frame → True, FrameLabel → {"T (K)", "-2δP/P"},
    LabelStyle → {FontSize → 12}, FrameTicks → MyFrameTicks,
    FrameStyle → Thickness[0.004], PlotStyle → {{Thickness[0.004], Red},
    {Blue, Thickness[0.004], Dashing[{0.018}]}, {Black, Thickness[0.003]}}]]

If[$PaperPlots,
  Export["Plots/PlotdPdrho.pdf", Style[PlotdPdrho, Magnification → 1], "PDF"];

```

Entropy and energy density of the plasma

Here, we compute the thermodynamics using thermodynamical equilibrium.

Indeed if we assume total neutrino decoupling then the collision rates inside the electrons/protons/photons plasma are so high that it is always both at thermal and chemical equilibrium.

Furthermore there are so many more photons than baryons today that the chemical potential of electrons and positrons can be ignored. See companion paper for a discussion on the chemical potentials of electrons/positrons.

We have two functions to tabulate.

The first function, gives the extra amount of entropy at high temperature due to electrons and positrons, in units of the entropy of photons.

It is noted S in the companion paper (Eq. 30b). We distinguish the case with and without QED plasma corrections (See Eq. 50 for QED plasma corrections).

In[246]:=

```

DSTNoQED = MyInterpolation@Table[{T, With[{x = me / (kB T)},
  1 +  $\frac{45}{2 \pi^4} \left( \frac{1}{3} \text{Imn}[1][0, 3][x] + \text{Imn}[1][2, 1][x] \right)$  }], {T, ListT}];

DSTQED[Tv_] := (3 dgPN[Tv] + dgPN[Tv]) / 8 + DSTNoQED[Tv];

DST[Tv_] := If[$QEDPlasmaCorrections, DSTQED[Tv], DSTNoQED[Tv]]

DSTN[T_?NumericQ] = Which[T < Tf, 1, T ≤ Ti, DST[T], T > Ti, DST[Ti]];

```

The second function, gives the extra amount of energy density at high temperature due to electrons and positrons, in units of the energy density of photons.

It is noted \mathcal{E} in the companion paper in Eq. 41b. We distinguish the case with and without QED plasma corrections (see Eq. 58 QED plasma corrections).

In[250]:=

```
DρTNoQED = MyInterpolation@
  Table[ $\left\{T, \text{With}\left[\left\{x = m_e / (k_B T)\right\}, \frac{30}{\pi^4} (\text{Imn}[1][2, 1][x])\right]\right\}, \{T, \text{ListT}\}$ ];
DρT[T_] := If[$QEDPlasmaCorrections,  $\frac{d g_N[T]}{2}$ , 0] + DρTNoQED[T];
```

We check that the ratio of entropy long before and long after electron/positrons annihilation is the famous 4/11, possibly corrected by the QED corrections.

In[251]:=

```
DST[10^8] / DST[10^12]
FourOverEleven // N
```

Out[251]=

```
0.36451369
```

Out[252]=

```
0.36451285
```

In[253]:=

```
If[$PaperPlots,
  LogLinearPlot[{DSTNoQED[T] - 1, DρTNoQED[T]},
    {T, 10^8, 10^12}, Frame → True, FrameStyle → Thickness[0.004],
    FrameLabel → {"T (K)", "S-1"}, LabelStyle → {FontSize → 12},
    GridLines → {{me / kB, {Darker@Gray, Thickness[0.005]}}, {}}, PlotStyle →
    {{Red, Thickness[0.0035]}, {Blue, Thickness[0.0035], Dashing[0.01]}}]
If[$PaperPlots,
  Export["Plots/PlotCalScale.pdf", Style[%, Magnification → 1], "PDF"];
```

Incomplete decoupling of neutrinos (method using heating functions)

We now refine to take into account the incomplete decoupling of neutrinos

In the previous PRIMAT version, we were using a fit from [ParthENoPE] for the heating function which characterizes how neutrinos are (slightly) reheating due to incomplete decoupling.

Now, if \$NEVO option is set to True, we use the one computed by NEVO. The heating function $\mathcal{N}(z)$ of Parthenope is found from Eqs. A.24 – A.25 in [ParthENoPE].

In[255]:=

```

Listnl = {-10.21703221236002, 61.24438067531452,
  -340.3323864212157, 1057.2707914654834, -2045.577491331372,
  2605.9087171012848, -2266.1521815470196, 1374.2623075963388,
  -586.0618273295763, 174.87532902234145, -35.715878215468045,
  4.7538967685808755, -0.3713438862054167, 0.012908416591272199};

NParthenope[z_] := If[z ≥ 4, 0, Exp[Plus@@Table[Listnl[[i + 1]] z^i, {i, 0, 13}]]]

If[$NEVO,
  zmin = me / (kB * First@TyNEVO);
  zmax = me / (kB * Last@TyNEVO);
  N[zNEVO_?NumericQ] := If[zNEVO < zmin || zNEVO ≥ 5, 0, NNEVO[zNEVO]];
  N[zNEVO_?NumericQ] := If[zNEVO ≥ 5, 0, NParthenope[zNEVO]];
];

```

Checking the agreement between N_{NEVO} and $N_{\text{ParthEnoPe}}$

(for $T > 10$ MeV Parthenope fit is not valid, hence the large difference)

We see some disagreement around 1.5 – 2 MeV ($\sim 1.5 - 2 \times 10^{10}$ K).

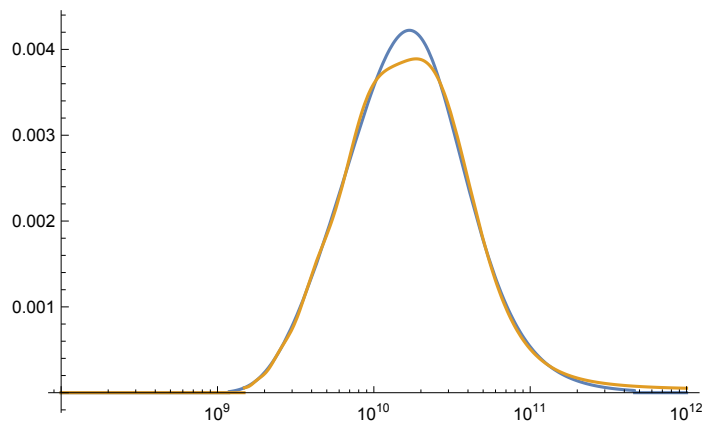
In[258]:=

```

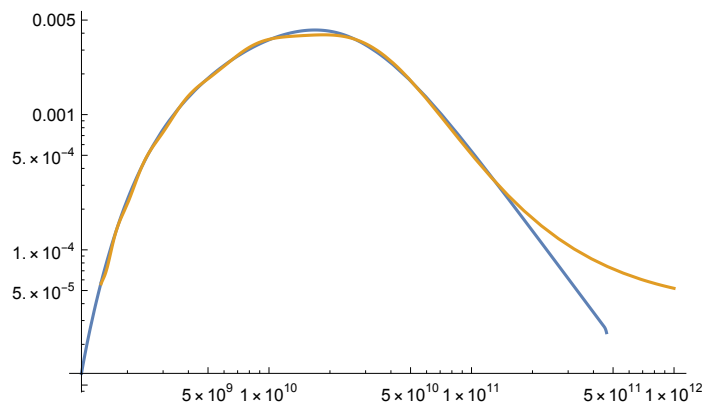
LogLinearPlot[{N[me / (kB * Tv)], NParthenope[me / (kB * Tv)]},
  [tracé log-linéaire
    {Tv, 10^8, 10^12}, PlotRange -> All]
    [zone de tracé [tout
LogLogPlot[{N[me / (kB * Tv)], NParthenope[me / (kB * Tv)]},
  [tracé log-log
    {Tv, 10^8, 10^12}, PlotRange -> All]
    [zone de tracé [tout

```

Out[258]=



Out[259]=



We can also perform a fit in Hermite polynomials of the NEVO heating function. Just for information.

In[260]:=

```

If[$NEVO,
  _si
  Nlz[lz_] :=  $\mathcal{N}$ NEVO[Exp@lz];
_exponentielle

(* Gram-Charlier expansion. Works better in that case *)
MyHermiteN[n_, x_] := Expand@Simplify[HermiteH[n, x / Sqrt[2]] / Sqrt[2]^n];
_développe _simplifie _H de Hermite _racine carrée _racine carrée

Fitfun = A / (Sqrt[2 Pi]  $\sigma$ ) Exp[-(lz -  $\mu$ )^2 / (2  $\sigma$ ^2)] (1
_racine _nombr _exponentielle
  + (S3 / 3! MyHermiteN[3, (lz -  $\mu$ ) /  $\sigma$ ])
  + (S4 / 4! MyHermiteN[4, (lz -  $\mu$ ) /  $\sigma$ ])
  + (S5 / 5! MyHermiteN[5, (lz -  $\mu$ ) /  $\sigma$ ])
  + (S6 / 6! MyHermiteN[6, (lz -  $\mu$ ) /  $\sigma$ ])
  + (S7 / 7! MyHermiteN[7, (lz -  $\mu$ ) /  $\sigma$ ])
  + (S8 / 8! MyHermiteN[8, (lz -  $\mu$ ) /  $\sigma$ ])
  + (S9 / 9! MyHermiteN[9, (lz -  $\mu$ ) /  $\sigma$ ])
  + (S10 / 10! MyHermiteN[10, (lz -  $\mu$ ) /  $\sigma$ ])));
fitlzmax = 1.7(*-1+2.7*);
fitlzmin = -4.3(*-1-3.*);
TableVals = Table[{lz, Nlz[lz]}, {lz, fitlzmin, fitlzmax, 0.01}];
_table
Fitvals = FindFit[TableVals, Fitfun, {{A, 0.01}, { $\mu$ , -1}, { $\sigma$ , 1}, {S3, 0},
_trouve ajustement
  {S4, 0}, {S5, 0}, {S6, 0}, {S7, 0}, {S8, 0}, {S9, 0}, {S10, 0}}, lz];
NlzFit[lz_] = Fitfun /. Fitvals;
NFit[z_] := NlzFit[Log[z]];
_logarithme
LogLogPlot[
_tracé log-log
  { $\mathcal{N}$ NEVO[me / (kB * Tv)],  $\mathcal{N}$ Parthenope[me / (kB * Tv)], NFit[me / (kB * Tv)]},
  {Tv, 1.2 * 10^9, 4 * 10^11}, PlotRange -> All, Frame -> True];
_zone de tracé _tout _cadre _vrai
]

```

By construction \mathcal{N} is the heat rate transfered in unit of Hubble rate. Or more precisely, the volumic heat rate (the source on the r.h.s of $d\rho/dt$ equation) is $dq/dt = H (kB T)^4 \mathcal{N}$ with plus sign for neutrinos and minus sign for electron/photons plasma.

See companion paper for more details in section II.F.

We transform it to a function of temperature or Log[T].

In[261]:=

```

 $\mathcal{N}$ T[Tv_] :=  $\mathcal{N}$ [me / (kB Tv)];
 $\mathcal{N}$ LT[lTv_] :=  $\mathcal{N}$ [me / (kB Exp@lTv)];
_exponentielle

```

Visualization of the heating period

In[263]:=

```

If[$PaperPlots, LogLogPlot[N[Tv], {Tv, Ti, 10^9}, Frame → True,
  si      tracé log-log      cadre      vrai
  FrameStyle → Thickness[0.004], FrameLabel → {"T (K)", "N(T)"},
  style de cadre      épaisseur      étiquette de cadre
  LabelStyle → {FontSize → 12}, PlotStyle → {Black, Thickness[0.003]}]]
  style d'étiquette      taille de police de ca...      style de tracé      noir      épaisseur
If[$PaperPlots,
  si
  Export["Plots/PlotCalN.pdf", Style[%, Magnification → 1], "PDF"];]
  exporte      style      agrandissement      fonction de dens

```

In[265]:=

```

DS2lTQED[lTv_] :=  $\frac{2 * 2 \pi^2}{45}$  DSTQED[Exp@lTv];
                                     exponentielle

DST2QED[Tv_] :=  $\frac{2 * 2 \pi^2}{45}$  DSTQED[Tv]

DS2lTNoQED[lTv_] :=  $\frac{2 * 2 \pi^2}{45}$  DSTNoQED[Exp@lTv];
                                     exponentielle

DST2NoQED[Tv_] :=  $\frac{2 * 2 \pi^2}{45}$  DSTNoQED[Tv]

```

We first solve $d \ln(aT) / d \ln(T)$ so as to get $a(T)$. This is computed using the fact that there is a cooling of electron/photon plasma due to interactions with neutrinos.

We distinguish the case with and without QED corrections.

The equation solved is Eq. 62 of companion paper. SolveaOFTwhenID calls the solver and can be recalled anytime we have varied parameters.

In[269]:=

```

SolveaOFTwhenID :=
  (
    laTCQED = NDSolveValue[
      {
        laTCN'[lTv] ==  $\frac{(\mathcal{N}lT[lTv] - DS2lTQED'[lTv])}{(\mathcal{N}lT[lTv] + 3 * DS2lTQED[lTv])}$ ,
        laTCN[Log@Tf] == Log[ $\frac{TCMB0}{DSTQED[Tf]^{(1/3)}}$ ]], {laTCN},
      {lTv, Log@Ti, Log@Tf}, PrecisionGoal → 40, AccuracyGoal → 9][[1]];

    laTCNoQED =
      NDSolveValue[
        {
          laTCNNoQED'[lTv] ==  $\frac{(\mathcal{N}lT[lTv] - DS2lTNoQED'[lTv])}{(\mathcal{N}lT[lTv] + 3 * DS2lTNoQED[lTv])}$ ,
          laTCNNoQED[Log@Tf] == Log[ $\frac{TCMB0}{DSTNoQED[Tf]^{(1/3)}}$ ]], {laTCNNoQED},
        {lTv, Log@Ti, Log@Tf}, PrecisionGoal → 40, AccuracyGoal → 9][[1]];
  );

```

We call SolveaOFTwhenID at first evaluation

In[270]:=

```
SolveaOFTwhenID
```

In[271]:=

```

aTCQED[Tv_] := Exp[laTCQED[Log@Tv]];
aCQED[Tv_] := aTCQED[Tv] / Tv;

```

In[273]:=

```

aCQED[Tf] Tf / TCMB0
aCQED[Ti] Ti / TCMB0

```

Out[273]=

1.

Out[274]=

0.71532294

In[275]:=

```

aTCNoQED[Tv_] := Exp[laTCNoQED[Log@Tv]];
aCNoQED[Tv_] := aTCNoQED[Tv] / Tv;

```

In[277]:=

```
aCNoQED[Tf] Tf / TCMB0
```

Out[277]=

1.

We invert numerically $a(T)$ to obtain $T(a)$. We also do it for $z=AT$ as a function of a . This operation is performed when we call the wrapping function `InvertaofTwhenID`.

In[278]:=

```

InvertaofTwhenID :=
  (TofaCQED = Interpolation@Table[{aCQED[T], T}, {T, ListT}];
    Interpolation Table
    TofaCNoQED = Interpolation@Table[{aCNoQED[T], T}, {T, ListT}];
    Interpolation Table

    aTofaCQED = Interpolation@Table[{aCQED[T], aTCQED[T]}, {T, ListT}];
    Interpolation Table
    aTofaCNoQED = Interpolation@Table[{aCNoQED[T], aTCNoQED[T]}, {T, ListT}];
    Interpolation Table
  );

```

We call it at first evaluation

In[279]:=

```
InvertaofTwhenID
```

In[280]:=

```

aC[T_] := If[$QEDPlasmaCorrections, aCQED[T], aCNoQED[T]]
si

```

We now solve $d(\rho_\nu)/d \ln(a)$

(we have to pay attention to units, hence the \hbar and c light placed in the system).

In fact we solve for the evolution of $a^4 \rho_\nu$ as a function of $\text{Log}[a]$. This is

Eq. 63 of companion paper. The function `Solve ρ_ν OFawhenID` calls the solver.

In[281]:=

```

SolvepvOFawhenID :=
(
Timing[a4pvLogaQED = NDSolveValue[{barpaNQED'[lav] == 1/(hbar^3 c light^5)}
    _chronométrage          _valeur donnée par solution numérique d'équation différentielle

    (kB aTofaCQED[Exp@lav]) ^ 4 NT[TofaCQED[Exp@lav]],
    _exponentielle          _exponentielle

    barpaNQED[Log@aCQED@Ti] == aBB (kB aTCQED[Ti])^4  $\frac{7}{8}$  Nneu},
    _logarithme

    {barpaNQED}, {lav, Log[aCQED[Ti]], Log[aCQED[Tf]]},
    _logarithme          _logarithme

    (*Method->"StiffnessSwitching",*)
    _méthode

    PrecisionGoal -> 13, AccuracyGoal -> 43][[1]];
    _objectif de précision    _objectif d'exactitude

Timing[a4pvLogaNoQED = NDSolveValue[{barpaNoQED'[lav] == 1/(hbar^3 c light^5)}
    _chronométrage          _valeur donnée par solution numérique d'équation différentielle

    (kB aTofaCNoQED[Exp@lav]) ^ 4 NT[TofaCNoQED[Exp@lav]],
    _exponentielle          _exponentielle

    barpaNoQED[Log@aCNoQED@Ti] == aBB (kB aTCNoQED[Ti])^4  $\frac{7}{8}$  Nneu},
    _logarithme

    {barpaNoQED}, {lav, Log[aCNoQED[Ti]], Log[aCNoQED[Tf]]},
    _logarithme          _logarithme

    (*Method->"StiffnessSwitching",*)
    _méthode

    PrecisionGoal -> 13, AccuracyGoal -> 43][[1]];
    _objectif de précision    _objectif d'exactitude

);

```

We call SolvepvOFawhenID at first evaluation

In[282]:=

```
SolvepvOFawhenID
```

In[283]:=

```

a4pvCQED[av_] := a4pvLogaQED[Log@av];
    _logarithme

pvCQED[av_] :=  $\frac{a4pvCQED[av]}{av^4}$ ;

```

In[285]:=

```

a4pvCNoQED[av_] := a4pvLogaNoQED[Log@av];
    _logarithme

pvCNoQED[av_] :=  $\frac{a4pvCNoQED[av]}{av^4}$ ;

```

```
In[287]:= aTofaCNoQED[a[10^12]] / TCMB0
```

```
Out[287]= 0.36690516
```

```
InterpolatingFunction[

+



Domain:  $\{\{1.95 \times 10^{-12}, 2.73 \times 10^{-7}\}\}$



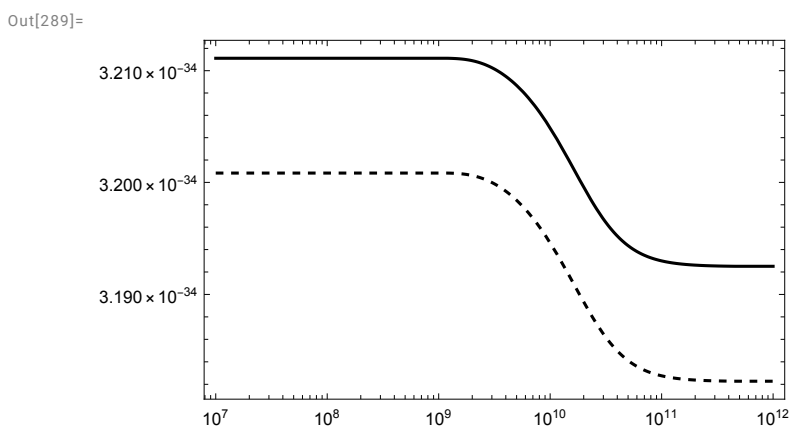
Output: scalar

][a[1 000 000 000 000]]
```

```
In[288]:= (*ρνC[av_] := If[$QEDPlasmaCorrections, ρνCQED[av], ρνCNoQED[av]];*)
_si
```

```
ρνIncompleteDecoupling[av_] :=
  If[$QEDPlasmaCorrections, ρνCQED[av], ρνCNoQED[av]];
_si
```

```
In[289]:= LogLogPlot[{a4ρνCQED[aC[Tv]], a4ρνCNoQED[aC[Tv]]},
_tracé log-log
  {Tv, Ti, Tf}, Frame → True, PlotStyle → {Black, {Dashed, Black}}]
_cadre _vrai _style de tracé _noir _en tirets _noir
```



We gather all operations needed to compute incomplete neutrino decoupling. Hence the function `RecomputeIncompleteNeutrinoDecoupling` can be called whenever we change some parameters so as to modify the incomplete decoupling of neutrinos.

```
In[290]:= RecomputeIncompleteNeutrinoDecoupling := (
  SolveaOFTwhenID;
  InvertaofTwhenID;
  SolveρνOFawhenID;
)
```

Neutrino Temperature

We extract the corresponding temperature of neutrinos

For full neutrino decoupling it is easy and deduced from entropy conservation.

We choose the correct temperature ratio depending on options chosen and we build the corre-

sponding energy density

In[291]:=

```
TvoverTDecoupling[T_] := (FourOverEleven DST[T])^(1/3);
ρvDecoupling[Tv_] := aBB (kB TvoverTDecoupling[Tv] Tv)^4  $\frac{7}{8}$  Nneu;
```

When considering incomplete decoupling of neutrinos, we find the temperature variation, which is defined as a brightness temperature. See Eq. 64 of companion paper.

In[293]:=

```
TvoverTIncompleteDecouplingQED[Tv_] :=  $\left( \frac{\rho_{\nu\text{QED}}[a_{\text{QED}}[Tv]]}{a_{\text{BB}} (kB Tv)^4 \frac{7}{8} N_{\text{neu}}} \right)^{(1/4)}$ ;
TvoverTIncompleteDecouplingNoQED[Tv_] :=  $\left( \frac{\rho_{\nu\text{NoQED}}[a_{\text{NoQED}}[Tv]]}{a_{\text{BB}} (kB Tv)^4 \frac{7}{8} N_{\text{neu}}} \right)^{(1/4)}$ ;
TvoverTIncompleteDecoupling[T_] := If[$QEDPlasmaCorrections,
    TvoverTIncompleteDecouplingQED[T], TvoverTIncompleteDecouplingNoQED[T]]
```

So that now we can decide to use either the full decoupling or the incomplete decoupling neutrino temperature, depending on the options chosen

In[296]:=

```
TvAverageoverT[T_] := If[$IncompleteNeutrinoDecoupling, If[$Taverage,
    TvoverTIncompleteDecoupling[T], If[$NEVO, TvAverageOverTγOFTγNEVOSafe[T],
    TvoverTIncompleteDecoupling[T]]], TvoverTDecoupling[T]];
If[$IncompleteNeutrinoDecoupling,
    If[$TrueTve && $NEVO, TvoverT[Tv_] := TveOverTγOFTγNEVOSafe[Tv],
    TvoverT[Tv_] := TvAverageoverT[Tv]]];
TvoverT[Tv_] := TvoverTDecoupling[Tv];;
```

Comparison of the two implementations of Taverage.

In[298]:=

```
(*If[$NEVO&&$IncompleteNeutrinoDecoupling,
    LogLinearPlot[{TvoverTIncompleteDecoupling[Tv],
    TvAverageOverTγOFTγNEVOSafe[Tv]}, {Tv, 10^8, 4*10^11}, Frame→True]])
(*If[$NEVO&&$IncompleteNeutrinoDecoupling, LogLinearPlot[
    {TvoverTIncompleteDecoupling[Tv]/TvAverageOverTγOFTγNEVOSafe[Tv]-1},
    {Tv, 10^8, 4*10^11}, Frame→True]])
```

Effective Description of Neutrinos

We check the translation into equivalent number of neutrinos. See section II.G of companion paper for the definition of N_{eff} .

This section is only informative.

In[299]:=

```

TvoverTDecouplingNoQED[T_] := (FourOverElevenNoQED * DSTNoQED[T])^(1/3);
TvoverTDecouplingQED[T_] := (FourOverElevenQED * DSTQED[T])^(1/3);

EffectiveNeutrinosQED[Tv_] := 3 (TvoverTIncompleteDecouplingQED[Tv])^4 /
  TvoverTDecouplingNoQED[Tv];

EffectiveNeutrinosNoQED[Tv_] := 3 (TvoverTIncompleteDecouplingNoQED[Tv])^4 /
  TvoverTDecouplingNoQED[Tv];

```

z (z is defined as $z=aT$ with the convention $z=1$ deep before BBN).

In[303]:=

```

zOFTDecouplingNoQED[T_] := (DSTNoQED[Ti])^(1/3) / DSTNoQED[T];
zOFTDecouplingQED[T_] := (DSTQED[Ti])^(1/3) / DSTQED[T];

zOFTIncompleteDecouplingNoQED[T_] := (aCNoQED[T] T) / (aCNoQED[Ti] Ti);
zOFTIncompleteDecouplingQED[T_] := (aCQED[T] T) / (aCQED[Ti] Ti);

```

The various z at the end depending on the physics. See table I in companion paper.

In[307]:=

```
zendQED = zOFTIncompleteDecouplingQED[Tf]
zendNoQED = zOFTIncompleteDecouplingNoQED[Tf]
```

```
zendDecouplingQED = zOFTDecouplingQED[Tf]
zendDecouplingNoQED = zOFTDecouplingNoQED[Tf]
```

$$\left(\frac{11.}{4}\right)^{(1/3)}$$

Out[307]=

1.39797

Out[308]=

1.3990925

Out[309]=

1.3998948

Out[310]=

1.4010185

Out[311]=

1.4010197

Effective number of neutrinos

In[312]:=

```
EffectiveNeutrinosQED[108]
EffectiveNeutrinosNoQED[108]
```

$$3 \left(\frac{\left(\frac{11.}{4}\right)^{(1/3)}}{\text{zendDecouplingQED}} \right)^4$$

Out[312]=

3.0439022

Out[313]=

3.0341556

Out[314]=

3.0096545

Neff from the neutrino energy density

In[315]:=

```
Neff := If[LSIIncompleteNeutrinoDecoupling, If[LSI$QEDPlasmaCorrections,
    EffectiveNeutrinosQED[108], EffectiveNeutrinosNoQED[108]],
    If[LSI$QEDPlasmaCorrections, 3  $\left(\frac{\left(\frac{11.}{4}\right)^{(1/3)}}{\text{zendDecouplingQED}}\right)^4$ , 3]]
```

In[316]:=

Neff

Out[316]=

3.0439022

The most accurate estimation of Neff, given the options chosen, is (because it uses directly the results of NEVO):

In[317]:=

```
Neff := 3 (TvAverageoverT[10^8] / TvovertDecouplingNoQED[10^8]) ^4
```

In[318]:=

Neff

Out[318]=

3.0439773

z_ν at the end of integration

(Only computed in the case of incomplete decoupling
because otherwise it remains unity)

In[319]:=

```
zvOFTQED[T_] :=
  TvovertIncompleteDecouplingQED[T] * zOFTIncompleteDecouplingQED[T];
zvOFTNoQED[T_] :=
  TvovertIncompleteDecouplingNoQED[T] * zOFTIncompleteDecouplingNoQED[T];
zvOFT[T_] := If[$QEDPlasmaCorrections, zvOFTQED[T], zvOFTNoQED[T]]
      |si
```

In[322]:=

```
zvendQED = zvOFTQED[10^8]
```

```
zvendNoQED = zvOFTNoQED[10^8]
```

Out[322]=

1.0014539

Out[323]=

1.0014548

The total energy density increase in neutrinos is

In[324]:=

```
zvendQED^4
```

```
zvendNoQED^4
```

Out[324]=

1.0058284

Out[325]=

1.0058317

N_{eff} is by definition $(z_\nu z_{\text{dec}} / z)^4$. See notation in companion paper.

So we recheck that we find the

N_{eff} given in the companion paper in Table I.

In[326]:=

$$3 * \left(z_{\text{vendQED}} * \frac{z_{\text{endDecouplingNoQED}}}{z_{\text{endQED}}} \right)^4$$

$$3 * \left(z_{\text{vendNoQED}} * \frac{z_{\text{endDecouplingNoQED}}}{z_{\text{endNoQED}}} \right)^4$$

$$3 * \left(\frac{z_{\text{endDecouplingNoQED}}}{z_{\text{endDecouplingQED}}} \right)^4$$

Out[326]=

3.0438923

Out[327]=

3.0341457

Out[328]=

3.0096447

However, in the case we use NEVO, the expression given directly above for Neff is exactly the one computed inside NEVO itself.

In[329]:=

```

If[$PaperPlots, ListLogLinearPlot[
  si      tracé log linéaire de liste
  Table[{Tv, TvoverTIncompleteDecouplingQED[Tv] / TvoverTDecouplingQED[Tv]},
    table
    {Tv, ListTRange[10^8, 10^12]}], Joined → True,
    joint   vrai
  PlotStyle → Black, FrameStyle → Thickness[0.004],
    style de tracé  noir    style de cadre  épaisseur
  Frame → True, FrameLabel → {"T(K)", "TvID / Tvdec"}]]
cadre     vrai    étiquette de cadre

If[$PaperPlots,
  si
  Export["Plots/PlotTnuvariation.pdf", Style[%, Magnification → 1], "PDF"];]
exporte    style    agrandissement    fonction de de

If[$PaperPlots,
  si
  ListLogLinearPlot[Table[{Tv, TvoverT[Tv]}, {Tv, ListTRange[10^8, 10^12]}],
    tracé log linéaire de liste  table
    Joined → True, PlotStyle → Black, FrameStyle → Thickness[0.004],
    joint   vrai    style de tracé  noir    style de cadre  épaisseur
    Frame → True, FrameLabel → {"T(K)", "TvID / Tvdec"}]]
    cadre     vrai    étiquette de cadre

If[$PaperPlots,
  si
  Export["Plots/PlotTnuOverT.pdf", Style[%, Magnification → 1], "PDF"];]
exporte    style    agrandissement    fonction de densité de

```


In[333]:=

```

If[$PaperPlots, ListLogLinearPlot[
  _si      _tracé log linéaire de liste
  {Table[{Tv, EffectiveNeutrinosQED[Tv]}, {Tv, ListTRange[10^8, 10^12]}],
    _table
    Table[{Tv, EffectiveNeutrinosNoQED[Tv]}, {Tv, ListTRange[10^8, 10^12]}],
    _table
    Table[{Tv, 3 (zOFTDecouplingNoQED[Tv] / zOFTDecouplingQED[Tv]) ^4},
    _table
      {Tv, ListTRange[10^8, 10^12]}] (*,
    Table[{Tv, 3 (zvOFT[Tv]) ^4}, {Tv, ListTRange[10^8, 10^12]}] *)},
  _table
  Joined → True, PlotStyle → {Black, {Red, Dashed}, {Blue, Dotted}},
  _joint    _vrai    _style de tracé    _noir    _rouge _en tirets    _bleu    _en pointillé
  Frame → True, FrameLabel → {"T (K)", "Neff"}, LabelStyle → {FontSize → 12},
  _cadre    _vrai    _étiquette de cadre    _style d'étiquette    _taille de police de caractères
  FrameStyle → Thickness[0.004], PlotRange → {2.99, 3.05}]]
  _style de cadre    _épaisseur    _zone de tracé
If[$PaperPlots,
  _si
  Export["Plots/PlotNeff.pdf", Style[%, Magnification → 1], "PDF"];]
  _exporte    _style    _agrandissement    _fonction de densité de prot

```

Scale factor determination

This is the function which gives the scale factor as a function of the temperature.

- 1) If neutrino decoupling is total, then the total entropy is conserved and so it is only a function of temperature (entropy density) and scale factor (volume), since $S = s a^3$. So this can be solved without a differential equation in time.
- 2) If Incomplete Neutrino decoupling is taken into account, we use the result aC previously obtained, that is we track the evolution of entropy, (see function SolveaOFTwhenID above).

In[335]:=

```

If[$IncompleteNeutrinoDecoupling,
  _si
  a[T_] := aC[T],
  a[T_] :=  $\frac{T_{CMB0}}{T_{DST}[T]^{(1/3)}}$ ];

```

In[336]:=

```

(*LogLinearPlot[{a[Tv], TCMB0/Tv}, {Tv, 10^9, 10^10},
  _tracé log-linéaire
  Frame → True, FrameLabel → {"T (K)", "a(T) / a0 TCMB0 / TCMB"}] *)
  _cadre    _vrai    _étiquette de cadre

```

Just for simplicity we define again the z and z_v variables which are $z = aT$ and $z = aT_v$ respectively

In[337]:=

$$zT[T_] := \frac{(a[T] T)}{(a[T_i] T_i)};$$

$$znuT[T_] := \frac{(a[T] \times T_{\text{over}}T[T] T)}{(a[T_i] \times T_{\text{over}}T[T_i] T_i)};$$

And again we check how they varied

In[339]:=

```
zT[Tf] // NP
znuT[Tf] // NP
```

Out[339]//NumberForm=

1.39797

Out[340]//NumberForm=

1.0017526

We build a table with (a, T) so as to obtain the inverse T(a) via an interpolation.

In the incomplete neutrino case we have already performed such inversion so there is a slight loss of time and computer energy.

In[341]:=

```
InvertaOFT := (Tofa = Interpolation@Table[{a[T], T}, {T, ListT}]);
(*aI=Interpolation@Table[{T,a[T]}, {T,ListT}];*)
```

In[342]:=

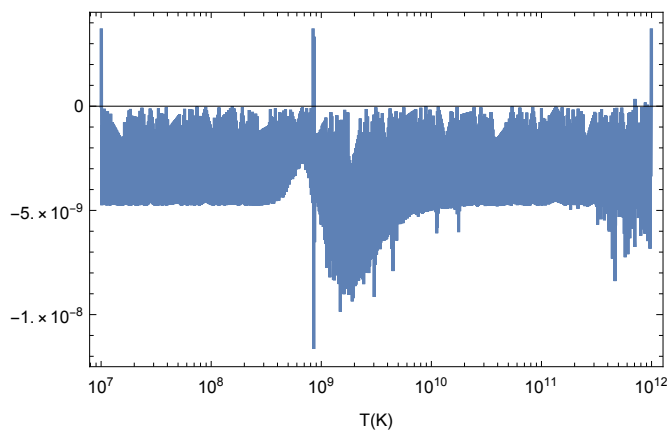
InvertaOFT

We can check how accurate is this inversion by checking how a(T(a)) is the identity. The error is below 10^{-8} so our numerics are precise enough.

In[343]:=

```
LogLinearPlot[{Tofa[a[T]] / T - 1},
  {T, Ti, Tf}, Frame → True, FrameLabel → {"T(K)", ""}]
```

Out[343]=



η factor with temperature dependence (because photons density has evolved due to electron positron recombination)

In[344]:=

$$\eta_{\text{factorT}}[\text{Tv_}] := \text{nB}[\text{a}[\text{Tv}]] * \frac{\pi^2}{2 \text{Zeta}[3]} \left(\frac{\hbar c \text{light}}{k_B \text{Tv}} \right)^3;$$

Another way to obtain this quantity

In[345]:=

$$\eta_{\text{factorTBis}}[\text{Tv_}] := \eta_{\text{factor}} * (\text{zT}[\text{Tf}] / \text{zT}[\text{Tv}])^3;$$

In[346]:=

```
 $\eta_{\text{factorTBis}}[\text{Ti}]$  // NP
```

```
 $\eta_{\text{factorT}}[\text{Ti}]$  // NP
```

Out[346]//NumberForm=

 1.6771735×10^{-9}

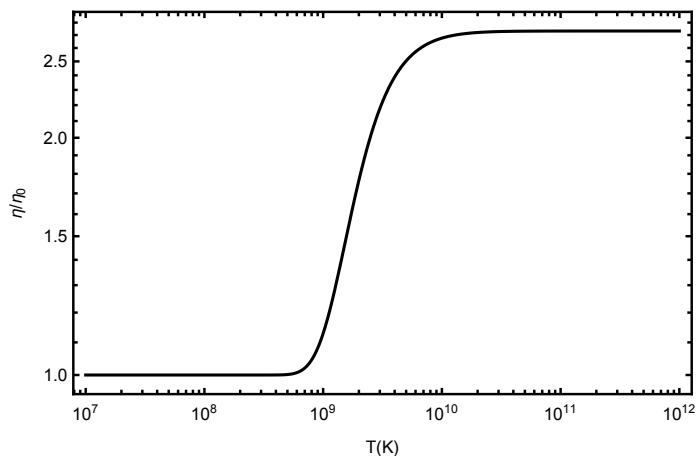
Out[347]//NumberForm=

 1.6771735×10^{-9}

In[348]:=

```
LogLogPlot[ $\eta_{\text{factorT}}[\text{Tv}] / \eta_{\text{factor}}$ , {Tv, Ti, Tf}, Frame → True,
  tracé log-log cadre vrai
  FrameStyle → Thickness[0.004], PlotStyle → Black, FrameLabel → {"T(K)", " $\eta/\eta_0$ "}]
  style de cadre épaisseur style de tracé noir étiquette de cadre
  If[$PaperPlots, Export["Plots/PlotEta.pdf", Style[%, Magnification → 1], "PDF"];]
  si exporte style agrandissement fonction de
```

Out[348]=



Check of normalization. We chose that today the scale factor is unity.

In[350]:=

```
 $\text{a}[\text{Tf}] \text{ Tf} / \text{TCMB0}$  // NP
```

Out[350]//NumberForm=

 $1.$

Weak reactions $n + \nu \leftrightarrow p + e$

Distribution functions derivatives

These Fermi-Dirac function and its derivatives are needed further. ϵ stands for energy. x for $1/(k_B T)$.

FDeipj means that it is the Fermi Dirac distribution multiplied by Energyⁱ and derived j times wrt to Energy. See Def B25 in companion paper.

In[351]:=

```

FDe2p0[en_, x_] = Simplify[FD[en, x] en^2];
                        [simplifie]

FDe3p0[en_, x_] = Simplify[FD[en, x] en^3];
                        [simplifie]

FDe2p2[en_, x_] = Simplify@D[D[FD[en, x] en^2, en], en];
                        [simplifie]      [··] [dérivée d]

FDe3p2[en_, x_] = Simplify@D[D[FD[en, x] en^3, en], en];
                        [simplifie]      [··] [dérivée d]

FDe4p2[en_, x_] = Simplify@D[D[FD[en, x] en^4, en], en];
                        [simplifie]      [··] [dérivée d]

FDe2p1[en_, x_] = Simplify@D[FD[en, x] en^2, en];
                        [simplifie]      [dérivée d]

FDe3p1[en_, x_] = Simplify@D[FD[en, x] en^3, en];
                        [simplifie]      [dérivée d]

FDe4p1[en_, x_] = Simplify@D[FD[en, x] en^4, en];
                        [simplifie]      [dérivée d]

```

In[359]:=

```

FDve2p0[en_, φ_, x_] = Simplify[FDv[en, φ, x] en^2];
                        [simplifie]

FDve3p0[en_, φ_, x_] = Simplify[FDv[en, φ, x] en^3];
                        [simplifie]

FDve2p2[en_, φ_, x_] = Simplify@D[D[FDv[en, φ, x] en^2, en], en];
                        [simplifie]      [··] [dérivée d]

FDve3p2[en_, φ_, x_] = Simplify@D[D[FDv[en, φ, x] en^3, en], en];
                        [simplifie]      [··] [dérivée d]

FDve4p2[en_, φ_, x_] = Simplify@D[D[FDv[en, φ, x] en^4, en], en];
                        [simplifie]      [··] [dérivée d]

FDve2p1[en_, φ_, x_] = Simplify@D[FDv[en, φ, x] en^2, en];
                        [simplifie]      [dérivée d]

FDve3p1[en_, φ_, x_] = Simplify@D[FDv[en, φ, x] en^3, en];
                        [simplifie]      [dérivée d]

FDve4p1[en_, φ_, x_] = Simplify@D[FDv[en, φ, x] en^4, en];
                        [simplifie]      [dérivée d]

```

λ_0

Born approximation λ_0

See e.g. Eq. 13 of [Lopez & Turner 1998]. Or Eq. 92 of companion paper.

The constant λ_0 is a proxy for $\tau_n^{-1} = \lambda_0 * [\cos^2(\theta_c) G_F^2 (C_V^2 + 3 C_A^2) / (2 \pi^3)]$

In[367]:=

```
λBORN = With[{q = Q / me}, NIntegrate[en (en - q)2 √{en2 - 1}, {en, 1, q}]]
```

avec intégrer numériquement

Out[367]=

1.636098

Corrections to λ_0

Radiative corrections to λ_0

1) If \$ResummedLogsRadiativeCorrections=False, we use Eq. 7 of [Czarnecki et al. 2004] which is B30 of companion paper.

Combined with Eq 20b of [Sirlin 1967] for Sirlin's universal function, that is Eq. B32 in companion paper.

In[368]:=

```
AgCzarnecki = -0.34;
CCzarnecki = 0.891;
mA = 1.2 GeV;
ConstantSirlin = 4 Log[mz / mp] + Log[mp / mA] + 2 CCzarnecki + AgCzarnecki;
```

logarithme logarithme

For information, this quantity is evaluated to 40 in [Dicus et al.]

In[372]:=

```
ConstantSirlin + 3 Log[mp / (me)] - 3 / 4
```

logarithme

Out[372]=

41.298783

In[373]:=

```
Rd[x_] := ArcTanh[x] / x;
```

NB : ArcTanh[x] = 1/2 Log[(1+x)/(1-x)]

In[374]:=

```
Lfun[x_] = Integrate[Log[1 - t] / t, {t, 0, x}, Assumptions -> x < 1 && x > 0]
```

intégrer hypothèses

(* Lfun is called the Spence function *)

Out[374]=

-PolyLog[2, x]

It can be computed from a Taylor expansion (this used to be done in the past), but a direct evaluation by Mathematica is much better

In[375]:=

```
LfunSeries[b_] = Normal@Series[-1/4*(1+b)^6* $\frac{4}{b}$ Lfun[ $\frac{2b}{1+b}$ ], {b, 0, 12(*22*)}]
```

[forme n... [développement en série entière]

Out[375]=

$$2 + 11b + \frac{224b^2}{9} + \frac{89b^3}{3} + \frac{1496b^4}{75} + \frac{596b^5}{75} + \frac{128b^6}{49} + \frac{68b^7}{49} + \frac{3704b^8}{3969} + \frac{21988b^9}{33075} + \frac{2016208b^{10}}{4002075} + \frac{946628b^{11}}{2401245} + \frac{43024472b^{12}}{135270135}$$

Sirlin universal function (Eq 20b of [Sirlin 1967]) on which we add the constants of Eq. 7 of [Czarnecki et al. 2004]) is obtained as (this is B32 of companion paper)

In[376]:=

```
$SeriesSpenceFunction = False;
```

[faux]

```
SirlinGFunction[b_, y_, en_] := (3 Log[mp / (me)] - 3 / 4 +
```

[logarithme]

$$4 (Rd[b] - 1) \left(\frac{y}{3 en} - 3 / 2 + \text{Log}[2 y] \right) + Rd[b] \left(2 (1 + b^2) + \frac{y^2}{6 en^2} - 4 b Rd[b] \right) +$$

[logarithme]

```
If[$SeriesSpenceFunction, -4 / (1 + b)^6 * LfunSeries[b],  $\frac{4}{b}$  Lfun[ $\frac{2b}{1+b}$ ]]];
```

[si]

```
Cd[b_, y_, en_] := (ConstantSirlin + SirlinGFunction[b, y, en]);
```

NB : In [Dicus et al. 1982] Eq 2.14 (or in [Lopez&Turner 1998] Eq. 17) they use the approximation

$$\frac{4}{b} \text{Lfun}\left[\frac{2b}{1+b}\right] = -4 \left(2 + 11b + \frac{224}{9}b^2 + \frac{89}{3}b^3 + \frac{1496}{75}b^4 + \frac{596}{75}b^5 + \frac{128}{49}b^6 \right) / (1+b)^6$$

However there is a typo and some incorrect coefficients. In [Kernan] there are approximate coefficients which are nearly correct.

2) if \$ResummedLogsRadiativeCorrections = True, we use a resummed version of all the $\text{Log}[m_Z/m_\rho]$. It consists in using Eq. 15 of [Czarnecki 2004] or B35 in companion paper. See also [Esposito et al. 1998])

We build the multiplicative factor for inner radiative corrections (Eq. 15 of [Czarnecki et al. 2004]), that is B35 of companion paper.

In[379]:=

```
LFactor = 1.02094;
SFactor = 1.02248;
deltafactor = -0.00043 * 2 Pi / alphaS;
```

[nombre pi]

```
NLL = -0.0001;
```

In[383]:=

```

RadiativeCorrectionsResummed[b_, y_, en_] :=
  (1 +  $\frac{\alpha_{FS}}{2\pi}$  (SirlinGFunction[b, y, en] - 3 Log[ $\frac{m_p}{2Q}$ ])) *
  (LFactor +  $\frac{\alpha_{FS}}{\pi}$  CCzarnecki +  $\frac{\alpha_{FS}}{2\pi}$   $\delta$ factor) *
  (SFactor +  $\frac{1}{134 * 2 * \text{Pi}}$  * (Log[ $\frac{m_p}{m_A}$ ] + AgCzarnecki) + NLL);

```

Finally we define a function which selects either choice depending on option

In[384]:=

```

RadiativeCorrections[b_, y_, en_] := If[$ResummedLogsRadiativeCorrections,
  RadiativeCorrectionsResummed[b, y, en], (1 +  $\frac{\alpha_{FS}}{2\pi}$  Cd[b, y, en])]

```

- Fermi function for Coulomb interactions of electron and proton (if on the same side of the interaction).

Either the relativistic or the non-relativistic depending on option \$RelativisticFermiFunction. That is either Eq. 99 or Eq. 100 of companion paper depending on option chosen.

In[385]:=

```

FermiRelat[b_] := With[{ $\gamma = \text{Sqrt}[1 - \alpha_{FS}^2]$  - 1,  $\lambda_{\text{Compton}} = 1 / (m_e / (\hbar c \text{light}))$ },
  (1 +  $\gamma / 2$ ) * 4 (  $\frac{2 \text{radiusproton } b}{\lambda_{\text{Compton}}}$  )2 $\gamma$  *
   $\frac{1}{\text{Gamma}[3 + 2\gamma]^2}$  Exp[ $\frac{\pi \alpha_{FS}}{b}$ ] *  $\frac{1}{(1 - b^2)^\gamma}$  Abs[Gamma[1 +  $\gamma + I \frac{\alpha_{FS}}{b}$ ]]2];

FermiNonRelat[b_] :=  $\frac{2\pi \alpha_{FS} / b}{1 - \text{Exp}[-2\pi \alpha_{FS} / b]}$ ;

```

In[387]:=

```

If[$RelativisticFermiFunction,
  Fermi[b_] := FermiRelat[b];
  bFermi[b_] := b Fermi[b];,

  Fermi[b_] := FermiNonRelat[b];
  bFermi[b_] :=  $\frac{2\pi \alpha_{FS}}{1 - \text{Exp}[-2\pi \alpha_{FS} / b]}$ ;]

```

In[388]:=

```

If[$PaperPlots,
  si
  DFermi = Plot[{100 * (FermiRelat[b] / FermiNonRelat[b] - 1)}, {b, 0, 1},
    tracé
    Frame → True, FrameStyle → Thickness[0.004], PlotStyle → {Black}, FrameLabel →
    cadre vrai style de cadre épaisseur style de tracé noir étiquette de cadre
    {"x", "100 x (Frel(x)/Fnon rel(x) - 1)"}, LabelStyle → {FontSize → 12}]
    style d'étiquette taille de police de caractères

  If[$PaperPlots,
    si
    Export["Plots/PlotDeltaFermi.pdf", Style[DFermi, Magnification → 1], "PDF"];
    exporte style agrandissement fonction de de

```

λ_0 when taking only Fermi function and not radiative corrections

In[390]:=

```

λFermiOnly = With[{q = Q / me, b =  $\sqrt{en^2 - 1}$  / en, y = Q / me - en},
  avec
  NIntegrate[en (en - q)2 en * bFermi[b], {en, 1.0000001, q}]
  intègre numériquement

```

Out[390]=

1.6923404

This already a 3.44 % correction

In[391]:=

```
λFermiOnly / (λBORN)
```

Out[391]=

1.034376

Adding the inner radiative corrections, the constant involved in the decay of the neutron is

In[392]:=

```

λRad = With[{q = Q / me, b =  $\sqrt{en^2 - 1}$  / en, y = Q / me - en},
  avec
  NIntegrate[en (en - q)2 en (RadiativeCorrections[b, y, en]) * bFermi[b],
    intègre numériquement
    {en, 1.0000001, q}]

```

Out[392]=

1.7583844

Note that in companion paper, Eq. 106, the value 1.75767 is quoted when it should be the value found here 1.758384.

This adds another 3.90 % correction as found in Eq 16 of [Czarnecki et al. 2004].

In[393]:=

```
λRad / λFermiOnly
```

Out[393]=

1.0390252

Finite mass corrections to λ_0

See companion papers for expression of finite nucleon mass corrections.

We take the general form of finite mass correction and remove all terms which involve temperature. So we consider Eq. 118 of companion paper.

In[394]:=

```
IntegrateCorrectionNeutronDecay[fun_] :=
  NIntegrate[fun[pe], {pe, 0.0000001, Sqrt[(Q/me)^2 - 1]}, WorkingPrecision -> 15];
  intègre numériquement précision de travail
```

In[395]:=

```
 $\chi$ FMNeutronDecay[en_, pe_] :=
  With[{M = mn/me, enu = en - Q/me,
    avec
    f1 =  $\frac{(1 + gA)^2 + 4 \text{fWM } gA}{(1 + 3 gA^2)}$ , f2 =  $\frac{(1 - gA)^2 - 4 \text{fWM } gA}{(1 + 3 gA^2)}$ , f3 =  $\frac{(gA^2 - 1)}{(1 + 3 gA^2)}$ },
    f1 * enu^2  $\left( \frac{pe^2}{M * en} \right)$ 
    + f2 * enu^3  $\left( -\frac{1}{M} \right)$ 
    + (f1 + f2 + f3)  $\frac{1}{2 M} * (4 \text{enu}^3 + 2 \text{enu } pe^2)$ 
    + f3 *  $\frac{1}{3 M} 3 \text{enu}^2 \frac{pe^2}{(en)}$ 
  ];
```

Without coupling to radiative corrections we get

In[396]:=

```
I $\lambda$ FMBasic[pe_] := With[{en = Sqrt[pe^2 + 1]}, With[{b = pe/en}, pe^2 *
  avec racine carrée avec
  ( $\chi$ FMNeutronDecay[en, pe])
  ]];
```

In[397]:=

```
IntegrateCorrectionNeutronDecay[I $\lambda$ FMBasic] /  $\lambda$ BORN
```

Out[397]=

```
-0.0020647828
```

However we couple to radiative corrections if \$CoupledFMandRC = True

In[398]:=

```

IλFM[pe_] := With[ $\left\{en = \sqrt{pe^2 + 1}\right\}$ , With[{b = pe / en}, pe2 *
  (χFMNeutronDecay[en, pe] * If[$RadiativeCorrections && $CoupledFMandRC,
    (RadiativeCorrections[b, Abs[en - Q / me], en]) Fermi[b], 1])
  ]];

```

In[399]:=

```

λFM = If[$FiniteNucleonMass, IntegrateCorrectionNeutronDecay[IλFM], 0]

```

Out[399]=

```
-0.00362833914973646
```

When taking only into account Fermi function and finite mass effect on should get 1.6887. See Eq. 6 of [Czarnecki et al. 2004]

In[400]:=

```

λFermiOnly + λFM
1.6887 / % // NP

```

Out[400]=

```
1.6887121
```

Out[401]//NumberForm=

```
0.99999283
```

We compare the correction to the Born result

In[402]:=

```
CorrectionRate = λFM / λBORN
```

Out[402]=

```
-0.0022176784
```

If Coulomb and Radiative corrections are set to False (0) this is exactly (-0.00206) what is found by [Lopez & Turner 1997] (see three lines after Eq. 23, in the text).

See also the value -0.00201 found in Eq. 20 of [Seckel 1993].

We also reproduce for comparison the exact finite mass correction to the neutron decay rate as computed by Eq. 19 of [Lopez & Turner 1998], which uses two-dimensional integrals. Again we find the -0.00206 correction so our finite mass method based one-dimensional integrals is reliable.

```

In[403]:=
λexact = With[{pe = Sqrt[en^2 - 1]},
  Avec[
    racine carrée
    With[{enu = ((mn^2 - mp^2) / me^2 + 1 - 2 mn / me en) / (2 (mn / me - en + pe Cnu))},
      Avec[
        With[{Ep = mn / me - enu - en,
          Avec[
            f1 = ((1 + gA)^2 + 4 fWM gA) / (1 + 3 gA^2), f2 = ((1 - gA)^2 - 4 fWM gA) / (1 + 3 gA^2),
            f3 = (gA^2 - 1) / (1 + 3 gA^2)}, With[{J = 1 + (enu + pe Cnu) / (Ep)},
              Avec[
                M2 = f1 mn / me enu (Ep en - (-pe^2 - pe enu Cnu)) + f2 mn / me en
                  (Ep enu - (-pe enu Cnu - enu^2)) + f3 mn / me Ep (enu en - Cnu enu pe)},
                NIntegrate[1 / 2 * M2 pe enu / (mn / me Ep J), {en, 1,
                  intègre numériquement
                  (Q - (Q^2 - me^2) / (2 mn)) / me}, {Cnu, -1, 1}, PrecisionGoal -> 10
                  objectif de précision
                  (*, Method -> {Automatic, "SymbolicProcessing" -> 0} *)]
                méthode automatique
              ]
            ]
          ]
        ]
      ]
    ]
  ]
];

```

In[404]:=

(λexact - λBORN) / λBORN // NP

Out[404]//NumberForm=

-0.0020636787

Total correction

The total correction for the neutron decay constant λ_0 is the sum of the Radiative corrected one plus the finite mass effects

We recall the result from radiative corrections (again we stress that Eq. 106 in companion paper should report that value).

In[405]:=

λRad

Out[405]=

1.7583844

The result from mass corrections (see text before Eq. 120 in companion paper.)

In[406]:=

λFM

Out[406]=

-0.00362833914973646

And we sum them to get Eq. 120 of companion paper.

In[407]:=

λRadandFM = λRad + λFM

Out[407]=

1.754756

This gives a total correction which is

In[408]:=

 $\lambda_{\text{RadandFM}} / \lambda_{\text{BORN}}$

Out[408]=

1.072525

If we compare with [Cooper et al 2010] (Phys. Rev. C 81, 035503 (2010)). They find that this parameter should be

In[409]:=

```

λCooper = 1.03887 * 1.6887
λCzarnecki = 1.0390 * 1.6887
(* = (1+RC)*f with f=1.6887 and RC = 0.0390(8) [Czarnecki et al. 2004]] *)

```

Out[409]=

1.7543398

Out[410]=

1.7545593

We compute the ratio to see how far we are

In[411]:=

$$\lambda_{\text{RadandFM}} / \lambda_{\text{Cooper}}$$

$$\lambda_{\text{RadandFM}} / \lambda_{\text{Czarnecki}}$$

Out[411]=

1.0002373

Out[412]=

1.0001121

We check that it is in agreement with the theoretical formula. This expression below should reproduce the neutron decay time. It is amazingly close !

In[413]:=

```

MixingCosAngle = 0.97420;
(* 0.97420(+/-16) from PDG2017 (The review onf Vud Vus of the PDG 2017).*)
(*The value from PDG2020 review on Vud Vus is 0.97370
(+/-20) but this brings tension on the unitarity of the CKM
matrix. Hence we stick to the previous value. This is only used
to check if one recovers from theory the neutrino lifetime.*)
MyK = MixingCosAngle2 (GF)2 (1 + 3 (gA)2) me5 / (2 π3 hbar)
1 / MyK / λRadandFM
1 / MyK / λCzarnecki
1 / MyK / λCooper

```

Out[414]=

0.00064820991

Out[415]=

879.15934

Out[416]=

879.25791

Out[417]=

879.36794

Indeed with the PDG2017 value of Vud, we recover $\tau_{\text{neutron}} \sim 879.4$ or so hence we feel that the

neutron lifetime value is not a much more solid ground than what it was in the pas.

Let us also compare with Eq. 17 of [Czarnecki et al. 2004] to check how close we are in including all corrections. See text after Eq. 120 in companion paper as well.

In[418]:=

```
ConstantVud = hbar * (2 π³) / (me)⁵ / (GF)² / λRadandFM
ConstantVud2 = hbar * (2 π³) / (me)⁵ / (GF)² / λCzarnecki
ConstantVud3 = hbar * (2 π³) / (me)⁵ / (GF)² / λCooper
```

Out[418]=

4907.3772

Out[419]=

4907.9274

Out[420]=

4908.5416

Born rates

We compute in this section the Born rates for $n \rightarrow p$ and $p \rightarrow n$.

Tools to perform integration on electron momentum

In[421]:=

```
pemin = 0.00001;
pemiddle[x_] :=
  Sqrt[Max[pemin², (Q / me)² - 1 - If[$QEDMassShift, dme2x[x], 0]]];
  racin· maximum | si
pemaxC[x_] := Max[12, 30 / x];
  maximum
pemax[x_] := Max[12, 30 / x];
  maximum
```

\$TnuEqualT is some cheat to check detailed balance. Should be False. When it is True neutrinos have always the plasma temperature.

In[425]:=

```
$TnuEqualT = False;
  faux
```

In[426]:=

```

IntegratedpNpoints[fun_, sgnq_, Tv_, Npoints_] :=
  With[ $\left\{x = \frac{me}{(kB\ Tv)}, znu = \frac{me}{(kB\ Tv\ T_{overT}[Tv])}\right\}$ ,
     $\int$ 
    If[$FastPENRatesIntegrals,
      IntegrateFunction[
        fun[#, x, If[$TnuEqualT, x, znu], sgnq] &, pemin, pemaxC[x], Npoints],
      NIntegrate[fun[pe, x, If[$TnuEqualT, x, znu], sgnq],
        {pe, pemin, pemiddle[x], pemax[x]}]
    ]

IntegrateRatedp[fun_, sgnq_, Tv_] :=
  IntegratedpNpoints[fun, sgnq, Tv, $PENRatesIntegralsPoints];

```

Energy as a function of momentum, taking into account QED mass shifts.

This is only used if the option is set to True but it is useless because it is taken into account in finite temperature corrections

In[428]:=

```

enOFpe[pe_, x_] := Sqrt[ $pe^2 + 1 + \text{If}[\$QEDMassShift, dme2x[x], 0]$ ];

```

Functions to build integrands in electron momentum. Without and with CCR corrections. This is then passed to the integrating routine IntegrateRatedp.

In[429]:=

```

IPENDpFromxNoCCR[en_, pe_, x_, znu_, sgnq_, xfunction_] :=
  With[ $\{q = Q / me\}$ , With[ $\{b = pe / en\}$ ,
     $pe^2 * (\chi\text{function}[en, pe, x, znu, sgnq] + \chi\text{function}[-en, pe, x, znu, sgnq])$ 
  ]];

```

In[430]:=

```

Fermi[sgnq_, signE_, b_?NumericQ] := If[sgnq signE > 0, Fermi[b], 1];
SetAttributes[Fermi, Listable];

```

In[432]:=

```

IPENDpFromχCCR[en_, pe_, x_, znu_, sgnq_, χfunction_] :=
  With[{q = Q / me, b = pe / en},
    avec
    pe2 * (χfunction[en, pe, x, znu, sgnq]
      (RadiativeCorrections[b, Abs[sgnq Q / me - en], en])
      valeur absolue
      Fermi[sgnq, 1, b] + χfunction[-en, pe, x, znu, sgnq]
      (RadiativeCorrections[b, Abs[sgnq Q / me + en], en]) Fermi[sgnq, -1, b])
    valeur absolue
  ];

```

Eq 2.29 in [Brown & Sawyer] for the Born approximation. It is Eq. 79 of companion paper

In[433]:=

```

χ[en_, pe_, x_, znu_, sgnq_] :=
  With[{q = Q / me}, FDv[en - sgnq q, sgnq ξv, znu] × FD[-en, x] (en - sgnq q)2];
avec

```

Eq 2.30 in [Brown & Sawyer], that is the integration of Eq. 78 in companion paper

In[434]:=

```

IPENDp[pe_, x_, znu_, sgnq_] :=
  IPENDpFromχNoCCR[enOFpe[pe, x], pe, x, If[$TnuEqualT, x, znu], sgnq, χ]
si

```

We also define a function which is incorrect in which we force the neutrinos to have the temperature of photons. This is to check detailed balance. Indeed, detailed balance is satisfied only if all species have the same temperature.

In[435]:=

```

IPENDpCheatNeutrinoTemperature[pe_, x_, znu_, sgnq_] :=
  IPENDpFromχNoCCR[Sqrt[pe2 + 1], pe, x, x, sgnq, χ]
racine carrée

```

Born rates are given by Eq 2.30 in [Brown & Sawyer].

In[436]:=

```

λnT0pBORN[Tv_] := IntegrateRatedp[IPENDp, 1, Tv];
λpT0nBORN[Tv_] := IntegrateRatedp[IPENDp, -1, Tv];

```

Born rates where the neutrino temperature is cheated to be equal to photons temperature are then given by

In[438]:=

```

λnT0pBORNcheatNeutrino[Tv_] :=
  IntegrateRatedp[IPENDpCheatNeutrinoTemperature, 1, Tv];
λpT0nBORNcheatNeutrino[Tv_] :=
  IntegrateRatedp[IPENDpCheatNeutrinoTemperature, -1, Tv];

```

Spectral distortions of neutrino spectrum

Shift in the weak rates due to spectral distortions of neutrino distribution functions.

In[440]:=

```
 $\delta\chi[\text{en\_}, \text{pe\_}, \text{x\_}, \text{znu\_}, \text{sgnq\_}] :=$ 
  With[{q = Q / me },  $\delta\text{FDv}[\text{en} - \text{sgnq } q, \text{sgnq } \xi v, \text{znu}] \times \text{FD}[-\text{en}, \text{x}] (\text{en} - \text{sgnq } q)^2$ ];
  avec
```

We add this spectral distortions effect at the Born approximation level.

In[441]:=

```
(* Spectral distortions with Born approximation*)
IPENDpSD[pe_, x_, znu_, sgnq_] :=
  IPENDpFromχNoCCR[enOFpe[pe, x], pe, x, If[$TnuEqualT, x, znu], sgnq,  $\delta\chi$ ]
  si

IPENDpSDCCR[pe_, x_, znu_, sgnq_] :=
  IPENDpFromχCCR[enOFpe[pe, x], pe, x, If[$TnuEqualT, x, znu], sgnq,  $\delta\chi$ ]
  si

λnT0pSD[Tv_] := IntegrateRatedp[IPENDpSD, 1, Tv];
λpT0nSD[Tv_] := IntegrateRatedp[IPENDpSD, -1, Tv];
λnT0pSDCCR[Tv_] := IntegrateRatedp[IPENDpSDCCR, 1, Tv];
λpT0nSDCCR[Tv_] := IntegrateRatedp[IPENDpSDCCR, -1, Tv];
```

Finite mass effects

For finite mass effects, we use a (Fokker-Planck) expansion which leads to one-dimensional integrals.

We include also the weak-magnetism which is important indeed. This is B23 of companion paper.

In[447]:=

```

χFM[en_, pe_, x_, znu_, sgnq_] :=
  With[ $\left\{ \phi = \text{sgnq } \xi v, q = Q / m_e, M = \frac{(m_p + m_n - \text{sgnq } Q)}{(2 m_e)}, \right.$ 
     $\left. \text{avec} \right.$ 
    Mp = mp / me , Mn = mn / me , enu = en - sgnq Q / me ,
    f1 =  $\frac{((1 + \text{sgnq } gA)^2 + 4 f_{WM} \text{sgnq } gA)}{(1 + 3 gA^2)},$ 
    f2 =  $\frac{((1 - \text{sgnq } gA)^2 - 4 f_{WM} \text{sgnq } gA)}{(1 + 3 gA^2)},$  f3 =  $\frac{(gA^2 - 1)}{(1 + 3 gA^2)} \},$ 
    f1 * FDve2p0[enu, ϕ, znu] × FD[-en, x]  $\left( \frac{pe^2}{M * en} \right)$ 
    + f2 * FDve3p0[enu, ϕ, znu] × FD[-en, x]  $\left( -\frac{1}{M} \right)$ 
    + (f1 + f2 + f3)  $\frac{1}{2 * M} *$ 
      (FDve4p2[enu, ϕ, znu] × FD[-en, x] + FDve2p2[enu, ϕ, znu] × FD[-en, x] pe2)
    + (f1 + f2 + f3)  $\frac{1}{2 M} *$ 
      (FDve4p1[enu, ϕ, znu] × FD[-en, x] + FDve2p1[enu, ϕ, znu] × FD[-en, x] pe2)
    - (f1 + f2)  $\frac{1}{x M} *$  (FDve3p1[enu, ϕ, znu] × FD[-en, x] +
      FDve2p1[enu, ϕ, znu] × FD[-en, x] pe2 / (-en))
    - f3 *  $\frac{3}{x M}$  FDve2p0[enu, ϕ, znu] × FD[-en, x]
    (* This term seems to give very small corrections *)
    + f3 *  $\frac{1}{3 M}$  FDve3p1[enu, ϕ, znu] × FD[-en, x]  $\frac{pe^2}{(en)}$ 
    + f3 *  $\frac{2}{2 * 3 M}$  FDve3p2[enu, ϕ, znu] × FD[-en, x]  $\frac{pe^2}{(en)}$ 
    - (f1 + f2 + f3) *  $\frac{3}{2 x} * \left( 1 - \left( \frac{Mn}{Mp} \right)^{\text{sgnq}} \right) * (FDve2p1[enu, \phi, znu] \times FD[-en, x])$ 
    (* Note that this last line is different from the published
      version of the Physics Report where there is a typo.*)
  ];

```

Integrand for finite mass corrections. We couple it to radiative corrections.

In[448]:=

```

IPENDpFMNoCCR[pe_, x_, znu_, sgnq_] :=
  IPENDpFromχNoCCR[enOFpe[pe, x], pe, x, If[$TnuEqualT, x, znu], sgnq, χFM]
   $\left[ \text{si} \right]$ 
IPENDpFMCCR[pe_, x_, znu_, sgnq_] :=
  IPENDpFromχCCR[enOFpe[pe, x], pe, x, If[$TnuEqualT, x, znu], sgnq, χFM]
   $\left[ \text{si} \right]$ 

```

Integrand for finite mass corrections when neutrinos are forced to have the same temperature as photons.

In[450]:=

```
IPENDpFMHeatNeutrinoTemperature[pe_, x_, znu_, sgnq_] :=
  IPENDpFromχNoCCR[enOFpe[pe, x], pe, x, x, sgnq, χFM]
```

In[451]:=

```
Clear[λnT0pFMCCR, λpT0nFMCCR, λnT0pFMNoCCR,
  efface
  λpT0nFMNoCCR, λnT0pHeatNeutrinoFM, λpT0nHeatNeutrinoFM]
```

Finite mass corrections using the λ_0 which is corrected with radiative corrections. The computation below implements Eqs. 114 of companion paper.

In[452]:=

```
λnT0pFMCCR[Tv_] := IntegrateRatedp[IPENDpFMCCR, 1, Tv];
λpT0nFMCCR[Tv_] := IntegrateRatedp[IPENDpFMCCR, -1, Tv];
```

Finite mass corrections using the λ_0 which is computed with the Born infinite mass expression (we do not advise to use it).

In[454]:=

```
λnT0pFMNoCCR[Tv_] := IntegrateRatedp[IPENDpFMNoCCR, 1, Tv];
λpT0nFMNoCCR[Tv_] := IntegrateRatedp[IPENDpFMNoCCR, -1, Tv];
```

Finite mass corrections cheating with the neutrino temperature and setting it equal to photons temperature

In[456]:=

```
λnT0pHeatNeutrinoFM[Tv_] :=
  IntegrateRatedp[IPENDpFMHeatNeutrinoTemperature, 1, Tv];
λpT0nHeatNeutrinoFM[Tv_] :=
  IntegrateRatedp[IPENDpFMHeatNeutrinoTemperature, -1, Tv];
```

Examples of neutron decay corrections for some low temperatures:

In[458]:=

 $\lambda nT0pFMCCR[10^8] / \lambda nT0pBORN[10^8] // \text{Timing}$ [\[chronométrage\]](#) $\lambda nT0pFMCCR[10^{7.5}] / \lambda nT0pBORN[10^{7.5}] // \text{Timing}$ [\[chronométrage\]](#)

General : 74.543986 5.066670300353 $\times 10^{-311}$ is too small to represent as a normalized machine number;

precision may be lost. [i](#)

General : 75.926243 6.935210373530 $\times 10^{-314}$ is too small to represent as a normalized machine number;

precision may be lost. [i](#)

General : 77.321277 9.489314655635 $\times 10^{-317}$ is too small to represent as a normalized machine number;

precision may be lost. [i](#)

General : Further output of General::munfl will be suppressed during this calculation. [i](#)

Out[458]=

{0.024698, -0.0022372813}

General : 7.6413445 6.885424250140 $\times 10^{-315}$ is too small to represent as a normalized machine number;

precision may be lost. [i](#)

General : 8.081966 8.074833693947 $\times 10^{-324}$ is too small to represent as a normalized machine number;

precision may be lost. [i](#)

General : 8.5351787 9.368036178799 $\times 10^{-333}$ is too small to represent as a normalized machine number;

precision may be lost. [i](#)

General : Further output of General::munfl will be suppressed during this calculation. [i](#)

Out[459]=

{0.036631, -0.0022253145}

In[460]:=

 $\lambda nT0pFMCCR[.8 \times 10^{10}] / \lambda nT0pBORN[.8 \times 10^{10}] // \text{Timing}$ [\[chronométrage\]](#) $\lambda nT0pFMCCR[10^{10}] / \lambda nT0pBORN[10^{10}] // \text{Timing}$ [\[chronométrage\]](#) $\lambda nT0pFMCCR[10^{10.5}] / \lambda nT0pBORN[10^{10.5}] // \text{Timing}$ [\[chronométrage\]](#)

Out[460]=

{0.005394, -0.0091468504}

Out[461]=

{0.005251, -0.01090925}

Out[462]=

{0.005077, -0.030541473}

In[463]:=

$$\lambda_{nT0pFMCCR}[10^{11}]$$

$$\lambda_{pT0nFMCCR}[10^{11}]$$

$$\lambda_{nT0pFMCCR}[10^{10}]$$

$$\lambda_{pT0nFMCCR}[10^{10}]$$

$$\lambda_{nT0pFMCCR}[6.061 \times 10^9]$$

$$\lambda_{pT0nFMCCR}[6.061 \times 10^9]$$

Out[463]=

$$-6.2187441 \times 10^6$$

Out[464]=

$$-5.2222519 \times 10^6$$

Out[465]=

$$-12.742772$$

Out[466]=

$$-2.067764$$

Out[467]=

$$-0.95780007$$

Out[468]=

$$-0.03905163$$

In[469]:=

$$\lambda_{nT0pFMCCR}[4.024 \times 10^7]$$

$$\lambda_{pT0nFMCCR}[4.02456 \times 10^7]$$

General : 12.060916748163941533 $\times 10^{-311}$ is too small to represent as a normalized machine number;
precision may be lost. [i](#)

General : 12.6151356063387914927 $\times 10^{-318}$ is too small to represent as a normalized machine number;
precision may be lost. [i](#)

General : 13.1820165416327725971 $\times 10^{-325}$ is too small to represent as a normalized machine number;
precision may be lost. [i](#)

General : Further output of General::munfl will be suppressed during this calculation. [i](#)

Out[469]=

$$-0.0036428412$$

General : 12.4678784824760596415 $\times 10^{-316}$ is too small to represent as a normalized machine number;
precision may be lost. [i](#)

General : 12.4904562500929507625 $\times 10^{-316}$ is too small to represent as a normalized machine number;
precision may be lost. [i](#)

General : 12.5578733527950520550 $\times 10^{-317}$ is too small to represent as a normalized machine number;
precision may be lost. [i](#)

General : Further output of General::munfl will be suppressed during this calculation. [i](#)

Out[470]=

$$1.5675753 \times 10^{-168}$$

Plots of the finite mass corrections

We plot the amplitude of the finite mass corrections. This is essentially similar to Fig 8.1 of [Kernan] but the result is a little different since here we have put all the finite nucleon mass effects, and [Kernan] did not.

In[471]:=

```

If[$PaperPlots,
  Si
    TabδλFMnT0p = Table[
      
$$\left\{ T, \frac{(\lambda nT0pFMCCR[T] * (\tau_{neutron} \lambda RadandFM)^{-1})}{(\lambda nT0pBORN[T] * (\tau_{neutron} \lambda BORN)^{-1})} \right\}, \{T, ListTRange[10^9, 10^{11}]\} \};$$

      TabδλFMpT0n = Table[
      
$$\left\{ T, \frac{(\lambda pT0nFMCCR[T] * (\tau_{neutron} \lambda RadandFM)^{-1})}{(\lambda pT0nBORN[T] * (\tau_{neutron} \lambda BORN)^{-1})} \right\}, \{T, ListTRange[10^9, 10^{11}]\} \};$$

      TFreeze = 0.8 MeV / kB;

      PlotdeltaGammaFM =
        Show[ListLogLinearPlot[{TabδλFMnT0p, TabδλFMpT0n}, Frame → True,
          
$$\text{FrameStyle} \rightarrow \text{Thickness}[0.004], \text{Joined} \rightarrow \text{True}, \text{PlotRange} \rightarrow \{-10^{-1}, 10^{-2}\},$$

          
$$\text{FrameLabel} \rightarrow \{"T \text{ (K)}", "\delta T/T"\}, \text{LabelStyle} \rightarrow \{\text{FontSize} \rightarrow 12\}, \text{PlotStyle} \rightarrow$$

          
$$\{\{\text{Black}, \text{Thickness}[0.003]\}, \{\text{Black}, \text{Dashing}\{0.01\}\}, \text{Thickness}[0.003]\},$$

          
$$\text{GridLines} \rightarrow \{\{\{TFreeze, \{Gray, \text{Thickness}[0.005]\}\}, \{\}\},$$

          
$$\text{FrameTicks} \rightarrow \text{MyFrameTicksLog},$$

          
$$\text{Graphics}[\{\text{Rotate}[\text{Text}[\text{Style}["0.8 MeV", \text{FontSize} \rightarrow 10, \text{Black}],$$

          
$$\{\text{Log}@TFreeze - .1, -0.06\}], 90 \text{ Degree}\}]]$$

        ]
      If[$PaperPlots, Export["Plots/PlotdeltaGammaFM.pdf",
        Style[PlotdeltaGammaFM, Magnification → 1], "PDF"]];

```

In[473]:=

```

If[$PaperPlots,
  si
  TabδλFMnT0p2 = Table[{T, Identity[(λnT0pFMCCR[T] * (τneutron λRadandFM)-1) /
    (λnT0pBORN[T] * (τneutron λBORN)-1)]}, {T, ListTRange[5 × 108, 1010.5]}];
  TabδλFMpT0n2 = Table[{T, Identity[(λpT0nFMCCR[T] * (τneutron λRadandFM)-1) /
    (λpT0nBORN[T] * (τneutron λBORN)-1)]}, {T, ListTRange[5 × 108, 1010.5]}];

  PlotdeltaGammaFM2 = Show[ListPlot[{TabδλFMnT0p2, TabδλFMpT0n2}, Frame → True,
    mon... tracé de liste
    cadre
    vrai
    FrameStyle → Thickness[0.004], Joined → True, PlotRange → {-3 × 10-2, 10-2},
    style de cadre
    épaisseur
    joint
    vrai
    zone de tracé
    FrameLabel → {"T (1010K)", "δT/T"}, LabelStyle → {FontSize → 12}, PlotStyle →
    étiquette de cadre
    style d'étiquette
    taille de police de ca...
    style de tracé
    {{Red, Thickness[0.003]}, {Blue, Dashing[{0.01}], Thickness[0.003]}},
    rouge
    épaisseur
    bleu
    style de tirets
    épaisseur
    FrameTicks → {{Automatic, Automatic}, {{0.5 × 1010, ".5"}, {1010, "1"},
    graduations de cadre
    automatique
    automatique
    {1.5 × 1010, "1.5"}, {2 × 1010, "2"}, {2.5 × 1010, "2.5"}}, Automatic}},
    automatique
    GridLines → {{TFreeze, {Gray, Thickness[0.005]}}, {}},
    lignes de grille
    gris
    épaisseur
    Graphics[{Rotate[Text[Style["0.8 MeV", FontSize → 10, Black],
    graphique
    fais pivo...
    texte
    style
    taille de police de ...
    noir
    {TFreeze 0.93, -0.02}], 90 Degree}]]
    degré
  ]

  If[$PaperPlots, Export["Plots/PlotdeltaGammaFM2.pdf",
    si
    exporte
    Style[PlotdeltaGammaFM2, Magnification → 1], "PDF"]];
    style
    agrandissement
    fonction de densité de probabilité

```

Checking detailed balance

Born approximation detailed balance checks.

In[475]:=

```

DetailedBalanceRatio0[T_] := Exp[- $\frac{Q}{(k_B T)}$  - ξv];
    l'exposant

```

At Born order, detailed balance is of course satisfied by construction.

In[476]:=

```

DetailedBalance0[T_] :=
  (λnT0pBORN[T]) / (λpT0nBORN[T]) * DetailedBalanceRatio0[T];
DetailedBalanceCheatNeutrino0[T_] :=
  (λnT0pBORNHeatNeutrino[T]) / (λpT0nBORNHeatNeutrino[T]) *
  DetailedBalanceRatio0[T];

```

If we enforce $T_\nu = T$ the precision is insane. It is numerical precision because detailed balance is rooted in our method.

In[478]:=

```

If[$PaperPlots,
  ListLogLinearPlot[Table[
    {T, DetailedBalanceCheatNeutrino0[T] - 1}, {T, ListTRange[10^9, 10^12]}],
    Joined → True, PlotStyle → {Black, {Black, Dashed}}, Frame → True]
]

```

Including corrections in Q / mp ,
detailed balance is given by (where α should be set to 0).

In[479]:=

```

DetailedBalanceRatio[T_] := Exp[- $\frac{Q}{(k_B T)}$  - ξv]  $\left(1 + (1 + \alpha) \frac{Q}{mp}\right)^{3/2}$ ;

```

[Lopez&Turner 1997] define some quantity to parameterize how good detailed balance is obtained, this is the parameter α .

α should be 0 so we use it to estimate the failure of detailed balance

We first solve for α correctly, and then we do it when forcing the temperature of neutrinos to be the one of photons, in which case detailed balance must be true. Indeed if we do not force the temperature of neutrinos to be the one of photons, then it is only true before electrons/positrons annihilate, and at low temperature detailed balance is not satisfied.

In[480]:=

```

Clear[αLopez, αLopezCheatNeutrino]
αLopez[T_] := αLopez[T] =
  α /. Solve[Normal@Series[(λnT0pBORN[T] + λnT0pFMNoCCR[T]) / (λpT0nBORN[T] +
    λpT0nFMNoCCR[T]) * DetailedBalanceRatio[T], {α, 0, 1}] == 1, α][[1]]

αLopezCheatNeutrino[T_] := αLopezCheatNeutrino[T] = α /.
  Solve[Normal@Series[(λnT0pBORNHeatNeutrino[T] + λnT0pCheatNeutrinoFM[T]) /
    (λpT0nBORNHeatNeutrino[T] + λpT0nCheatNeutrinoFM[T]) *
    DetailedBalanceRatio[T], {α, 0, 1}] == 1, α][[1]]

```

We plot something similar to Fig. 4 of [Lopez&Turner 1997]. The remaining error should come from

second order corrections. For instance at low temperature it remains an effect of order $(Q/mn)^2$ so α is expected to differ from a null value by something of order $Q/mn=0.002$. We also see that below 5×10^{10} K, detailed balance does not work because neutrinos no longer have the temperature of photons but it works if we force neutrinos to be at the temperature of photons.

In[483]:=

```

If[$PaperPlots, PlotDetailedBalance = Show[
  ListLogLinearPlot[{Table[{T,  $\alpha$ Lopez[T]}], {T, ListTRange[10^8.5, 10^11.5]}]},
    Table[{T,  $\alpha$ LopezCheatNeutrino[T]}, {T, ListTRange[10^8.5, 10^11.5]}]}],
  Frame → True, FrameTicks → MyFrameTicksLog,
  GridLines → {{TFreeze, {Gray, Thickness[0.005]}}, {}},
  Joined → True, FrameLabel → {"T (K)", " $\alpha$ "},
  FrameStyle → Thickness[0.003], LabelStyle → {FontSize → 12},
  PlotRange → {-1, 1}, PlotStyle → {{Red}, {Black, Dashed, Blue}},
  Graphics[{Rotate[Text[Style["0.8 MeV", FontSize → 10, Black],
    {Log@TFreeze - 0.2, 0.5}], 90 Degree]}]]
]

If[$PaperPlots, Export["Plots/PlotDetailedBalance.pdf",
  Style[PlotDetailedBalance, Magnification → 1], "PDF"]];

```

Radiative Corrections (T=0)

The CCR corrected rates are described in details in the companion paper.

When electron and protons are on the same side we use the Fermi function to apply the Coulomb corrections.

For all processes the radiative correction are a factor $(1 + \frac{\alpha}{2\pi} C)$, (similarly to Eq. 18 of [Lopez&Turner 1998] or Eq. 2.13 of [Dicus .et. al]) as described in companion paper.

In[485]:=

```

IPENDpCCR[pe_, x_, znu_, sgnq_] :=
  IPENDpFromxCcr[enOfpe[pe, x], pe, x, If[$TnuEqualT, x, znu], sgnq, x]

```


In[486]:=

```
λnT0pCCR[Tv_] := IntegrateRatedp[IPENDpCCR, 1, Tv];
λpT0nCCR[Tv_] := IntegrateRatedp[IPENDpCCR, -1, Tv];
```

Plots of the radiative corrections

In[488]:=

```
If[$PaperPlots, TabδλnT0p = Table[
$$\left\{ T, \frac{(\lambda n T 0 p C C R[T] * (\tau_{\text{neutron}} \lambda_{\text{RadandFM}})^{-1})}{(\lambda n T 0 p B O R N[T] (\tau_{\text{neutron}} \lambda_{\text{BORN}})^{-1})} - 1 \right\},$$
  

  {T, ListTRange[10^8.5, 10^10.5]}];  

  TabδλpT0n = Table[
$$\left\{ T, \frac{(\lambda p T 0 n C C R[T] * (\tau_{\text{neutron}} \lambda_{\text{RadandFM}})^{-1})}{(\lambda p T 0 n B O R N[T] (\tau_{\text{neutron}} \lambda_{\text{BORN}})^{-1})} - 1 \right\},$$
  

  {T, ListTRange[10^8.5, 10^10.5]}];  

  PlotdeltaGammaCCR = Show[  

    ListLogLinearPlot[{TabδλnT0p, TabδλpT0n}, Frame → True,  

    FrameStyle → Thickness[0.004], Joined → True, FrameLabel → {"T (K)", "δΓ/Γ"},  

    LabelStyle → {FontSize → 12}, PlotStyle → {{Red}, {Blue, Dashing[{0.01}]}}},  

    GridLines → {{TFreeze, {Gray, Thickness[0.004]}}, {}},  

    FrameTicks → {{Automatic, Automatic}, {{Log[10^8.5], "10^8.5"},  

    {Log[10^9], "10^9"}, {Log[10^9.5], "10^9.5"}, {Log[10^10], "10^10"},  

    {Log[10^10.5], "10^10.5"}, {Log[10^11], "10^11"}, Automatic}}},  

    Graphics[{Rotate[Text[Style["0.8 MeV", FontSize → 10, Black],  

    {Log@TFreeze - 0.1, -0.01}], 90 Degree]}]  

  ]  

  If[$PaperPlots, Export["Plots/PlotdeltaGammaCCR.pdf",  

    Style[PlotdeltaGammaCCR, Magnification → 1], "PDF"]];  

  (*ListPlot[{TabδλnT0p, TabδλpT0n}, Frame → True, Joined → True,  

  FrameLabel → {"T (K)", "δΓ/Γ"}, PlotStyle → {Black, {Black, Dashing[{0.01}]}}]  

  Export["Plots/NoLogPlotdeltaGammaCCR.pdf", Style[%, Magnification → 1], "PDF"];*)
```

Finite-temperature Radiative Corrections

Method

We use exclusively [Brown&Sawyer] for the finite temperature radiative corrections, but the equations are gathered and summarized in companion paper essentially in section III.F and related appendix. The various terms are in Eq. 108.

Note also that on top of that, we also need to add what we called Brehmstrahlung corrections (Eqs. 107 in companion paper)

Real photons processes integrand

Bose Einstein distribution function (and if $s_q = -1$ it is stimulated emission)

In[490]:=

```
BEQ[en_, sq_] := sq BE[sq en];
```

$\tilde{\chi}$ is defined in companion paper in Eq. B45.

In[491]:=

```
xtilde[en_, znu_, sgnq_] :=  
  With[{q = Q / me}, FDv[en - sgnq q, sgnq ξv, znu] (en - sgnq q) ^ 2]  
  avec
```

We first compute real photons processes (absorption or stimulated emission)

We put Fermi function everywhere to be consistent.

The integrand is (Eq. 109 of companion paper)

In[492]:=

```

IPENCCRT[en_, k_, x_, znu_, sgnq_] := With[{p = Sqrt[en2 - 1]},
  With[{b = p / en, A = (2 en2 + k2) Log[ $\frac{en + p}{en - p}$ ] - 4 p en, B = 2 en Log[ $\frac{en + p}{en - p}$ ] - 4 p},
     $\frac{\alpha_{FS}}{2 \pi} * \left( \frac{BE[x k]}{k} \right) *$ 
    (A (FD[-en, x] × Fermi[sgnq, 1, b] (χtilde[en - k, znu, sgnq] + χtilde[
      en + k, znu, sgnq] - 2 χtilde[en, znu, sgnq]) +
      FD[en, x] × Fermi[sgnq, -1, b] (χtilde[-en + k, znu, sgnq] +
      χtilde[-en - k, znu, sgnq] - 2 χtilde[-en, znu, sgnq]))
    - k B * (FD[-en, x] × Fermi[sgnq, 1, b] (χtilde[en - k, znu, sgnq] -
      χtilde[en + k, znu, sgnq]) + FD[en, x] × Fermi[sgnq, -1, b]
      (χtilde[-en + k, znu, sgnq] - χtilde[-en - k, znu, sgnq]))
    )
  ]];

(* Compiled version to compute the integrals slightly faster *)
IPENCCRTC =
MyCompile[{{en, _Real}, {k, _Real}, {x, _Real}, {znu, _Real}, {sgnq, _Integer}},
  Evaluate[IPENCCRT[en, k, x, znu, sgnq]]];
IPENCCRTC[en_?NumericQ, k_, x_, znu_, sgnq_] := IPENCCRTC[en, k, x, znu, sgnq]

```

Bremsstrahlung corrections (needed to obtain all real photons processes and eventually detailed balance).

The integrand is (Eqs. B48 and B49 using definitions B41)

In[494]:=

```

Clear[IPENCCRDiffBremsstrahlungCN,
efface
IPENCCRDiffBremsstrahlungC, IPENCCRDiffBremsstrahlung]
IPENCCRDiffBremsstrahlung[en_, k_, x_, znu_, sgnq_] :=
  With[avec {p = Sqrt[en2 - 1], q = Q / me},
racine carrée
avec {b = p / en, A = (2 en2 + k2) Log[logarithme  $\frac{en + p}{en - p}$ ] - 4 p en, B = 2 en Log[logarithme  $\frac{en + p}{en - p}$ ] - 4 p},
avec {Fp = A + k B, Fm = A - k B},

$$\frac{\alpha FS}{2 \pi k}$$

  ((FD[-en, x] × Fermi[sgnq, 1, b] (Fp  $\chi$ tilde[en + k, znu, sgnq] - If[k < Abs[
si valeur absolue
    en - sgnq q], Fp FD[en - sgnq q, znu] (Abs[en - sgnq q] - k)2, 0]))
  + (FD[en, x] × Fermi[sgnq, -1, b]
    (Fm  $\chi$ tilde[-en + k, znu, sgnq] - If[k < Abs[en + sgnq q],
si valeur absolue
      Fp FD[-en - sgnq q, znu] (Abs[en + sgnq q] - k)2, 0)))
valeur absolue
  )
  ]];

(* We compile for the integration *)
IPENCCRDiffBremsstrahlungC =
  MyCompile[{{en, nombre réel _Real}, {k, nombre réel _Real}, {x, nombre réel _Real}, {znu, nombre réel _Real}, {sgnq, nombre entier _Integer}},
  Evaluate[IPENCCRDiffBremsstrahlung[en, k, x, znu, sgnq]]];
évalue
IPENCCRDiffBremsstrahlungCN[en_?NumericQ, k_, x_, znu_, sgnq_] :=
expression numérique ?
  IPENCCRDiffBremsstrahlungC[en, k, x, znu, sgnq]

```

The expression IPENFiveBodyT0 implements the integrand of 5.21 of [Brown&Sawyer] (and not 5.20. Careful).

We showed in companion paper that this is only a part of the bremsstrahlung corrections needed.

In[497]:=

```

IPENFiveBodyT0[en_, k_, x_, znu_, sgnq_] := With[{p = Sqrt[en2 - 1]},
  With[{A = (2 en2 + k2) Log[ $\frac{en + p}{en - p}$ ] - 4 p en, B = 2 en Log[ $\frac{en + p}{en - p}$ ] - 4 p},
    
$$\frac{\alpha FS}{2 \pi k} (FD[en, x]) \chi_{\text{tilde}}[-en + k, znu, sgnq] (A - k B) ] ]$$


(* Compiled version *)
IPENFiveBodyT0C =
  Compile[{{en, _Real}, {k, _Real}, {x, _Real}, {znu, _Real}, {sgnq, _Integer}},
    Evaluate[With[{p = Sqrt[en2 - 1]},
      With[{A = (2 en2 + k2) Log[ $\frac{en + p}{en - p}$ ] - 4 p en, B = 2 en Log[ $\frac{en + p}{en - p}$ ] - 4 p},
        
$$\frac{\alpha FS}{2 \pi k} (-BE[-k x]) * (FD[en, x]) \chi_{\text{tilde}}[-en + k, znu, sgnq] (A - k B) ] ] ]],
    "RuntimeOptions" -> "Speed", CompilationTarget -> "C"];
IPENFiveBodyT0CN[en_?NumericQ, k_, x_, znu_, sgnq_] :=
  IPENFiveBodyT0C[en, k, x, znu, sgnq]$$

```

Mass shift and pe+ee integrand

We finally consider the mass shift and ep + ee corrections

We split the contributions of the last two terms of 5.14 in [Brown and Sawyer] in two parts.

Indeed these are given by 5.15 [Brown and Sawyer]. The first part has only one integral, while the second part has two integrals.

The integrand of first part with only one integral is C1dE, and the integrand of the second part with two integrals is C2dE1dE2 (see 5.16 [Brown and Sawyer]).

This is the simple integration in Eq. B50a of companion paper (more precisely the integrand and the integration itself is performed further below)

In[500]:=

```

C1dE[en_, x_, znu_, sgnq_] := With[{pe = Sqrt[en2 - 1], q = Q / me},
  - 
$$\frac{\alpha FS en}{2 \pi pe} * \frac{2 \pi^2}{3 x^2} (\chi[en, pe, x, znu, sgnq] + \chi[-en, pe, x, znu, sgnq]) ] ;$$

```

NB : L is given by 5.18 [Brown and Sawyer]

And this is the double integration integrand of Eq.B50a of companion paper

In[501]:=

```

C2dE1dE2[e1_, e2_, x_, znu_, sgnq_] := With[
  {p1 =  $\sqrt{e1^2 - 1}$ , p2 =  $\sqrt{e2^2 - 1}$ , q = Q / me }, With[{L =  $\text{Log}\left[\frac{e1 e2 + p1 p2 + 1}{e1 e2 - p1 p2 + 1}\right]$ },
    
$$\frac{\alpha_{FS}}{2 \pi} (\chi[e1, p1, x, znu, sgnq] + \chi[-e1, p1, x, znu, sgnq])$$

    * 
$$\left(-\frac{1}{4} \text{Log}\left[\left(\frac{p1 + p2}{p1 - p2}\right)^2\right]^2\right) *$$

    
$$\left(FDp[e2, x] \frac{p2}{p1} \frac{e1^2}{e2} (e1 + e2) + FD[e2, x] \frac{e1^2}{p1 p2} \left(e2 + \frac{e1}{e2^2}\right)\right)$$

    + 
$$\text{Log}\left[\left(\frac{p1 + p2}{p1 - p2}\right)^2\right] \left(FDp[e2, x] \left(p2^2 \frac{e1}{e2} \left(\frac{1}{p1^2} + 2\right) - e1^2 \frac{p2}{p1} L\right) +\right.$$

    
$$FD[e2, x] \left(\frac{e1}{p1^2 e2^2} (e2^2 + 2 p1^2 + 1) - \frac{(e1^2 + e2^2)}{(e1 + e2)} - \frac{e1^2 e2}{p1 p2} L\right)$$

    - 
$$FD[e2, x] \left(4 e1 \frac{p2}{p1} + 2 e2 L\right)$$

  ]];

(* Compiled version *)
C2dE1dE2C =
Compile[{{e1, _Real}, {e2, _Real}, {x, _Real}, {znu, _Real}, {sgnq, _Integer}},
  Evaluate[With[{p1 =  $\sqrt{e1^2 - 1}$ , p2 =  $\sqrt{e2^2 - 1}$ , q = Q / me },
    With[{L =  $\text{Log}\left[\frac{e1 e2 + p1 p2 + 1}{e1 e2 - p1 p2 + 1}\right]$ },
      
$$\frac{\alpha_{FS}}{2 \pi} (\chi[e1, p1, x, znu, sgnq] + \chi[-e1, p1, x, znu, sgnq])$$

      * 
$$\left(-\frac{1}{4} \text{Log}\left[\left(\frac{p1 + p2}{p1 - p2}\right)^2\right]^2\right) *$$

      
$$\left(FDp[e2, x] \frac{p2}{p1} \frac{e1^2}{e2} (e1 + e2) + FD[e2, x] \frac{e1^2}{p1 p2} \left(e2 + \frac{e1}{e2^2}\right)\right)$$

      + 
$$\text{Log}\left[\left(\frac{p1 + p2}{p1 - p2}\right)^2\right] \left(FDp[e2, x] \left(p2^2 \frac{e1}{e2} \left(\frac{1}{p1^2} + 2\right) - e1^2 \frac{p2}{p1} L\right) +\right.$$


```

```

      FD[e2, x]  $\left( \frac{e1}{p1^2 e2^2} (e2^2 + 2 p1^2 + 1) - \frac{(e1^2 + e2^2)}{(e1 + e2)} - \frac{e1^2 e2}{p1 p2} L \right)$ 
      - FD[e2, x]  $\left( 4 e1 \frac{p2}{p1} + 2 e2 L \right)$ 
    ]]], "RuntimeOptions" → "Speed", CompilationTarget → "C"];
    [options de durée d'exécution] [cible de compilation] [constante C]
C2dE1dE2CN[e1_?NumericQ, e2_?NumericQ, x_, znu_, sgnq_] :=
    [expression numéri...] [expression numérique ?]
C2dE1dE2C[e1, e2, x, znu, sgnq];

```

Integrations on momenta

We perform all integrations now that we have expressed their integrands.

TruePhoton -> real photon processes

Diffbremsstrahlung -> bremsstrahlung corrections

Thermal -> mass shift and pe+ee corrections

Let us start with n -> p processes

In[504]:=

```

Clear[λnT0pThermal, λnT0pThermalTruePhoton,
[efface
λpT0nThermalTruePhoton, λnT0p5bodies, λpT0nThermal,
λnT0pThermalDiffBremsstrahlung, λpT0nThermalDiffBremsstrahlung]

```

In[505]:=

```

λnT0pThermalTruePhoton[Tv_] := (*λnT0pThermalTruePhoton[Tv]=*)
  With[ $\left\{x = \frac{me}{(kB Tv)}, znu = \frac{me}{(kB Tv Tv_{\text{overT}}[Tv])}, q = Q / me\right\}$ ,
     $\int_{\text{avec}}$ 
    NIntegrate[IPENCCRTC[en, k, x, If[$TnuEqualT, x, znu], 1],
       $\int_{\text{intégrer numériquement}}$ 
      {k, 0.001,  $\text{Max}[10, 20 / x]$ }, {en, 1.001,  $\text{Max}[10, 20 / x]$ }, PrecisionGoal → 4]
     $\int_{\text{si}}$ 
    {en, 1.001,  $\text{Max}[10, 20 / x]$ },
       $\int_{\text{maximum}}$ 
      {k, 0.001,  $\text{Abs}[en - q]$ ,  $\text{Abs}[en + q]$ ,  $\text{Max}[10, 20 / x]$ }, PrecisionGoal → 4]
     $\int_{\text{objectif de précision}}$ 
  ];
λnT0pThermalDiffBremsstrahlung[Tv_] :=
  (*λnT0pThermalDiffBremsstrahlung[Tv]=*)
  With[ $\left\{x = \frac{me}{(kB Tv)}, znu = \frac{me}{(kB Tv Tv_{\text{overT}}[Tv])}, q = Q / me\right\}$ ,
     $\int_{\text{avec}}$ 
    NIntegrate[IPENCCRDiffBremsstrahlungCN[en, k, x, If[$TnuEqualT, x, znu], 1],
       $\int_{\text{intégrer numériquement}}$ 
      {en, 1.001,  $\text{Max}[10, 20 / x]$ },
       $\int_{\text{si}}$ 
      {k, 0.001,  $\text{Abs}[en - q]$ ,  $\text{Abs}[en + q]$ ,  $\text{Max}[10, 20 / x]$ }, PrecisionGoal → 4]
       $\int_{\text{valeur absolue}}$   $\int_{\text{valeur absolue}}$   $\int_{\text{maximum}}$   $\int_{\text{objectif de précision}}$ 
    ];

```

The global 1/2 factor is Jacobian of the change of variables (we integrate in the sum and difference of energies).

In[507]:=

```

λnT0pThermal[Tv_] := (*λnT0pThermal[Tv]=*)
  With[ $\left\{x = \frac{me}{(kB Tv)}, znu = \frac{me}{(kB Tv Tv_{\text{overT}}[Tv])}, q = Q / me\right\}$ ,
     $\int_{\text{avec}}$ 
    NIntegrate[C1dE[en, x, If[$TnuEqualT, x, znu], 1], {en, 1,  $\text{Max}[25, 150 / x]$ }]
     $\int_{\text{intégrer numériquement}}$ 
    + NIntegrate[1 / 2 C2dE1dE2CN[(e1pe2 + e1me2) / 2, (e1pe2 - e1me2) / 2, x,
      If[$TnuEqualT, x, znu], 1], {e1me2, - $\text{Max}[10, 15 / x]$ , -0.001},
       $\int_{\text{si}}$ 
      {e1pe2, 2.001 +  $\text{Abs}[e1me2]$ , 2 +  $\text{Abs}[e1me2]$  +  $\text{Max}[10, 15 / x]$ },
       $\int_{\text{valeur absolue}}$   $\int_{\text{valeur absolue}}$   $\int_{\text{maximum}}$ 
      PrecisionGoal → 3, Exclusions → {0}]
     $\int_{\text{objectif de précision}}$   $\int_{\text{exclusions}}$ 
    + NIntegrate[1 / 2 C2dE1dE2CN[(e1pe2 + e1me2) / 2, (e1pe2 - e1me2) / 2, x,
      If[$TnuEqualT, x, znu], 1], {e1me2, 0.001,  $\text{Max}[10, 15 / x]$ }, {e1pe2,
       $\int_{\text{si}}$ 
      2.001 +  $\text{Abs}[e1me2]$ , 2 +  $\text{Abs}[e1me2]$  +  $\text{Max}[10, 15 / x]$ }, PrecisionGoal → 3]
       $\int_{\text{valeur absolue}}$   $\int_{\text{valeur absolue}}$   $\int_{\text{maximum}}$   $\int_{\text{objectif de précision}}$ 
    ];

```


The 5 body is just for comparison with [Brown & Sawyer]

In[508]:=

```
λnT0p5bodies[Tv_] := (*λnT0p5bodies[Tv]=*)
  With[ $\left\{x = \frac{me}{(kB Tv)}, znu = \frac{me}{(kB Tv Tv_{overT}[Tv])}, q = Q / me\right\}$ ,
     $\int_{\text{avec}}$ 
    NIntegrate[IPENFiveBodyT0CN[en, k, x, If[$TnuEqualT, x, znu], 1],
       $\int_{\text{intégrer numériquement}}$ 
      {en, 1, Max[20, 20 / x]}, {k, en + q, en + q + Max[20, 20 / x]}, PrecisionGoal → 4]
     $\int_{\text{si}}$ 
    ];
```

Let us finish with p -> n processes

When we have computed the corrections to n -> p rates, the converse are obtained from detailed balance arguments. That is we perform the replacement $Q \rightarrow -Q$ as argued in [Brown&Sawyer]. This is also explained in details in the companion paper.

In[509]:=

```
λpT0nThermalTruePhoton[Tv_] := (*λpT0nThermalTruePhoton[Tv]=*) If[Tv < 10^8.2,
   $\int_{\text{si}}$ 
  (* When the temperature is too low it is better to put 0 *), 0,
  With[ $\left\{x = \frac{me}{(kB Tv)}, znu = \frac{me}{(kB Tv Tv_{overT}[Tv])}, q = Q / me\right\}$ ,
     $\int_{\text{avec}}$ 
    NIntegrate[IPENCCRTC�[en, k, x, If[$TnuEqualT, x, znu], -1],
       $\int_{\text{intégrer numériquement}}$ 
      {k, 0.001, Max[10, 20 / x]}, {en, 1.001, Max[10, 20 / x]}, PrecisionGoal → 4]
    ];

λpT0nThermalDiffBremsstrahlung[Tv_] :=
  (*λpT0nThermalDiffBremsstrahlung[Tv]=*) If[Tv < 10^8.2,
     $\int_{\text{si}}$ 
    (* When the temperature is too low it is better to put 0 *), 0,
    With[ $\left\{x = \frac{me}{(kB Tv)}, znu = \frac{me}{(kB Tv Tv_{overT}[Tv])}, q = Q / me\right\}$ ,
       $\int_{\text{avec}}$ 
      NIntegrate[IPENCCRDiffBremsstrahlungCN[en, k,
         $\int_{\text{intégrer numériquement}}$ 
        x, If[$TnuEqualT, x, znu], -1], {en, 1.001, Max[10, 20 / x]},
         $\int_{\text{si}}$ 
        {k, 0.001, Abs[en - q], Abs[en + q], Max[10, 20 / x]}, PrecisionGoal → 4]
         $\int_{\text{valeur absolue}}$   $\int_{\text{valeur absolue}}$   $\int_{\text{maximum}}$   $\int_{\text{objectif de précision}}$ 
      ];

λpT0nThermal[Tv_] := (*λpT0nThermal[Tv]=*) If[Tv < 10^8.2,
   $\int_{\text{si}}$ 
```

```

(* When the temperature is too low it is better to put 0 *), 0,
With[ $\left\{x = \frac{me}{(kB Tv)}, znu = \frac{me}{(kB Tv Tv_{overT}[Tv])}, q = Q / me\right\}$ ,
  _avec
  NIntegrate[
    _intégrer numériquement
    C1dE[en, x, If[$TnuEqualT, x, znu], -1], {en, 1, Max[25, 150 / x]}]
    _si _maximum
  + NIntegrate[1 / 2 C2dE1dE2CN[(e1pe2 + e1me2) / 2, (e1pe2 - e1me2) / 2, x,
    _intégrer numériquement
    If[$TnuEqualT, x, znu], -1], {e1me2, -Max[10, 15 / x], -0.001}, {e1pe2,
    _si _maximum
    2.001 + Abs[e1me2], 2 + Abs[e1me2] + Max[10, 15 / x]}, PrecisionGoal → 3]
    _valeur absolue _valeur absolue _maximum _objectif de précision
  + NIntegrate[1 / 2 C2dE1dE2CN[(e1pe2 + e1me2) / 2, (e1pe2 - e1me2) / 2, x,
    _intégrer numériquement
    If[$TnuEqualT, x, znu], -1], {e1me2, 0.001, Max[10, 15 / x]}, {e1pe2,
    _si _maximum
    2.001 + Abs[e1me2], 2 + Abs[e1me2] + Max[10, 15 / x]}, PrecisionGoal → 3]
    _valeur absolue _valeur absolue _maximum _objectif de précision
  ]];

```

Collecting all finite-temperature corrections

We collect the various contributions of Eq. 108. The TruePhoton contribution refers to the first term on the rhs of Eq. 108, and the Thermal contribution to the second and third. The Bremsstrahlung corrections are Eqs. 107.

In[512]:=

```

λnT0pCCRTh[Tv_] := (λnT0pThermal[Tv] +
  λnT0pThermalTruePhoton[Tv] + If[$CorrectionBremsstrahlung,
    _si
    λnT0pThermalDiffBremsstrahlung[Tv], λnT0p5bodies[Tv]]);

λpT0nCCRTh[Tv_] := (λpT0nThermal[Tv] + λpT0nThermalTruePhoton[Tv] +
  If[$CorrectionBremsstrahlung, λpT0nThermalDiffBremsstrahlung[Tv], 0]);
  _si

```

We gather all corrections according to the Booleans chosen at the beginning

In[514]:=



```

λ0 := If[$RadiativeCorrections, λRad, λBORN] + If[$FiniteNucleonMass, λFM, 0]
  _si _si

```

In[515]:=

```
λnT0pCCRTh[10^9]
λnT0pCCR[10^9]
λnT0pBORN[10^9]
```

 **NIntegrate** : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration is 0, highly oscillatory integrand, or WorkingPrecision too small. 

Out[515]=

0.0002091131

Out[516]=

1.7777578

Out[517]=

1.6546389

Gathering all corrections

For info the normalized rates are :

In[518]:=

```
λnT0pNormalizedSDCCR[Tv_] := (λ0)^-1 λnT0pSDCCR[Tv];
λpT0nNormalizedSDCCR[Tv_] := (λ0)^-1 λpT0nSDCCR[Tv];
λnT0pNormalizedSD[Tv_] := (λ0)^-1 λnT0pSD[Tv];
λpT0nNormalizedSD[Tv_] := (λ0)^-1 λpT0nSD[Tv];
λnT0pNormalizedCCR[Tv_] := (λ0)^-1 λnT0pCCR[Tv];
λpT0nNormalizedCCR[Tv_] := (λ0)^-1 λpT0nCCR[Tv];
λnT0pNormalizedBORN[Tv_] := (λ0)^-1 λnT0pBORN[Tv];
λpT0nNormalizedBORN[Tv_] := (λ0)^-1 λpT0nBORN[Tv];
```

We add them depending on options chosen

In[526]:=

```

Clear[λnT0p, λpT0n, λnT0pNormalized, λpT0nNormalized];
efface
λnT0pNormalized[Tv_] := (λ0)-1 (
  If[$RadiativeCorrections, λnT0pCCR[Tv], λnT0pBORN[Tv]]
si
  + If[$RadiativeThermal, λnT0pCCRTh[Tv], 0]
si
  + If[$IncompleteNeutrinoDecoupling && $SpectralDistortions,
si
    If[$RadiativeCorrections, λnT0pSDCCR[Tv], λnT0pSD[Tv]], 0]
si
  + If[$FiniteNucleonMass,
si
    If[$CoupledFMandRC, λnT0pFMCCR[Tv], λnT0pFMNoCCR[Tv]], 0]
si
);

λpT0nNormalized[Tv_] := (λ0)-1 (
  If[$RadiativeCorrections, λpT0nCCR[Tv], λpT0nBORN[Tv]]
si
  + If[$RadiativeThermal, λpT0nCCRTh[Tv], 0]
si
  + If[$IncompleteNeutrinoDecoupling && $SpectralDistortions,
si
    If[$RadiativeCorrections, λpT0nSDCCR[Tv], λpT0nSD[Tv]], 0]
si
  + If[$FiniteNucleonMass,
si
    If[$CoupledFMandRC, λpT0nFMCCR[Tv], λpT0nFMNoCCR[Tv]], 0]);
si

```

In[529]:=

```

Clear[λnT0p]
efface
λnT0p[Tv_] := 1 / τneutron λnT0pNormalized[Tv];
λpT0n[Tv_] := 1 / τneutron λpT0nNormalized[Tv];

```

Detailed balance check

```

In[532]:=
Tt = 10^10.8;
(λnT0pThermalTruePhoton[Tt] + 1 λnT0pThermalDiffBremsstrahlung[Tt]) /
(λpT0nThermalTruePhoton[Tt] + 1 λpT0nThermalDiffBremsstrahlung[Tt])
λnT0p[Tt] / λpT0n[Tt]
λnT0pCCR[Tt] / λpT0nCCR[Tt]
λnT0pBORN[Tt] / λpT0nBORN[Tt]
λnT0pThermal[Tt] / λpT0nThermal[Tt]
Exp[Q / kB / Tt]
|exponentielle

Out[533]=
1.2685145

Out[534]=
1.2656636

Out[535]=
1.2685433

Out[536]=
1.2685439

Out[537]=
1.2685535

Out[538]=
1.2685426

```

Precomputation and storage of rates

We precompute the weak rates the first time and then we save them on the disk.

If the options \$RecomputeWeakRates is set to True, the computation is forced.

We first build the name of the file to store the PEN rates. It includes as a postfix the values of the booleans for the various effects taken or not taken into account.

```

In[539]:=
LetterFromBoolean[Bool_] := If[Bool, "T", "F"];
|si

StringFromBoolean[BoolList_List] :=
|liste
StringJoin[LetterFromBoolean /@ BoolList];
|joins chaînes de caractères

BooleanSuffix = StringFromBoolean[
{$RadiativeCorrections, $RadiativeThermal, $FiniteNucleonMass,
$CoupledFMandRC, $QEDPlasmaCorrections, $IncompleteNeutrinoDecoupling}]

Out[541]=
TTTTTT

In[542]:=
NamePENFilenp = "Interpolations/PENRatenp" <> BooleanSuffix <> ".dat";
NamePENFilepn = "Interpolations/PENRatepn" <> BooleanSuffix <> ".dat";

```

In[544]:=

```

$BornBool = Not[$RadiativeThermal] &&
  Not[$RadiativeCorrections] && Not[$FiniteNucleonMass];

```

We store the rate but without the division by τ_{neutron} . So that

we can use the same fit for the reaction rates, and still vary τ_{neutron} .

The rates in the files, once loaded, are interpolated once the division by τ_{neutron} has been added.

If spectral distortions are taken into account, we could avoid ParallelComputation because it seems it loses time in sharing the big interpolation.

TODO maybe there is a solution to that.

In[545]:=

```

MyTableWeakRate := If[$ParallelWeakRates (*&&Not[$SpectralDistortions]*),
  ParallelEvaluate[Off[NIntegrate::slwcon];];
ParallelTable, Table]

PreComputeWeakRates := (
  Off[NIntegrate::slwcon];
   $\lambda_{nT0pTab}$  = MyTableWeakRate[{T,  $\lambda_{nT0pNormalized}[T]$ }, {T, ListT}];
   $\lambda_{pT0nTab}$  = MyTableWeakRate[{T,  $\lambda_{pT0nNormalized}[T]$ }, {T, ListT}];
  TabRatepn =  $\lambda_{nT0pTab}$ ;
  TabRatepn =  $\lambda_{pT0nTab}$ ;
  On[NIntegrate::slwcon];
   $\lambda_{nT0pI}$  = MyInterpolationRate[ToExpression[TabRatepn]];
   $\lambda_{pT0nI}$  = MyInterpolationRate[ToExpression[TabRatepn]];
);

```

Importing the rates previously stored if they exist, and recompute if not.

In[547]:=

```

TabRatenp = Check[Import[NamePENFilenp, "TSV"],
  |vérifie |importe
  Print["Precomputed n -> p rate not found. We recompute
  |imprime
    the rates and store them. This can take very long"];
  $Failed, Import::nffil];
  |échoué |importe

TabRatepn = Check[Import[NamePENFilepn, "TSV"],
  |vérifie |importe
  Print["Precomputed p -> n rate not found. We recompute
  |imprime
    the rates and store them. This can take very long"];
  $Failed, Import::nffil];
  |échoué |importe

Timing[If[TabRatenp === $Failed || TabRatepn === $Failed || $RecomputeWeakRates,
|chrono... |si |échoué |échoué
  PreComputeWeakRates;
  Export[NamePENFilenp, TabRatenp, "TSV"];
  |exporte
  Export[NamePENFilepn, TabRatepn, "TSV"];,
  |exporte
  λnT0pI = MyInterpolationRate[ToExpression[TabRatenp]];
  |en expression
  λpT0nI = MyInterpolationRate[ToExpression[TabRatepn]];
  |en expression
];]

```

Out[549]=

{0.006087, Null}

We give standard names that are used in the network of reactions later. The factor $1/\tau_n$ is the one appearing in the constant defined in Eq. 91 of companion paper.

In[550]:=

```

LnT0p[Tv_] := 1 / τneutron * λnT0pI[Tv];
LpT0n[Tv_] := 1 / τneutron * λpT0nI[Tv];
LbarnT0p[Tv_] := LpT0n[Tv];

```

We check that the rate for neutron decay is what we expect at low temperature, that is it is τ_{neutron} only. At 10^8 K it should be the case.

In[553]:=

```
1 / LnT0p[Tf]
```

Out[553]=

879.40466

Comparison with [Serpico et al. 2004] weak rates fits

Uncomment if you want to compare with the fit of Parthenope, and also to find the coefficients of a

similar fit with the PRIMAT weak rates.

In[554]:=

```
(*AParthenope={1,0.15735,4.6172,-0.40520*10^2,
  0.13875*10^3,-0.59898*10^2,0.66752*10^2,-0.16705*10^2,3.8071,
  -0.39140,0.023590,-0.83696*10^-4,-0.42095*10^-4,0.17675*10^-5};
qnpParthenope=0.33979;
λnT0pParthenope[T_]=With[{z=me/(kB T)},
  1/τneutron*Exp[-qnpParthenope/z]Sum[AParthenope[[l+1]]z^(-l),{l,0,13}]]];
(* Beware of type in Serpico 2004 C2. The
  sum on the bl coefficients starts at l=0, not l=1*)
BParthenope={-0.62173,0.22211*10^2,-0.72798*10^2,0.11571*10^3,-0.11763*10^2,
  0.45521*10^2,-3.7973,0.41266,-0.026210,0.87934*10^-3,-0.12016*10^-4};
qpnParthenope=2.8602;
λpT0nParthenope[T_]=If[kB T/MeV<0.1,0, With[{z=me/(kB T)},
  1/τneutron*Exp[-qpnParthenope*z]Sum[BParthenope[[l+1]]z^(-l),{l,0,10}]]];
```

```
LogLinearPlot[
  {(λnT0pParthenope[T]/LnT0p[T]-1)*100,(λpT0nParthenope[T]/LpT0n[T]-1)*100},
  {T,10^9,10^11},Frame→True,FrameLabel→{"T (K)",""},
  PlotLabel→"Difference (in %) PRIMAT vs Parthenope's fits (Serpico 2004)",
  PlotRange→{-2,2}]*)
```

Fit of n → p rate

In[555]:=

```
(*Clear[λnT0pFit]
(* We look for a fit like C1 of Serpico et al. 2004 *)
λnT0pFit[T_]:=With[{z=me/(kB T)},Exp[-qnpFit/z]Sum[AFit[l]z^(-l),{l,0,13}]]];
datanT0p=
  Table[{Log@T,Log@λnT0pI[T]},{T,Select[ListT,kB#/MeV≥0.01&&kB#/MeV≤10&}}];
(* Select range as in Appendix C of Serpico et al. 2004 *)
VariablesAndGuessnT0p=Transpose[{Join[Table[AFit[i],{i,0,13}],{qnpFit}],
  Join[AParthenope,{qnpParthenope}]]];
```


In[556]:=

```
(*FitA=
FindFit[datanT0p,Log@Abs[λnT0pFit[Exp@LogT]],VariablesAndGuessnT0p,LogT]


```

Fit of p -> n rate

In[557]:=

```
(*Clear[λpT0nFit]
|efface
We look for a fit like C1 of Serpico et al. 2004 *)
λpT0nFit[T_] :=
With[{z = me / (kB T)}, Exp[-qpnFit * z] Sum[BFit[l] z^(-l), {l, 0, 10}]];
|avec      |exponentielle      |somme
datapT0n = Table[{Log@T, Log@λpT0nI[T]},
|table      |logarit... |logarithme
{T, Select[ListT, kB # / MeV ≥ 0.1 && kB # / MeV ≤ 10 &]}];
|sélectionne
(* Select range as in Appendix C of Serpico et al. 2004 *)
|sélectionne      |constante C
VariablesAndGuesspT0n = Transpose[{Join[Table[BFit[i], {i, 0, 10}], {qpnFit}],
|transpose      |joins |table
Join[BParthenope, {qpnParthenope}]}];
|joins
... Transpose : The first two levels of
{{BFit [0], BFit [1], BFit [2], BFit [3], BFit [4], BFit [5], BFit [6], BFit [7], BFit [8], BFit [9], <<2>>}, Join [BParthenope,
{qpnParthenope }]} cannot be transposed. ⓘ
```

In[560]:=

```
(*FitB=
FindFit[datapT0n,Log@Abs[λpT0nFit[Exp@LogT]],VariablesAndGuesspT0n,LogT]


```

Plots of finite-temperature corrections

This is very long so I comment this section but to get the plot of finite temperature corrections of the companion paper, this should be uncommented.

In[561]:=

```
(*
```

```

Off[NIntegrate::slwcon];
|dé... |intègre numériquement
TabdλmbdanT0p=
  MyTableWeakRate[{T, (λRadandFM*(λnT0pThermal[T]+λnT0pThermalTruePhoton[T]+
    λnT0pThermalDiffBremsstrahlung[T]))/
    (λBORN*λnT0pBORN[T]))},{T,ListTRange[1 10^9,10^11]}}];
Print[TabdλmbdanT0p];
|imprime

TabdλmbdapT0n=
  MyTableWeakRate[{T, (λRadandFM*(λpT0nThermal[T]+λpT0nThermalTruePhoton[T]+
    λpT0nThermalDiffBremsstrahlung[T]))/
    (λBORN*λpT0nBORN[T]))},{T,ListTRange[1 10^9,10^11]}}];

Export["Interpolations/TabdλmbdanT0p.dat",TabdλmbdanT0p,"TSV"];
|exporte
Export["Interpolations/TabdλmbdapT0n.dat",TabdλmbdapT0n,"TSV"];
|exporte

TabdλmbdanT0pBrown=MyTableWeakRate[
  {T,  $\frac{\lambda_{\text{RadandFM}}(\lambda_{\text{nT0pThermal}}[T]+\lambda_{\text{nT0pThermalTruePhoton}}[T])}{(\lambda_{\text{BORN}}*\lambda_{\text{nT0pBORN}}[T])}$ },{T,ListTRange[1 10^9,10^11]}}];
TabdλmbdanT0pBrown5Bodies=MyTableWeakRate[
  {T, (λRadandFM*(λnT0pThermal[T]+λnT0pThermalTruePhoton[T]+λnT0p5bodies[T]))/
    (λBORN*λnT0pBORN[T]))},{T,ListTRange[1 10^9,10^11]}}];
TabdλmbdapT0nBrown=MyTableWeakRate[
  {T,  $\frac{\lambda_{\text{RadandFM}}(\lambda_{\text{pT0nThermal}}[T]+\lambda_{\text{pT0nThermalTruePhoton}}[T])}{(\lambda_{\text{BORN}}*\lambda_{\text{pT0nBORN}}[T])}$ },{T,ListTRange[1 10^9,10^11]}}];

Export["Interpolations/TabdλmbdanT0pBrown.dat",TabdλmbdanT0pBrown,"TSV"];
|exporte
Export["Interpolations/TabdλmbdanT0pBrown5Bodies.dat",
|exporte
  TabdλmbdanT0pBrown5Bodies,"TSV"];
Export["Interpolations/TabdλmbdapT0nBrown.dat",TabdλmbdapT0nBrown,"TSV"];
|exporte

TabdλmbdanT0pBrehm=MyTableWeakRate[
  {T,  $\frac{\lambda_{\text{RadandFM}}(\lambda_{\text{nT0pThermalDiffBremsstrahlung}}[T])}{(\lambda_{\text{BORN}}*\lambda_{\text{nT0pBORN}}[T])}$ },{T,ListTRange[1 10^9,10^11]}}];
TabdλmbdapT0nBrehm=MyTableWeakRate[
  {T,  $\frac{\lambda_{\text{RadandFM}}(\lambda_{\text{pT0nThermalDiffBremsstrahlung}}[T])}{(\lambda_{\text{BORN}}*\lambda_{\text{pT0nBORN}}[T])}$ },{T,ListTRange[1 10^9,10^11]}}];
On[NIntegrate::slwcon];
|ac· |intègre numériquement

Export["Interpolations/TabdλmbdanT0pBrehm.dat",TabdλmbdanT0pBrehm,"TSV"];
|exporte
Export["Interpolations/TabdλmbdapT0nBrehm.dat",
|exporte

```

```
TabdλmbdapT0nBrehm,"TSV"] ;*)
```

In[562]:=

```
If[$PaperPlots,
  si
    TabδλnT0p = Import["Interpolations/TabdλmbdanT0p.dat"];
    TabδλpT0n = Import["Interpolations/TabdλmbdapT0n.dat"];

    TabδλnT0pBrown = Import["Interpolations/TabdλmbdanT0pBrown.dat"];
    TabδλnT0pBrown5Bodies =
      Import["Interpolations/TabdλmbdanT0pBrown5Bodies.dat"];
    TabδλpT0nBrown = Import["Interpolations/TabdλmbdapT0nBrown.dat"];

    TabδλnT0pBrehm = Import["Interpolations/TabdλmbdanT0pBrehm.dat"];
    TabδλpT0nBrehm = Import["Interpolations/TabdλmbdapT0nBrehm.dat"];
  ]
```

In[563]:=

```
If[$PaperPlots,
  si
    TFreeze = 0.8 MeV / kB;
    RCT = ListLogLinearPlot[{TabδλnT0p, TabδλpT0n, TabδλnT0pBrown, TabδλpT0nBrown,
      TabδλnT0pBrehm, TabδλpT0nBrehm, TabδλnT0pBrown5Bodies}, Frame → True,
      FrameStyle → Thickness[0.004], Joined → True, FrameLabel → {"T (K)", "δT/T"},
      LabelStyle → {FontSize → 12}, GridLines → {{{TFreeze, {Gray, Thickness[
        0.005]}}, {}}, PlotStyle → {{Darker@Darker@Green, Thickness[0.005]},
        {Darker@Darker@Green, Thickness[0.005], Dashing[{0.02]}},
        {Red, Thickness[0.006]}, {Red, Thickness[0.006], Dashing[{0.02]}},
        {Blue, Thickness[0.004]}, {Blue, Thickness[0.004], Dashing[{0.02]}},
        {Red, Thickness[0.002], Thickness[0.003]}}, FrameTicks → MyFrameTicksLog]
    If[$PaperPlots, Export["Plots/LogPlotdeltaGammaCCRT.pdf",
      Style[RCT, Magnification → 1], "PDF"];
```

Nuclear reactions network

Nuclear Species

Names and (N,Z)

There is the list of short names with their (neutron, proton) weights. So by definition a neutron has (1, 0) and a proton has (0, 1) and He4 has (2, 2) and so on.

In[565]:=

```
NamesWithWeightsAll = {{ "n", {1, 0}}, {"p", {0, 1}}, {"d", {1, 1}}, {"t", {2, 1}},
  {"He3", {1, 2}}, {"a", {2, 2}}, {"He5", {3, 2}}, {"He6", {4, 2}},
  {"Li6", {3, 3}}, {"Li7", {4, 3}}, {"Li8", {5, 3}}, {"Li9", {6, 3}},
  {"Be7", {3, 4}}, {"Be8", {4, 4}}, {"Be9", {5, 4}},
  {"Be10", {6, 4}}, {"Be11", {7, 4}}, {"Be12", {8, 4}},
  {"B8", {3, 5}}, {"B9", {4, 5}}, {"B10", {5, 5}}, {"B11", {6, 5}},
  {"B12", {7, 5}}, {"B13", {8, 5}}, {"B14", {9, 5}}, {"B15", {10, 5}},
  {"C9", {3, 6}}, {"C10", {4, 6}}, {"C11", {5, 6}}, {"C12", {6, 6}},
  {"C13", {7, 6}}, {"C14", {8, 6}}, {"C15", {9, 6}}, {"C16", {10, 6}},
  {"N12", {5, 7}}, {"N13", {6, 7}}, {"N14", {7, 7}},
  {"N15", {8, 7}}, {"N16", {9, 7}}, {"N17", {10, 7}},
  {"O13", {5, 8}}, {"O14", {6, 8}}, {"O15", {7, 8}}, {"O16", {8, 8}},
  {"O17", {9, 8}}, {"O18", {10, 8}}, {"O19", {11, 8}}, {"O20", {12, 8}},
  {"F17", {8, 9}}, {"F18", {9, 9}}, {"F19", {10, 9}}, {"F20", {11, 9}},
  {"Ne18", {8, 10}}, {"Ne19", {9, 10}}, {"Ne20", {10, 10}},
  {"Ne21", {11, 10}}, {"Ne22", {12, 10}}, {"Ne23", {13, 10}},
  {"Na20", {9, 11}}, {"Na21", {10, 11}}, {"Na22", {11, 11}}, {"Na23", {12, 11}}};
```

Let us visualize the nuclides used in as a table in (Z,N).

In[566]:=

```
TableNZNuclons = Table[" ", {i, 0, 13}, {j, 0, 11}];





```

```
In[568]:= Grid[Transpose@TableNZNucleons, Frame → All]
      grille  transpose          cadre  tout
(* This is table III in companion paper*)
```

```
Out[568]=
```

	n												
p	d	t											
	He3	a	He5	He6									
			Li6	Li7	Li8	Li9							
			Be7	Be8	Be9	Be10	Be11	Be12					
			B8	B9	B10	B11	B12	B13	B14	B15			
			C9	C10	C11	C12	C13	C14	C15	C16			
					N12	N13	N14	N15	N16	N17			
					O13	O14	O15	O16	O17	O18	O19	O20	
								F17	F18	F19	F20		
								Ne18	Ne19	Ne20	Ne21	Ne22	Ne23
								Na20	Na21	Na22	Na23		

The list of the species names only

```
In[569]:= ShortNamesAll = NamesWithWeightsAll[All, 1];
      tout
```

The list of {n, p} pairs only.

```
In[570]:= ListNPPairs = NamesWithWeightsAll[All, 2];
      tout
```

Functions to check if a name exists

```
In[571]:= ExistName[name_] := MemberQ[ShortNamesAll, name];
      appartient ?
ExistPair[pair_List] := MemberQ[ListNPPairs, pair];
      liste      appartient ?
```

This function selects the names of the species which all have the same mass number A

```
In[573]:= NamesMassNumberAll[A_] :=
  Select[NamesWithWeightsAll, ((Plus@@ (#[2])) == A) &] [All, 1]
      sélectionne      plus      tout
```

Dictionaries between names and numbers

We define dictionaries to handle species. We associate a number to each species

It is a simple correspondance between names and position of the species in the list, or between the names and the pair (neutron,proton).

In[574]:=

```
KeySpecies = Association@ (Rule@@@NamesWithWeightsAll)
               [association]      [règle]
```

Out[574]=

```
<| n → {1, 0}, p → {0, 1}, d → {1, 1}, t → {2, 1}, He3 → {1, 2}, a → {2, 2}, He5 → {3, 2},
   He6 → {4, 2}, Li6 → {3, 3}, Li7 → {4, 3}, Li8 → {5, 3}, Li9 → {6, 3}, Be7 → {3, 4},
   Be8 → {4, 4}, Be9 → {5, 4}, Be10 → {6, 4}, Be11 → {7, 4}, Be12 → {8, 4},
   B8 → {3, 5}, B9 → {4, 5}, B10 → {5, 5}, B11 → {6, 5}, B12 → {7, 5}, B13 → {8, 5},
   B14 → {9, 5}, B15 → {10, 5}, C9 → {3, 6}, C10 → {4, 6}, C11 → {5, 6}, C12 → {6, 6},
   C13 → {7, 6}, C14 → {8, 6}, C15 → {9, 6}, C16 → {10, 6}, N12 → {5, 7}, N13 → {6, 7},
   N14 → {7, 7}, N15 → {8, 7}, N16 → {9, 7}, N17 → {10, 7}, O13 → {5, 8}, O14 → {6, 8},
   O15 → {7, 8}, O16 → {8, 8}, O17 → {9, 8}, O18 → {10, 8}, O19 → {11, 8},
   O20 → {12, 8}, F17 → {8, 9}, F18 → {9, 9}, F19 → {10, 9}, F20 → {11, 9},
   Ne18 → {8, 10}, Ne19 → {9, 10}, Ne20 → {10, 10}, Ne21 → {11, 10}, Ne22 → {12, 10},
   Ne23 → {13, 10}, Na20 → {9, 11}, Na21 → {10, 11}, Na22 → {11, 11}, Na23 → {12, 11} |>
```

We have a reverse dictionary if we want

In[575]:=

```
KeyNucleons = Association@ (Rule@@@ (Reverse /@NamesWithWeightsAll))
               [association]      [règle]      [inverse]
```

Out[575]=

```
<| {1, 0} → n, {0, 1} → p, {1, 1} → d, {2, 1} → t, {1, 2} → He3, {2, 2} → a, {3, 2} → He5,
   {4, 2} → He6, {3, 3} → Li6, {4, 3} → Li7, {5, 3} → Li8, {6, 3} → Li9, {3, 4} → Be7,
   {4, 4} → Be8, {5, 4} → Be9, {6, 4} → Be10, {7, 4} → Be11, {8, 4} → Be12,
   {3, 5} → B8, {4, 5} → B9, {5, 5} → B10, {6, 5} → B11, {7, 5} → B12, {8, 5} → B13,
   {9, 5} → B14, {10, 5} → B15, {3, 6} → C9, {4, 6} → C10, {5, 6} → C11, {6, 6} → C12,
   {7, 6} → C13, {8, 6} → C14, {9, 6} → C15, {10, 6} → C16, {5, 7} → N12, {6, 7} → N13,
   {7, 7} → N14, {8, 7} → N15, {9, 7} → N16, {10, 7} → N17, {5, 8} → O13, {6, 8} → O14,
   {7, 8} → O15, {8, 8} → O16, {9, 8} → O17, {10, 8} → O18, {11, 8} → O19,
   {12, 8} → O20, {8, 9} → F17, {9, 9} → F18, {10, 9} → F19, {11, 9} → F20,
   {8, 10} → Ne18, {9, 10} → Ne19, {10, 10} → Ne20, {11, 10} → Ne21, {12, 10} → Ne22,
   {13, 10} → Ne23, {9, 11} → Na20, {10, 11} → Na21, {11, 11} → Na22, {12, 11} → Na23 |>
```

N, Z, A from name

In[576]:=

```
Ni["Bm"] := 1;
Ni["Bp"] := -1;
Ni["g"] := 0;

Zi["Bm"] := -1;
Zi["Bp"] := 1;
Zi["g"] := 0;

Ai["Bm"] := 0;
Ai["Bp"] := 0;
Ai["g"] := 0;
```

In[585]:=

```

Ni[key_] := KeySpecies[key][[1]]
Zi[key_] := KeySpecies[key][[2]]
Ai[key_] := Zi[key] + Ni[key]

```

In[588]:=

```

Ai["Li7"]
Ni["Li7"]
Zi["Li7"]
Ni["Bm"]

```

Out[588]=

7

Out[589]=

4

Out[590]=

3

Out[591]=

1

Binding energies and spins of nuclei

Tools to reshape the file "nubase2016.asc"

In[592]:=

```

SpinFromCharList[charlist_List] := StringReplace[StringJoin@@charlist,
  {"(" → "", ")" → "", "," → "", "+" → "", "-" → "", " " → "", "#" → ""}]
MassFromCharList[charlist_List] :=
  StringReplace[StringJoin@@charlist, {" " → "", "#" → ""}]

```

We load the file "nubtab03.asc"

In[594]:=

```

StringListParticles = #[[1]] & /@ Import["nubase2016.asc", "TSV"];
NubTabChar = Select[Characters /@ StringListParticles, Length[#] ≥ 93 &];

```

NUBASE Syntax :

The first three characters of each line are the A.

Characters from 5 to 8 are $10 \cdot Z$

From 20 to 29 it is mass excess.

From 80 to 93 it is some information on spin and parity.

In[596]:=

```

Alist = ToExpression /@ StringJoin /@ (Take[#, {1, 3}] & /@ NubTabChar);
      |en expression      |joins chaînes de c... |prends
Zlist = # / 10 & /@ ToExpression /@ StringJoin /@ (Take[#, {5, 8}] & /@ NubTabChar);
      |en expression      |joins chaînes de c... |prends
MassExcessesString = MassFromCharList /@ (Take[#, {20, 29}] & /@ NubTabChar);
                        |prends
Spins = SpinFromCharList /@ (Take[#, {80, 93}] & /@ NubTabChar);
                        |prends
Nlist = Alist - Zlist;

```

We gather all this information in a table

In[601]:=

```

MyGrid[ListNPBindingSpinName =
  Flatten[{KeyNucleons[{{#[[1]], #[[2]]}], #}] & /@ Select[Transpose[
    |aplatis      |sélection... |transpose
    {Nlist, Zlist, MassExcessesString, Spins}], ExistPair[{{#[[1]], #[[2]]}] &]]

```

Out[601]=

n	1	0	8071.3171	1/2
p	0	1	7288.9706	1/2
d	1	1	13135.7217	1
t	2	1	14949.8099	1/2
He3	1	2	14931.2179	1/2
a	2	2	2424.9156	0
He5	3	2	11231	3/2
He6	4	2	17592.10	0
Li6	3	3	14086.8789	1
Li7	4	3	14907.105	3/2
Be7	3	4	15769.00	3/2
Li8	5	3	20945.80	2
Be8	4	4	4941.67	0
B8	3	5	22921.6	2
Li9	6	3	24954.90	3/2
Be9	5	4	11348.45	3/2
B9	4	5	12416.5	3/2
C9	3	6	28911.0	3/2
Be10	6	4	12607.49	0
B10	5	5	12050.609	3
C10	4	6	15698.67	0
Be11	7	4	20177.17	1/2
B11	6	5	8667.707	3/2
C11	5	6	10649.40	3/2
Be12	8	4	25077.8	0
B12	7	5	13369.4	1
C12	6	6	0.0	0
N12	5	7	17338.1	1
B13	8	5	16561.9	3/2
C13	7	6	3125.0088	1/2
N13	6	7	5345.48	1/2
O13	5	8	23115	3/2

B14	9	5	23664	2
C14	8	6	3019.893	0
N14	7	7	2863.4167	1
O14	6	8	8007.781	0
B15	10	5	28958	3/2
C15	9	6	9873.1	1/2
N15	8	7	101.4387	1/2
O15	7	8	2855.6	1/2
C16	10	6	13694	0
N16	9	7	5683.9	2
O16	8	8	-4737.0013	0
N17	10	7	7870	1/2
O17	9	8	-808.7635	5/2
F17	8	9	1951.70	5/2
O18	10	8	-782.8156	0
F18	9	9	873.1	1
Ne18	8	10	5317.6	0
O19	11	8	3332.9	5/2
F19	10	9	-1487.4442	1/2
Ne19	9	10	1752.05	1/2
O20	12	8	3796.2	0
F20	11	9	-17.463	2
Ne20	10	10	-7041.9305	0
Na20	9	11	6850.6	2
Ne21	11	10	-5731.78	3/2
Na21	10	11	-2184.63	3/2
Ne22	12	10	-8024.719	0
Na22	11	11	-5181.51	3
Ne23	13	10	-5154.05	5/2
Na23	12	11	-9529.8525	3/2

We define a dictionary for excess masses (in keV)

In[602]:=

```
ExcessMassKeys =
  Association[ {#[[1]] → ToExpression[#[[4]]]} & /@ ListNPBindingSpinName]
  [association] [en expression]
```

Out[602]=

```
<| n → 8071.3171, p → 7288.9706, d → 13 135.722, t → 14 949.81, He3 → 14 931.218,
  a → 2424.9156, He5 → 11 231, He6 → 17 592.1, Li6 → 14 086.879, Li7 → 14 907.105,
  Be7 → 15 769., Li8 → 20 945.8, Be8 → 4941.67, B8 → 22 921.6, Li9 → 24 954.9,
  Be9 → 11 348.45, B9 → 12 416.5, C9 → 28 911., Be10 → 12 607.49, B10 → 12 050.609,
  C10 → 15 698.67, Be11 → 20 177.17, B11 → 8667.707, C11 → 10 649.4,
  Be12 → 25 077.8, B12 → 13 369.4, C12 → 0., N12 → 17 338.1, B13 → 16 561.9,
  C13 → 3125.0088, N13 → 5345.48, O13 → 23 115, B14 → 23 664, C14 → 3019.893,
  N14 → 2863.4167, O14 → 8007.781, B15 → 28 958, C15 → 9873.1, N15 → 101.4387,
  O15 → 2855.6, C16 → 13 694, N16 → 5683.9, O16 → -4737.0013, N17 → 7870,
  O17 → -808.7635, F17 → 1951.7, O18 → -782.8156, F18 → 873.1, Ne18 → 5317.6,
  O19 → 3332.9, F19 → -1487.4442, Ne19 → 1752.05, O20 → 3796.2, F20 → -17.463,
  Ne20 → -7041.9305, Na20 → 6850.6, Ne21 → -5731.78, Na21 → -2184.63,
  Ne22 → -8024.719, Na22 → -5181.51, Ne23 → -5154.05, Na23 → -9529.8525 |>
```

And a dictionary for spins

In[603]:=

```
SpinKeys = Association[ {#[[1]] → ToExpression[#[[5]]]} & /@ ListNPBindingSpinName]
  [association] [en expression]
```

Out[603]=

```
<| n →  $\frac{1}{2}$ , p →  $\frac{1}{2}$ , d → 1, t →  $\frac{1}{2}$ , He3 →  $\frac{1}{2}$ , a → 0, He5 →  $\frac{3}{2}$ , He6 → 0, Li6 → 1, Li7 →  $\frac{3}{2}$ ,
  Be7 →  $\frac{3}{2}$ , Li8 → 2, Be8 → 0, B8 → 2, Li9 →  $\frac{3}{2}$ , Be9 →  $\frac{3}{2}$ , B9 →  $\frac{3}{2}$ , C9 →  $\frac{3}{2}$ , Be10 → 0,
  B10 → 3, C10 → 0, Be11 →  $\frac{1}{2}$ , B11 →  $\frac{3}{2}$ , C11 →  $\frac{3}{2}$ , Be12 → 0, B12 → 1, C12 → 0, N12 → 1,
  B13 →  $\frac{3}{2}$ , C13 →  $\frac{1}{2}$ , N13 →  $\frac{1}{2}$ , O13 →  $\frac{3}{2}$ , B14 → 2, C14 → 0, N14 → 1, O14 → 0, B15 →  $\frac{3}{2}$ ,
  C15 →  $\frac{1}{2}$ , N15 →  $\frac{1}{2}$ , O15 →  $\frac{1}{2}$ , C16 → 0, N16 → 2, O16 → 0, N17 →  $\frac{1}{2}$ , O17 →  $\frac{5}{2}$ , F17 →  $\frac{5}{2}$ ,
  O18 → 0, F18 → 1, Ne18 → 0, O19 →  $\frac{5}{2}$ , F19 →  $\frac{1}{2}$ , Ne19 →  $\frac{1}{2}$ , O20 → 0, F20 → 2,
  Ne20 → 0, Na20 → 2, Ne21 →  $\frac{3}{2}$ , Na21 →  $\frac{3}{2}$ , Ne22 → 0, Na22 → 3, Ne23 →  $\frac{5}{2}$ , Na23 →  $\frac{3}{2}$  |>
```

From excess masses we can find binding energies (in keV). WE only need the excess mass of proton and neutron and the (Z,A,N) of the nuclide.

In[604]:=

```
Eneutron := ExcessMassKeys["n"];
Eproton := ExcessMassKeys["p"];
```

In[606]:=

```

BindingEnergy[name_] := Module[{Pair, A, Z, N},
  Pair = KeySpecies[name];
  Z = Pair[[2]];
  N = Pair[[1]];
  A = Z + N;
  N Eneutron + Z Eproton - ExcessMassKeys[name]]

Mass[name_] := Module[{Pair, A, Z, N},
  Pair = KeySpecies[name];
  Z = Pair[[2]];
  N = Pair[[1]];
  A = Z + N;
  A ma + keV ExcessMassKeys[name] - Z me]

```

We check a few binding energies (in keV)

In[608]:=

```

BindingEnergy["n"]
BindingEnergy["p"]
BindingEnergy["d"]
BindingEnergy["a"]

```

Out[608]=

0.

Out[609]=

0.

Out[610]=

2224.566

Out[611]=

28 295.66

In[612]:=

```

Mass["n"] / MeV
mn / MeV

```

Out[612]=

939.56538

Out[613]=

939.56542

```
In[614]:=
Mass["p"] / MeV
mp / MeV
```

```
Out[614]=
938.27203
```

```
Out[615]=
938.27209
```

```
In[616]:=
Mass["d"] / MeV
```

```
Out[616]=
1875.6128
```

Nuclear Statistical Equilibrium

This is Eq. A24 of companion paper.

```
In[617]:=
YNSE[name_, Yn_, Yp_, Tv_] := Module[{Pair, N, A, Z, mN, A320vermn},
  mN = (mn + mp) / 2;
  Pair = KeySpecies[name];
  Z = Pair[[2]];
  N = Pair[[1]];
  A = Z + N;
  A320vermn =  $\left( \frac{\text{Mass}[name]}{mn^{A-Z} * mp^Z} \right)^{3/2}$ ;
  (2 * SpinKeys[name] + 1) Zeta[3]^(A-1)  $\pi^{((1-A)/2)}$  2^((3 A-5)/2) A320vermn
  (kB Tv)^(3 (A-1)/2) ( $\eta\text{factorT}[Tv]$ )^(A-1) Yp^Z Yn^(A-Z) Exp[ $\frac{\text{BindingEnergy}[name] * \text{keV}}{kB Tv}$ ]
]
```

Reverse reaction information

The reverse reaction depends on three constants (α, β, γ) defined in companion paper in Eq. 142. From the Spin, mass and binding energy we can find these constants.

In[618]:=

```

Qreaction[ListIn_, ListOut_] := Module[
  {Ni = Length@ListIn, Nf = Length@ListOut, factorin, factorout, Units},
  factorin = Plus@@ (BindingEnergy[#] & /@ ListIn);
  factorout = Plus@@ (BindingEnergy[#] & /@ ListOut);
  Units = keV;
  -Units (factorin - factorout)
];

```

In[619]:=

```

PowerT9[ListIn_, ListOut_] := Module[{Ni = Length@ListIn, Nf = Length@ListOut},
  3 / 2. * (Ni - Nf)
];

```

In[620]:=

```

FactorInverseReaction[ListIn_, ListOut_] := Module[
  {Ni = Length@ListIn, Nf = Length@ListOut, factorin, factorout, Units},
  factorin =
    Times@@ (((2 SpinKeys[#[[1]]] + 1) (2 Pi / Mass[#[[1]]] / (kB 10^9)) ^ (-3 / 2)) ^
      (#[[2]] / (#[[2]]!)) & /@ (Tally@ListIn));
  factorout =
    Times@@ (((2 SpinKeys[#[[1]]] + 1) (2 Pi / Mass[#[[1]]] / (kB 10^9)) ^ (-3 / 2)) ^
      (#[[2]] / (#[[2]]!)) & /@ (Tally@ListOut));
  Units = (ma / c light^2) / (hbar c light) ^3 ^ (Ni - Nf);
  factorin / factorout Units
];

```

In[621]:=

```

GatherInfoReac[ListIn_, ListOut_] :=
  {Qreaction[ListIn, ListOut] / MeV, FactorInverseReaction[ListIn, ListOut],
   PowerT9[ListIn, ListOut], -Qreaction[ListIn, ListOut] / kB / 10^9};

RemoveNonNuclear[Species_List] :=
  Select[Species, # != "g" && # != "Bm" && # != "Bp" &];
InfoReaction[{ListIn_, ListOut_}] :=
  GatherInfoReac[RemoveNonNuclear@ListIn, RemoveNonNuclear@ListOut];
InfoReaction[ListIn_, ListOut_] := InfoReaction[{ListIn, ListOut}]

```

For a given reaction, defined by the list of initial particles and final particles, we get these constants with the function InfoReaction.

For example

```
In[625]:= InfoReaction[{"n", "p"}, {"d", "g"}]
Out[625]= {2.224566, 4.7161402 × 109, 1.5, -25.81502}
```

Check reaction coherence (formal conservation of N and Z)

```
In[626]:= CheckReaction[{ListIn_, ListOut_}] := Module[{Znet, Nnet, Anet},
  Znet = -Plus@@(Zi /@ ListIn) + Plus@@(Zi /@ ListOut);
  Nnet = -Plus@@(Ni /@ ListIn) + Plus@@(Ni /@ ListOut);
  Anet = -Plus@@(Ai /@ ListIn) + Plus@@(Ai /@ ListOut);
  (*Print[ListIn, " ", ListOut, " ", Znet, Nnet, Anet];*)
  If[Znet != 0 || Nnet != 0 || Anet != 0,
    Print["ERROR! This reaction ", ListIn, " -> ", ListOut,
      " is not possible.\nThe net result for Z, N and A are ",
      Znet, " ", Nnet, " ", Anet];
    MessageDialog["A reaction is not possible. Kernel has been aborted."];
    Print["We abort the evaluation !"];
    Beep[];
    Quit[];
    (*TODO Maybe a better
      handling of errors than just a violent Quit[...]... *)
  ];
]
```

```
CheckReaction[ListIn_, ListOut_] := CheckReaction[{ListIn, ListOut}]
```

```
In[628]:= (*CheckReaction[{"n"}, {"p", "Bm"}];*)
```

Nuclear Reaction rates

Radiative Corrections from pair productions

For photon producing processes we need to rescale the rates because of pair producing reactions. This has been computed in [Pitrou&Pospelov 2020]

In[629]:=

```
RescaleElectricDipoleKroll[Ea_] =
  1 -  $\frac{10 \alpha_{\text{FS}}}{9 \pi} + \frac{3 \alpha_{\text{FS}}}{4 E a^4 \pi} + \frac{3 \alpha_{\text{FS}} \text{Log}[16]}{4 E a^4 \pi} + \frac{2 \alpha_{\text{FS}} \text{Log}[64]}{9 \pi} - \frac{\alpha_{\text{FS}} \text{Log}\left[\frac{4}{E a^2}\right]}{3 \pi} - \frac{3 \alpha_{\text{FS}} \text{Log}\left[\frac{4}{E a^2}\right]}{4 E a^4 \pi};$ 
  (* Ea is the available energy in units of electron mass.*)

EML[{Z1_, A1_}, {Z2_, A2_}][T9_] =
  0.1220 * Z1^(2/3) * Z2^(2/3) * ((A1 * A2) / (A1 + A2))^(1/3) T9^(2/3) MeV;
  (* This is the most likely Kinetic energy in MeV*)

Clear[RescaleElectricDipoleZAZAQ]
RescaleElectricDipoleZAZAQ[{Z1_, A1_}, {Z2_, A2_}, Qv_][T9_] =
  (RescaleElectricDipoleKroll[EML[{Z1, A1}, {Z2, A2}][T9] + Qv] / me);

(*We use a rescaling computed at the most likely kinetic energy,
On top of which we add the mass gap to the total available energy *)
RescaleElectricDipoleZAZAQ[{Zi[Nucl1], Ai[Nucl1]}, {Zi[Nucl2],
  Ai[Nucl2]}, (Mass[Nucl1] + Mass[Nucl2] - Mass[Nuclfinal])][T9]]
```

The some rescaling factors due to possible pair production for reactions which produce a photon in the final state. These are functions of T9 (=Temperature in GigaK).

```

In[635]:=
RescaleElectricDipole["d", "p", "He3"] [0.8]
RescaleElectricDipole["t", "a", "Li7"] [0.8]
RescaleElectricDipole["He3", "a", "Be7"] [0.8]
RescaleElectricDipole["t", "p", "a"] [0.8]

Out[635]=
1.0021967

Out[636]=
1.0010648

Out[637]=
1.0005748

Out[638]=
1.0041641

In[639]:=
RescaleElectricDipole["d", "p", "He3"] [0.008]
RescaleElectricDipole["t", "a", "Li7"] [0.008]
RescaleElectricDipole["He3", "a", "Be7"] [0.008]
RescaleElectricDipole["t", "p", "a"] [0.008]

Out[639]=
1.0021723

Out[640]=
1.0009542

Out[641]=
1.000346

Out[642]=
1.004157

```

Random Number Generation for nuclear rates uncertainties

Generator of random number according to Normal distribution. But we make sure to use always the same sequence to avoid noise in Monte-Carlo.

This is crucial because this reduces Monte-Carlo noise when evaluating uncertainty in rates.

So for a given seed we precompute a list of 1000 random numbers.

Then we call several times `MyNormalRandom[seed]` which gives successively the random numbers which were generated with the seed.

```

In[643]:=
Clear[TableRandom, MyNormalRandom]
efface
$NRandomPoints = 1000;
(* We put something larger than the max number of reactions *)
TableRandom[seed_] := TableRandom[seed] = (SeedRandom[seed];
amorçage aléatoire
Table[RandomVariate[NormalDistribution[]], {i, 1, $NRandomPoints}])
table variable aléatoire loi normale

```


In[646]:=

```
InitializeRandom[seed_] := (IndexRandom[seed] = 1);
RandomFromTable[seed_] := With[{r = TableRandom[seed][[IndexRandom[seed]]}],
  IndexRandom[seed] = IndexRandom[seed] + 1;
  r]
MyNormalRandom[seed_] := RandomFromTable[seed]
```

In[649]:=

```
$Seed := 0;
Initialize[$Seed];
NormalRealisation := If[$RandomNuclearRates, MyNormalRandom[$Seed], 0];
```

Importation of reactions from external files (336 reactions)

We collect tools to read the reactions from the external file. This is low level code... because we need to deal with syntax.

This function constructs the reverse reaction. Its arguments are the name of the reverse reaction, the front factor, the power on T9 and the Q of the reaction.

In[652]:=

```
ReverseReaction[Name_, FrontFactor_, PoweronT9_, Qoverkb_] :=
  With[{Reversname = ToExpression["Hold@Lbar" <> Name],
    name = Evaluate[Symbol["L" <> Name]]},
    If[FrontFactor > 0,
      MySet[Reversname, Function[{Tvr}, With[{T9 = Tvr / Giga},
        FrontFactor (T9)PoweronT9 * Exp[ $\frac{Qoverkb}{T9}$ ] * name[Tvr]]];
      MySet[Reversname, Function[{Tvr}, 0]];
    ]];
```

We need a tool which replaces "2a", by "a,a" in the description of the reaction. Hence a reaction with identical particles in the initial or final states X (e.g. something which gives two neutrons) can be written X + X or 2 X.

In[653]:=

```
ListSimplificationNotation = {"He4" → "a", "H3" → "t", "H2" → "d", "H1" → "p"};
```

In[654]:=

```

ReplaceMultipleElement[str_String] := Module[{first, rest, rule},
  [chaîne de caractères [module]
  If[StringLength[str] == 1, str,
    [si [longueur de chaîne de caractères]
    first = StringTake[str, 1] // ToExpression;
    [extrais chaîne de caractères [en expression]
    If[NumericQ[first],
      [si [expression numérique ?]
      rest = StringDrop[str, 1];
      [laisse tomber chaîne de caractères]
      rule = str -> Sequence @@ Table[rest, first];
      [séquence [table]
      str /. rule,
      str]
    ]
  ]
  ReplaceMultipleElementList[list_List] := ReplaceMultipleElement /@ list;
  [liste]

```

In[656]:=

```
ReplaceMultipleElementList[{"He4", "2H1"}]
```

Out[656]=

```
{He4, H1, H1}
```

For a line (the list of elements of this line more precisely) describing a reaction we build the rates and inverse rates, and we output a formal description of the reaction in terms of initial and final particles

In[657]:=

```

TreatData[Data_] := Module[{reac, constants, ReferencePaper,
  module
  dat, rest, list, resultat, reacreshaped, replacements},
  resultat = {};
  list = Data;

  While[Length@list > 0,
    penda longueur
    reac = list[[1]];
    ReferencePaper = StringDrop[list[[2, 1], 2];
      laisse tomber chaîne de caractères
    (*Print[ReferencePaper];*)
      imprime
    (*Print[reac];*)
      imprime
    constants = list[[3]];
    rest = Drop[list, 3];
      laisse tomber
    dat = {};
    While[rest != {} && NumericQ[rest[[1, 1]]],
      pendant que expression numérique ?
      dat = Append[dat, rest[[1]]];
        ajoute en fin
      rest = Rest@rest;
        reste
    ];
    list = rest;
    reacreshaped = Append[
      ajoute en fin
      {ReplaceMultipleElementList@Select[reac, {# != "+" && # != "*-"} &],
        sélectionne
        constants, ReferencePaper}, dat];
    resultat = Append[resultat, reacreshaped /. ListSimplificationNotation];
      ajoute en fin
  ];
  resultat
];

```

In[658]:=

```

TruncateRateVariation[rate_] := Min[$MaxVariationRate, rate]
  minimum

```

In[659]:=

```

TreatReactionLine[line_] :=
  Module[{rescalefactor, reac, constants, interpfunction, data, len,
    module
    table, Tmin, rmin, wedgeposition, colonposition, InitialParticles,
    FinalParticles, BooleanFileData, Q, FrontFactor, PoweronT9,
    Qoverkb, Name, Lname, rv, ReferencePaper, InfoFromAudi2017},
    reac = line[[1]];

```

```

(*Print["Treating reaction : ",reac];*)
  |imprime
constants = line[[2]];
ReferencePaper = line[[3]];
data = line[[4]];

len = Length@line;
  |longueur
wedgeposition = Position[reac, ">"][[1, 1]];
  |position
colonposition = Position[reac, ";"][[1, 1]];
  |position
InitialParticles = Take[reac, {1, wedgeposition - 1}];
  |prends
FinalParticles = Take[reac, {wedgeposition + 1, colonposition - 1}];
  |prends

(* We quit if the reaction does not conserve formally Z or N,
  |valeur numérique
that is if it cannot exist *)
CheckReaction[InitialParticles, FinalParticles];

Name = StringJoin@@ToString/@InitialParticles<>
  |joins chaînes de... |en chaîne de caractères
  "T0"<>StringJoin@@ToString/@FinalParticles;
  |joins chaînes de... |en chaîne de caractères

Q = constants[[1]];
FrontFactor = constants[[2]];
PoweronT9 = constants[[3]];
Qoverkb = constants[[4]];

(* We check the constants used in reverse rates *)
InfoFromAudi2017 = InfoReaction[InitialParticles, FinalParticles];
(*Print[InitialParticles," ",FinalParticles," ",InfoFromAudi2017];*)
  |imprime

If[Abs[FrontFactor / InfoFromAudi2017[[2]] - 1] > 0.001,
  |si |valeur absolue
  Print[Name, " WARNING. We use  $\alpha$ =", FrontFactor,
  |imprime
  " but we should use ", InfoFromAudi2017[[2]]
  ];

If[Abs[Qoverkb / InfoFromAudi2017[[4]] - 1] > 0.001,
  |si |valeur absolue
  Print[Name, " WARNING. We use  $Q/k_B$ =",
  |imprime
  Qoverkb, " but we should use ", InfoFromAudi2017[[4]]

```

```

];

If[PoweronT9 != InfoFromAudi2017[[3]],
  _si
  Print[Name, " WARNING. We use power on T9 =",
    _imprime
    PoweronT9, " but we should use ", InfoFromAudi2017[[3]]
  ];
(* ***** *)

(* *** *)
(*For exploration of parameters we can redefine some front
  _boucle for
  factors to rescales reactions. For instance the DPG reaction*)
  _boucle for
(* Added on request of Antony Lewis *)

rescalefactor = 1;
(*Print[Name,FullForm[Name]];*)
  _imprime      _forme pleine

If[$RescaleSomeRates,
  _si
  If[NumericQ[Symbol[Name <> "Factor"]],
    _si _expression... _symbole      _factorise
    rescalefactor = Symbol[Name <> "Factor"];
    _symbole      _factorise
    Print[Name, " reaction is rescaled by ", rescalefactor];
    _imprime
  ]
];

table = Map[{Giga#[[1]], #[[2]] Hz, #[[3]]} &, data];
  _applique à travers
Tmin = Last[table][[1]];
  _dernier
rmin = Last[table][[2]];
  _dernier
Lname = ToExpression["Hold@L" <> Name];
  _en expression      _maintiens
rv = NormalRealisation;

(* We also rescale some nuclear according to QED corrections
  (pair productions in the final state if it produces a  $\gamma$  *)
(* This is based on [Pitrou&Posepelov 2020] *)
RescaleFunction[T9_] := 1;
(* Possible RescaleFunction for some selected rates, in function of T9*)

```

```

(* This is the np→dg correction from pair particle production,
found from our electric/dipole modelization of base rate,
etc... using Landau-Lifshitz.
See [Pitrou&Pospelov 2020]*)

If[$NuclearRatesQEDCorrections,
si

If[Name === "npT0dg", RescaleFunction[T9_] :=
si
  Min[1.0009003934476768`, (1.0003328617393168` +
minimum
    0.00010013475534938917` T9 + 0.00004089993260910648` T92 -
    0.000011824673537229535` T93 + 1.0522377796855455` *-6 T94)]];

If[Name === "dpT0He3g",
si
  RescaleFunction[T9_] := RescaleElectricDipole["d", "p", "He3"] [T9];];

If[Name === "tpT0ag",
si
  RescaleFunction[T9_] := RescaleElectricDipole["t", "p", "a"] [T9];];

If[Name === "taT0Li7g",
si
  RescaleFunction[T9_] := RescaleElectricDipole["t", "a", "Li7"] [T9];];

If[Name === "He3aT0Be7g",
si
  RescaleFunction[T9_] := RescaleElectricDipole["He3", "a", "Be7"] [T9];];
];

MySet[Lname, MyInterpolationRate[
  {#[1], Identity[rescalefactor * RescaleFunction[#[1] / Giga] * #[2] *
identité
    If[$RandomNuclearRates, TruncateRateVariation[#[3] ^ rv, 1]]} & /@
si
    table]];
(* We do not rescale the reverse because
it is computed FROM the forward rate. So rescaling the
forward rate by rescalefactor rescales them both *)
ReverseReaction[Name, FrontFactor, PoweronT9, Qoverkb];

{Name, InitialParticles, FinalParticles, rv, ReferencePaper}

];

```

```
SetAttributes[TreatReactionLine, SequenceHold]
|alloue attributs |maintiens séquence
```

Lists of analytic reactions (86 reactions)

We have a list of 86 reactions for which we use analytic fits from the literature

In principle these reactions could be tabulated and incorporated into the external file but we prefer to keep their analytic forms.

- We need a few tools (this is painful low level code)

In[661]:=

```
ListTWagoner =
{0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.011,
 0.012, 0.013, 0.014, 0.015, 0.016, 0.018, 0.02, 0.025, 0.03, 0.04,
 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16,
 0.18, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.6, 0.7, 0.8, 0.9, 1.,
 1.25, 1.5, 1.75, 2., 2.5, 3., 3.5, 4., 5., 6., 7., 8., 9., 10.} * 10^9;
```

In[662]:=

```
$ListTWagoner = False;
|faux
```

In[663]:=

```
TableInterpolationTemperature =
If[$ListTWagoner, ListTWagoner, ListTRange[0.9 Tf, 10^10]];
|si
```

In[664]:=

```

SimplifyReactionStringRules =
  {"+" → " + ", ">" → " > ", ";" → " ; ", "2n" → " n + n ", "2p" → " p + p ",
   "2g" → " g + g ", "2d" → " d + d ", "2a" → " a + a ", "He4" → " a "};

ReshapeReactionString[string_String] :=
  [chaîne de caractères]
  Select[StringSplit[StringReplace[string, SimplifyReactionStringRules], " "],
    [sélectionne les chaînes de caractères]
    [fractionne chaînes de caractères]
    [remplace dans chaîne de caractères]
    (# != "+" && # != "*" && # != ";") &];

TreatReactionString[reac_String, source_String, f_] :=
  [chaîne de caractères] [chaîne de caractères]
  Module[{reacshaped, wedgeposition,
    [module]
    colonposition, InitialParticles, FinalParticles, Name},
    reacshaped = ReshapeReactionString[reac];
    wedgeposition = Position[reacshaped, ">"][[1, 1]];
    [position]
    colonposition = Position[reacshaped, ";"][[1, 1]];
    [position]
    InitialParticles = Take[reacshaped, {1, wedgeposition - 1}];
    [prends]
    FinalParticles = Take[reacshaped, {wedgeposition + 1, colonposition - 1}];
    [prends]

    (* We check that the reaction is possible,
    that is it should conserve N and Z*)
    [valeur numérique]

    (* If not the case,
    [si]
    the code will violently quit after spitting out warning messages.*/)
    CheckReaction[InitialParticles, FinalParticles];

    Name = StringJoin@@ToString/@InitialParticles<>
    [joins chaînes de caractères] [en chaîne de caractères]
    "T0" <> StringJoin@@ToString/@FinalParticles;
    [joins chaînes de caractères] [en chaîne de caractères]
    {Name, InitialParticles, FinalParticles, NormalRealisation, source, f}
  ]

```


In[667]:=

```

PostTreatT9[var_, funT9_] := If[$InterpolateAnalytics,
    _si
    MyInterpolationRate[Table[
        _table
        {i, var MyChop[funT9[i / 10^9]]}, {i, TableInterpolationTemperature}}],
    (var MyChop[funT9[# / 10^9]]) &];

GenRateT9[var_, funT9_] := PostTreatT9[var, funT9];

```

- This is the actual function where all analytic rates are defined. And it outputs the list of reactions.

In[669]:=

```

DefineAnalyticRates :=
Module[
_module
    {f, Var, Name, λReac, λbarReac, treatedreac, source, reac,
        analyticforward, AddReaction, initialparticles, finalparticles,
        InfoFromAudi2017, FrontFactor, Qoverkb, PoweronT9, forward, reslist},

    (* Most recent implementation
        _plupart
        with automatic computation of reverse rate *)
    AddReaction[reac_String, source_String, f_, ForwardT9_, BoolBackward_] := (
        _chaîne de caractères
        treatedreac = TreatReactionString[reac, source, f];
        Name = treatedreac[[1]];

        (* Building the backward ratio *)
        initialparticles = treatedreac[[2]];
        finalparticles = treatedreac[[3]];
        InfoFromAudi2017 = InfoReaction[initialparticles, finalparticles];
        (*Print[InitialParticles, " ", FinalParticles, " ", InfoFromAudi2017];*)
        _imprime
        FrontFactor = InfoFromAudi2017[[2]];
        Qoverkb = InfoFromAudi2017[[4]];
        PoweronT9 = InfoFromAudi2017[[3]];
        (* End of building backward ratio *)
        _termine

        λReac = ToExpression["Hold@L" <> Name];
        _en expression _maintiens
        λbarReac = ToExpression["Hold@Lbar" <> Name];
        _en expression _maintiens
        Sow[treatedreac];
        _sème
        Var = f^treatedreac[[4]];

        MySet[λReac, GenRateT9[Var, ForwardT9]];
        (*MySet[λbarReac, GenRateT9[Var, BackwardT9 ]];*)

```

```

If[BoolBackward,
  si
  MySet[λbarReac, GenRateT9[Var,
    (FrontFactor * #^PoweronT9 * Exp[Qoverkb / #] * ForwardT9[#]) & ]],
    exponentielle
  MySet[λbarReac, GenRateT9[0, 0 & ]]; (* No backward reaction *)
];

treatedreac);

reslist = Reap[
  récolte

  (* This is where all extra analytic reactions must be listed. *)
  (* For each reactions added analytically we need to specify
    boucle for
    a String source which is the paper in which it is found *)
    chaîne de caractères
  (* Then we give a string reac which is the reaction considered. *)
  (* The factor of uncertainty for Monte-Carlo is then given *)
  (* The analytic function forward[T9_],
  which is a function of T9 (that is the temperature in GK) *)
  (* With all these definitions we call AddReaction. The
    avec
    last argument is a boolean. If True it computes also
    si lrai
    the reverse rate from detailed balance arguments,
  and if False it does not do so. This is essentially for pure decay
    faux
    reactions that there is no need to compute the reverse rates. *)

  (*****
  *4He,3He,D,7Li (Extra reactions)
    dérivée d
  *****
  ******)
  source = "Nag06";
  reac = " d + n > t + g ; dng";
  f = 2.;
  forward[T9_] := With[{T923 = T9^ (2 / 3)}, (214. T90.075 + 7.42 T9)];
    avec
  AddReaction[reac, source, f, forward, True];
    lrai
  (* End of first reaction added analytically *)
    termine

  source = "Nag06";
  reac = "t+t>a+n+n;ttn";
  f = 3.;

```

```

forward[T9_] := With[avec {T923 = T9 ^ (2 / 3), T913 = T9 ^ (1 / 3),

    T943 = T9 ^ (4 / 3), T953 = T9 ^ (5 / 3)},  $\left( \frac{1}{T923} 1.67 \cdot 10^{-4.872/T913} \right.$ 

     $\left. (1. - 0.272 T9 + 0.086 T913 - 0.455 T923 + 0.148 T943 + 0.225 T953) \right)$ ];

AddReaction[reac, source, f, forward, True];vrai

source = "Wag69";
reac = "He3 +n > He4 + g ; hng";
f = 3.;
forward[T9_] := 6.62 * (1 + 905 * T9);
AddReaction[reac, source, f, forward, True];vrai

source = "CF88";
reac = "He3 + t > He4 + d ; htd";
f = 10.;
forward[T9_] := With[avec {T9A = T9 / (1. + 0.128 * T9), T932 = T9 ^ (3 / 2)},

    With[avec {T9A13 = T9A ^ (1. / 3.), T9A56 = T9A ^ (5. / 6.)},

         $5.46 \cdot 10^{-7.733 / T9A13}$ exponentielle

    ];
AddReaction[reac, source, f, forward, True];vrai

source = "CF88";
reac = "He3 + t > He4 + n + p ; htp";
f = 10.;
forward[T9_] := With[avec {T9A = T9 / (1. + 0.115 * T9), T932 = T9 ^ (3 / 2)},

    With[avec {T9A13 = T9A ^ (1. / 3.), T9A56 = T9A ^ (5. / 6.)},

         $7.71 \cdot 10^{-7.733 / T9A13}$ exponentielle

    ];
AddReaction[reac, source, f, forward, True];vrai

source = "NACRE";
reac = "a + a + n > Be9 + g ; aang";
f = 1.25;

```

```

forward[T9_] :=
  With[{T932 = T9 ^ (3 / 2), T923 = T9 ^ (2 / 3), T913 = T9 ^ (1 / 3)},
    Avec
      With[{he4abe8 = 2.43*^9 * (1. + 74.5 * T9) / T923 * Exp[
        -13.49 / T913 - (T9 / 0.15) ^ 2] + 6.09*^5 / T932 * Exp[-1.054 / T9]},
          Exponentielle
            If[T9 < 0.03,
              Si
                (he4abe8) * 6.69*^-12 * (1. - 192 * T9 + 2.48*^4 * T9 ^ 2 -
                  1.50*^6 * T9 ^ 3 + 4.13*^7 * T9 ^ 4 - 3.90*^8 * T9 ^ 5),
                (he4abe8) * 2.42*^-12 * (1. - 1.52 * Log10[T9] +
                  0.448 * (Log10[T9]) ^ 2 + 0.435 * (Log10[T9]) ^ 3)]],
              Logarithme en base 10
                Logarithme en base 10
            ]
          ]
      ]
    AddReaction[reac, source, f, forward, True];
    Vrai

source = "CF88&MF89";
reac = "Li7 + t > a + a + n + n; li7ta";
f = 3.;
forward[T9_] := With[{T923 = T9 ^ (2 / 3), T913 = T9 ^ (1 / 3)},
  Avec
    8.81*^+11 / T923 * Exp[-11.333 / T913]];
    Exponentielle
  AddReaction[reac, source, f, forward, True];
  Vrai

source = "CF88&MF89";
reac = "Li7 + He3 > a + a + n + p; li7haa";
f = 3.;
forward[T9_] := With[{T923 = T9 ^ (2 / 3), T913 = T9 ^ (1 / 3)},
  Avec
    1.11*^+13 / T923 * Exp[-17.989 / T913]];
    Exponentielle
  AddReaction[reac, source, f, forward, True];
  Vrai

(* Idem problem T93 not divided in Coc *)

source = "Ba195";
reac = " Li8 + d > Li9 + p ; li8dp";
f = 3.;
forward[T9_] := With[{T923 = T9 ^ (2 / 3), T913 = T9 ^ (1 / 3)},
  Avec
    9.63*^6 / T923 * Exp[-10.324 / T913] * (1. + 0.404 * T913) * 74.];
    Exponentielle
  AddReaction[reac, source, f, forward, True];
  Vrai

```

```

source = "Has09c";
reac = " Li8 + d > Li7 + t ; li8dt";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T913 = T9^(1/3)},
  _avec
  (3.02*^8 / T9^0.624 * Exp[-3.51 / T9] +
    _exponentielle
    5.82*^11 / T923 * Exp[-19.72 / T913] * (1.0 + 0.280 * T913))];
AddReaction[reac, source, f, forward, True];
_vrai

(*Now given in tabulated file,
_maintenant
hence we comment out this old analytic form *)
(*source="CF88";
reac="Be7 + d > a + a + p ; be7dp";
f=3.;
forward[T9_] :=
  With[{T923=T9^(2/3), T913=T9^(1/3)}, 1.07*^12/T923*Exp[-12.428/T913]];
_avec _exponentielle
AddReaction[reac,source,f,forward,True];*)
_vrai

source = "CF88&MF89";
reac = "Be7 + t > a + a + n + p ; be7t";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T913 = T9^(1/3)},
  _avec
  2.91*^12 / T923 * Exp[-13.729 / T913]];
_exponentielle
AddReaction[reac, source, f, forward, True];
_vrai

source = "CF88&MF89";
reac = "Be7 + He3 > 2a + p + p ; be7h";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T913 = T9^(1/3)},
  _avec
  6.11*^13 / T923 * Exp[-21.793 / T913]];
_exponentielle
AddReaction[reac, source, f, forward, True];
_vrai

```

```

source = "Wie89";
reac = "C9 + a > N12 + p ; c9an";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T932 = T9^(3/2),
  avec
    T913 = T9^(1/3), T943 = T9^(4/3), T953 = T9^(5/3)},
  (1.668*^+15 / T923 * Exp[-31.272 / T913 - (T9 / .307)^2] *
    exponentielle
    (1. + 1.33*^-2 * T913 - 6.42 * T923 - .599 * T9 + 14.4 * T943 + 3.42 * T953) +
    56.8 / T932 * Exp[-5.292 / T9] + 1.7*^+5 / T932 * Exp[-14.08 / T9] +
    exponentielle
    6.52*^7 / T932 * Exp[-23.09 / T9])]);
AddReaction[reac, source, f, forward, True];
  vrai

(* =====
*6Li (Extra reactions)
*=====
=====*)

source = "CF88";
f = 3.;
(*(*This is an endothermic reaction. So we comment it and
  replace it with the exothermic reverse reaction below *)
reac="t+a>Li6+n;tan";
forward[T9_] :=
  With[{T9A=T9/(1.+49.18*T9)},With[{ T9A32=T9A^(3./2.),T932=T9^(3/2)},
    avec
    (1.80*^8*Exp[-55.494/T9]*(1.-.261*T9A32/T932)+
      exponentielle
      2.72*^9/T932*Exp[-57.884/T9])
      exponentielle
    ]];*)

(* Here is the same reaction but presented backward,
  ici
  such that it is exothermic in the forward direction *)
reac = "Li6+n>t+a;tan";
forward[T9_] := With[{T9A = T9 / (1. + 49.18 * T9)},
  avec
  With[{ T9A32 = T9A^(3. / 2.), T932 = T9^(3 / 2) },
    avec
    (1.80*^+8 * (1. - .261 * T9A32 / T932) * .935 +
      2.72*^9 / T932 * Exp[(55.494 - 57.884) / T9] * .935)
      exponentielle
  )];

```

```

]];
AddReaction[reac, source, f, forward, True];
    _vrai

source = "FK90";
reac = "He3 + t > Li6 + g ; htg";
f = 3.;
forward[T9_] :=
    With[{T92 = T9^2, T923 = T9^(2/3), T932 = T9^(3/2), T913 = T9^(1/3),
    _avec
        T943 = T9^(4/3), T953 = T9^(5/3)}, 2.21*^5 / T923 * Exp[-7.720 / T913] *
        _exponentielle
        (1. + 2.68 * T923 + 0.868 * T9 + 0.192 * T943 + 0.174 * T953 + 0.044 * T92) ];
AddReaction[reac, source, f, forward, True];
    _vrai

source = "CF88";
reac = "a + n + p > Li6 + g ; anpg";
f = 3.;
forward[T9_] :=
    If[T9 > 1, 4.62*^-6 / T9^2 * (1. + 0.075 * T9) * Exp[-19.353 / T9], 0];
    _si _exponentielle
AddReaction[reac, source, f, forward, True];
    _vrai

source = "MF89";
reac = "Li6 + n > Li7 + g ; li6ng";
f = 3.;
forward[T9_] := 5.10*^3;
AddReaction[reac, source, f, forward, True];
    _vrai

source = "MF89";
reac = "Li6 + d > Li7 + p ; li6dp";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T913 = T9^(1/3)},
    _avec
        1.48*^12 / T923 * Exp[-10.135 / T913]];
    _exponentielle
AddReaction[reac, source, f, forward, True];
    _vrai

source = "MF89";
reac = "Li6 + d > Be7 + n ; li6dn";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T913 = T9^(1/3)},
    _avec

```

```

1.48*^12 / T923 * Exp[-10.135 / T913]]];
      [exponentielle]
AddReaction[reac, source, f, forward, True];
      [vrai]

(*=====
  *Beryllium& Boron (Main reactions)
  *=====
    =====*)
source = "CF88";
reac = "Li6 + a > B10 + g ; li6ag";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T932 = T9^(3/2),
      [avec]
      T913 = T9^(1/3), T943 = T9^(4/3), T953 = T9^(5/3)},
      (4.06*^06 / T923 * Exp[-18.79 / T913 - (T9 / 1.326)^2] *
      [exponentielle]
      (1. + 0.022 * T913 + 1.54 * T923 + 0.239 * T9 + 2.2 * T943 + 0.869 * T953)
      + 1.91*^3 / T932 * Exp[-3.484 / T9] + 1.01*^4 / T9 * Exp[-7.269 / T9])];
      [exponentielle]
AddReaction[reac, source, f, forward, True];
      [vrai]

source = "NACRE";
reac = " Li7 + a > B10 + n ; li7an / b10na";
f = 1.08;
forward[T9_] :=
  1.325 * 1.66*^7 * (1. + 1.064 * T9) * 1 / 1.3242 * Exp[-32.3755 / T9];
      [exponentielle]
AddReaction[reac, source, f, forward, True];
      [vrai]

(* MF89 replace Wiescher et al.ApJ 464 (1989) 464.C Voir Blackmon
      [constante C]
      et al.PRC 54 (1996) 383& Heil et al.ApJ 507 (1998) 997.*)
(*MF89Hei98 *)
source = "MF89&Hei98";
reac = "Li7+n>Li8+g;li7ng";
f = 3.;
forward[T9_] :=
  With[{T932 = T9^(3/2)}, (6.015*^3 + 1.141*^4 / T932 * Exp[-2.576 / T9])];
      [avec]
      [exponentielle]
AddReaction[reac, source, f, forward, True];
      [vrai]

(*Should we replace by exothermic reaction ? *)
source = "MF89";

```



```

reac = "Li7 + d > Li8 + p ; li7dp ! Q<0 !";
f = 3.;
forward[T9_] :=
  With[{T932 = T9 ^ (3 / 2)}, 8.31*^8 / T932 * Exp[-6.998 / T9]];
  _avec _exponentielle
AddReaction[reac, source, f, forward, True];
  _vrai

source = "Rau94";
reac = "Li8 + n > Li9 + g ; li8ng";
f = 3.;
forward[T9_] :=
  With[{T932 = T9 ^ (3 / 2)}, (3.260*^3 + 6.328*^4 / T932 * Exp[-2.866 / T9])];
  _avec _exponentielle
AddReaction[reac, source, f, forward, True];
  _vrai

source = "Men12";
reac = "Li8 + p > a + a + n ; li8pn";
forward[T9_] := With[{T932 = T9 ^ (3 / 2), T913 = T9 ^ (1 / 3),
  _avec
    T923 = T9 ^ (2 / 3), T92 = T9 ^ 2, T93 = T9 ^ 3, T94 = T9 ^ 4, T95 = T9 ^ 5},
  If[T9 < 5, (
    _si
      5.36*^8 / T932 * Exp[-4.41 / T9] + 1.99*^8 / T932 * Exp[-7.08 / T9] +
      _exponentielle _exponentielle
      5.85*^10 / T923 * Exp[-8.50 / T913] * (1. - 1.70 * T9 +
      _exponentielle
      0.849 * T92 - 0.175 * T93 + 1.62*^-2 * T94 - 5.60*^-4 * T95)),
    7.777*^7]]];
AddReaction[reac, source, f, forward, True];
  _vrai

source = "Bal95";
reac = "Li8 + d > Be9 + n ; li8dn";
f = 3.;
forward[T9_] := With[{T913 = T9 ^ (1 / 3), T923 = T9 ^ (2 / 3)},
  _avec
    9.63*^6 / T923 * Exp[-10.324 / T913] * (1. + 0.404 * T913) * 188.];
    _exponentielle
AddReaction[reac, source, f, forward, True];
  _vrai

(*****
*Beryllium& Boron (Extra reactions)
*****
===== *)

```

```

source = "Rau94";
reac = "Be9 + n > Be10 + g ; be9ng";
f = 3.;
forward[T9_] :=
  With[{T913 = T9^(1/3), T923 = T9^(2/3), T932 = T9^(3/2)}, (1.01*^3 +
    avec
    1.01*^4 / T932 * Exp[-6.487 / T9] + 5.41*^4 / T932 * Exp[-8.471 / T9])];
    exponentielle exponentielle
AddReaction[reac, source, f, forward, True];
vrai

source = "NACRE";
reac = "Be9 + p > a + a + p + n ; be9pn";
f = 1.05;
forward[T9_] :=
  5.06*^7 * Exp[-21.479 / T9] * (1. + 1.26 * T9 - 0.0302 * T9^2);
    exponentielle
AddReaction[reac, source, f, forward, True];
vrai

source = "NACRE";
reac = "B11 + p > C11 + n ; b11pn ! Q < 0 !";
f = 1.1;
forward[T9_] := 1.36*^8 * Exp[-32.085 / T9] *
    exponentielle
    (1. + 0.963 * T9 - 0.285 * T9^2 + 3.36*^-2 * T9^3 - 1.37*^-3 * T9^4);
AddReaction[reac, source, f, forward, True];
vrai

source = "Rau94";
reac = " Be10 + n > Be11 + g ; be10ng";
f = 3.;
forward[T9_] :=
  With[{T932 = T9^(3/2)}, (5.96*^2 + 6.67*^5 / T932 * Exp[-14.85 / T9])];
    avec exponentielle
AddReaction[reac, source, f, forward, True];
vrai

source = "Rau94";
reac = "Be11 + n > Be12 + g ; be11ng";
f = 3.;
forward[T9_] := With[{T932 = T9^(3/2)}, 3.56*^2];
    avec
AddReaction[reac, source, f, forward, True];
vrai

```

```

source = "Des99Bea01";
reac = "B8 + p > C9 + g ; b8pg";
f = 3.;
forward[T9_] := With[{T932 = T9^(3/2), T913 = T9^(1/3), T92 = T9^2},
  _avec
  6.253*^5 * Exp[-11.971 / T913] * (1. - 7.03*^-2 * T9 + 6.25*^-3 * T92) ] ;
_exponentielle
AddReaction[reac, source, f, forward, True];
_vrai

(* =====
  *Leaks to CNO
  !*=====
  =====*)

source = "NACRE";
reac = "a + a + a > C12 + 2g ; aaag";
f = 1.15;
forward[T9_] :=
  With[{T932 = T9^(3/2), T923 = T9^(2/3), T913 = T9^(1/3)},
    _avec
    With[{he4abe8 = 2.43*^9 * (1. + 74.5 * T9) / T923 *
      _avec
      Exp[-13.49 / T913 - (T9 / 0.15)^2] + 6.09*^5 / T932 * Exp[-1.054 / T9],
        _exponentielle
        _exponentielle
      be8agc12 = 2.76*^7 *
        (1. + 5.47 * T9 + 326 * T9^2) / T923 * Exp[-23.570 / T913 - (T9 / 0.4)^2] +
          _exponentielle
          130.7 / T932 * Exp[-3.338 / T9] + 2.51*^4 / T932 * Exp[-20.307 / T9] },
        _exponentielle
        _exponentielle
      If[T9 < 0.03,
        _si
        he4abe8 * be8agc12 * 3.07*^-16 * (1. - 29.1 * T9 + 1308 * T9^2),
        he4abe8 * be8agc12 * 3.44*^-16 * (1. + 0.0158 / T9^0.65) ] ] ;
AddReaction[reac, source, f, forward, True];
_vrai

source = "Tang03";
reac = "C11+p>N12+g;c11pg";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T932 = T9^(3/2), T913 = T9^(1/3),
  _avec
  T943 = T9^(4/3), T953 = T9^(5/3)}, (1.670*^2 * Exp[-4.166 / T9] / T932 +
    _exponentielle
    2.148*^5 * Exp[-13.281 / T913] / T923 * (1. + 4.639 * T913 -
      _exponentielle
      2.641 * T923 - 1.543 * T9 + 2.030 * T943 + 4.657 * T953) ) ] ;
AddReaction[reac, source, f, forward, True];
_vrai

```

```

source = "CF88";
reac = "B10 + a > N13 + n ; b10an";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T913 = T9^(1/3)},
  _avec
  1.2*^13 / T923 * Exp[-27.989 / T913 - (T9 / 9.589)^2] ];
_exponentielle
AddReaction[reac, source, f, forward, True];
_vrai

source = "Wan91";
reac = "B11+a>C14+p;b11ap";
f = 3.;
forward[T9_] := With[{T923 = T9^(2/3), T932 = T9^(3/2),
  _avec
  T913 = T9^(1/3), T943 = T9^(4/3), T953 = T9^(5/3)},
  (8.403*^15 * Exp[-31.914 / T913 - (T9 / 0.3432)^2] *
_exponentielle
  (1. + 0.022 * T913 + 5.712 * T923 + 0.642 * T9 + 15.982 * T943 + 4.062 * T953)
  + 5.44*^-3 / T932 * Exp[-2.868 / T9] +
_exponentielle
  2.419*^2 / T932 * Exp[-5.147 / T9] + 4.899*^2 / T932 * Exp[-5.157 / T9] +
_exponentielle
  4.944*^6 / T9^(3/5) * Exp[-11.26 / T9]) );
_exponentielle
AddReaction[reac, source, f, forward, True];
_vrai

source = "Rau94";
reac = "C11+n>C12+g;c11ng";
f = 3.;
forward[T9_] := With[{T932 = T9^(3/2)}, (3.18*^4 +
_avec
  3.30*^3 / T932 * Exp[-0.917 / T9] + 1.05*^6 / T932 * Exp[-5.57 / T9]) );
_exponentielle
AddReaction[reac, source, f, forward, True];
_vrai

(*=====
====*)
(*      Decay Rates                                *)
(*=====
====*)
(* %Aud03 *)
(* All decay rates from %Aud03 *)
_tout

source = "Aud03";

```

```

reac = "He6>Li6+Bm";
forward[T9_] := Log[2] / 8.0670*^-1 ;
               logarithme
AddReaction[reac, source, 1, forward, False];
               \_faux

(* The 1 is because we do not put uncertainty on decays,
and the False because we do not put reverse reactions on decays *)
               \_faux

reac = "Li8>2a+Bm";
forward[T9_] := Log[2] / 8.4030*^-1 ;
               logarithme
AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "Li9>Be9+Bm";
forward[T9_] := Log[2] / 1.7830*^-1 * 0.492 ;
               logarithme
AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "Li9>a+a+n+Bm";
forward[T9_] := Log[2] / 1.7830*^-1 * 0.508 ;
               logarithme
AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "Be11>B11+Bm";
forward[T9_] := Log[2] / (1.3810*^1) ;
               logarithme
AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "Be12>B12+Bm";
forward[T9_] := Log[2] / (2.15*^-2) ;
               logarithme
AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "B8>a+a+Bp";
forward[T9_] := Log[2] / (7.70*^-1) ;
               logarithme
AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "B12>C12+Bm";
forward[T9_] := Log[2] / (2.02*^-2) ;
               logarithme
AddReaction[reac, source, 1, forward, False];
               \_faux

```

```

    reac = "B13>C13+Bm;";
    (* !04/11/2010 *)
    forward[T9_] := Log[2] / (1.733*^-2) ;
    logarithme
    AddReaction[reac, source, 1, forward, False];
    \_faux

    reac = "B14>C14+Bm;";
    (* !04/11/2010 *)
    forward[T9_] := Log[2] / (1.25*^-2) ;
    logarithme
    AddReaction[reac, source, 1, forward, False];
    \_faux

    reac = "B15>C15+Bm;";
    (* !04/11/2010 *)
    forward[T9_] := Log[2] / (9.87*^-3) ;
    logarithme
    AddReaction[reac, source, 1, forward, False];
    \_faux

    reac = "C9>a+a+p+Bp;";
    forward[T9_] := Log[2] / (1.26*^-1) ;
    logarithme
    AddReaction[reac, source, 1, forward, False];
    \_faux

    reac = "C10>B10+Bp;";
    forward[T9_] := Log[2] / (19.29) ;
    logarithme
    AddReaction[reac, source, 1, forward, False];
    \_faux

    reac = "C11>B11+Bp;";
    forward[T9_] := Log[2] / 1.2234*^3 ;
    logarithme
    AddReaction[reac, source, 1, forward, False];
    \_faux

    reac = "C15>N15+Bm;";
    (*28/10/2010*)
    forward[T9_] := Log[2] / 2.449 ;
    logarithme
    AddReaction[reac, source, 1, forward, False];
    \_faux

    reac = "C16>N16+Bm;";
    (*14/01/2011*)
    forward[T9_] := Log[2] / 7.4700*^-1 ;
    logarithme
    AddReaction[reac, source, 1, forward, False];
    \_faux

```

```

reac = "N12>C12+Bp;";
forward[T9_] := Log[2] / 1.100*^-2 ;
               |logarithme
AddReaction[reac, source, 1, forward, False];
               |faux

reac = "N13>C13+Bp;";
(*14/01/2011*)
forward[T9_] := Log[2] / 5.979*^2 ;
               |logarithme
AddReaction[reac, source, 1, forward, False];
               |faux

reac = "N16>O16+Bm;";
(*14/01/2011*)
forward[T9_] := Log[2] / 7.13 ;
               |logarithme
AddReaction[reac, source, 1, forward, False];
               |faux

reac = "N17>O16+n+Bm;";
(*14/01/2011*)
forward[T9_] := Log[2] / 4.1730 ;
               |logarithme
AddReaction[reac, source, 1, forward, False];
               |faux

reac = "O13>N13+Bp;";
(*14/01/2011*)
forward[T9_] := Log[2] / 8.58*^-3 ;
               |logarithme
AddReaction[reac, source, 1, forward, False];
               |faux

reac = "O14>N14+Bp;";
(*14/01/2011*)
forward[T9_] := Log[2] / 70.598 ;
               |logarithme
AddReaction[reac, source, 1, forward, False];
               |faux

reac = "O15>N15+Bp;";
(*14/01/2011*)
forward[T9_] := Log[2] / 122.24;
               |logarithme
AddReaction[reac, source, 1, forward, False];
               |faux

reac = "O19>F19+Bm;";

```

```

(*14/01/2011*)
forward[T9_] := Log[2] / 26.464;
               logarithme

AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "O20>F20+Bm";
(*14/01/2011*)
forward[T9_] := Log[2] / 13.51;
               logarithme

AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "F17>O17+Bp";
(*04/11/2010*)
forward[T9_] := Log[2] / 64.49;
               logarithme

AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "F18>O18+Bp";
(*04/11/2010*)
forward[T9_] := Log[2] / 6.5863*^3;
               logarithme

AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "F20>Ne20+Bm";
(*04/11/2010*)
forward[T9_] := Log[2] / 11.1630;
               logarithme

AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "Ne18>F18+Bp";
(*04/11/2010*)
forward[T9_] := Log[2] / 1.6720;
               logarithme

AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "Ne19>F19+Bp";
(*04/11/2010*)
forward[T9_] := Log[2] / 17.296;
               logarithme

AddReaction[reac, source, 1, forward, False];
               \_faux

reac = "Ne23>Na23+Bm";
(*04/11/2010*)

```



```

forward[T9_] := Log[2] / 37.240;
               |logarithme

AddReaction[reac, source, 1, forward, False];
               |faux

reac = "Na20>Ne20+Bp;";
(*14/01/2011*)
forward[T9_] := Log[2] / 4.4790*^-1;
               |logarithme

AddReaction[reac, source, 1, forward, False];
               |faux

reac = "Na21>Ne21+Bp;";
(*04/11/2010*)
forward[T9_] := Log[2] / 22.49;
               |logarithme

AddReaction[reac, source, 1, forward, False];
               |faux

(* =====
   *New reactions following Thomas,Schramm et al.1993;1994
   *=====
   =====*)

source = "Efr96";
reac = "He4 + 2n  > He6 + g ";
f = 3.;
forward[T9_] := If[T9 < 2,
               |si
               (2.65*^-3 * T9^2.555 * Exp[0.181 / Max[T9, .1]]),
               |exponentielle |maximum
               (2.93*^-1 * T9^(-3.51*^-1) * Exp[-5.24 / T9])];
               |exponentielle

AddReaction[reac, source, f, forward, True];
               |vrai

source = "Iga95";
reac = "O16 + n  > O17 + g ";
f = 3.;
forward[T9_] := (2.7*^1 + 1.38*^4 * T9 );
AddReaction[reac, source, f, forward, True];
               |vrai

source = "CF88";
reac = "N14 + n  > C14 + p ";
f = 3.;
forward[T9_] :=
  With[{T912 = T9^ (1 / 2)}, ( 7.19*^5 * (1. + .361 * T912 + .502 * T9) +
  |avec

```

```

3.34*^8 / T912 * Exp[-4.983 / T9]) * .333];
      |exponentielle
AddReaction[reac, source, f, forward, True];
      |vrai

source = "CF88";
reac = "O14 + n > N14 + p ";
f = 3.;
forward[T9_] :=
  With[{T912 = T9^(1/2)}, (6.74*^7 * (1. + 0.658 * T912 + 0.379 * T9) * 2.99)];
      |avec
AddReaction[reac, source, f, forward, True];
      |vrai

source = "Wie87";
reac = "O14 + a > Ne18 + g ";
f = 3.;
forward[T9_] :=
  With[{T932 = T9^(3/2)}, (1.16*^-1 / T932 * Exp[-11.73 / T9] +
      |avec      |exponentielle
      3.40*^1 / T932 * Exp[-22.61 / T9] + 9.10*^-3 * T9^5 * Exp[-12.159])];
      |exponentielle      |exponentielle
AddReaction[reac, source, f, forward, True];
      |vrai

source = "NACRE";
reac = "C11 + a > N14 + p ";
f = 2.;
forward[T9_] := With[{T913 = T9^(1/3), T92 = T9^2},
      |avec
      (0.2719 * 3.01*^16 * Exp[-31.884 / T913] *
      |exponentielle
      Exp[-1.379 * T9 + .215 * T92 - 2.13*^-2 * T92 * T9 + 8*^-4 * T92 * T92] *
      |exponentielle
      (1. + 0.14 * Exp[-.275 / T9 - .210 * T9]) )];
      |exponentielle
AddReaction[reac, source, f, forward, True];
      |vrai

source = "Bar97C";
reac = "O14 + a > F17 + p ";
f = 3.;
forward[T9_] := With[{T932 = T9^(3/2), T923 = T9^(2/3),
      |avec
      T913 = T9^(1/3), T943 = T9^(4/3), T953 = T9^(5/3)},
      With[{offset = 1.330*^5 / T932 * Exp[-11.86 / T9] +
      |avec      |exponentielle

```

```

      8.42*^-47 * T932 * Exp[-0.453 / T9] + 6.74*^4 / T932 * Exp[-13.60 / T9] +
      1.21*^7 / T932 * Exp[-22.51 / T9] + 1.26*^8 / T932 * Exp[-26.00 / T9] },
      (offset + If[T9 < 1,
      7.906*^15 / T923 * Exp[-40.33 / T913] * (1. - 1.884*^1 * T913 + 2.446*^2 *
      T923 - 7.735*^2 * T9 + 9.485*^2 * T943 - 3.961*^2 * T953), 0]) ]]);
AddReaction[reac, source, f, forward, True];

source = "Koe91";
reac = " O17 + n > C14 + a ";
f = 3.;
forward[T9_] := With[{T932 = T9^(3/2)}, (3.11*^4 +
9.18*^5 / T932 * Exp[-1.961 / T9] + 7.02*^7 / T932 * Exp[-2.759 / T9])];
AddReaction[reac, source, f, forward, True];

source = "NACRE";
reac = "F17 + n > N14 + a ";
f = 1.05;
forward[T9_] := (1.38*^8 * T9^0.053 * Exp[-(55.0 - 54.943) / T9] *
(1. + .039 * Exp[-.012 / T9 + .217 * T9]) / 1.478);
AddReaction[reac, source, f, forward, True];

source = "CF88";
reac = "F18 + n > N15 + a ";
f = 3.;
forward[T9_] :=
With[{T912 = T9^(1/2)}, (3.14*^8 * (1. - 0.641 * T912 + 0.108 * T9) * 2.);
AddReaction[reac, source, f, forward, True];

source = "Kaw91";
reac = "C14 + d > N15 + n ";
f = 3.;
forward[T9_] :=
With[{T923 = T9^(2/3)}, (4.27*^13 / T923 * Exp[-16.939])];

```

```

AddReaction[reac, source, f, forward, True];
vrai

source = "CF88";
reac = "p + p + n > d + p ";
f = 3.;
forward[T9_] :=
  With[{T923 = T9^(2/3), T913 = T9^(1/3)}, (1.35*^7 * Exp[-3.720/T913] *
avec exponentielle
    (1. + 0.784 * T913 + 0.346 * T923 + 0.690 * T9) / 2.3590*^9)];
AddReaction[reac, source, f, forward, True];
vrai

source = "Kaw91";
reac = "C14 + n > C15 + g ";
f = 3.;
forward[T9_] := (3240. * T9);
AddReaction[reac, source, f, forward, True];
vrai

source = "CF88";
reac = " 016 + p > N13 + a ";
f = 3.;
forward[T9_] := With[{T953 = T9^(5/3), T932 = T9^(3/2)},
avec
  With[{T9A = T9 /
avec
    (1. + 7.76*^-2 * T9 + 2.64*^-2 * T953 / (1. + 7.76*^-2 * T9)^(2./3.))},
    With[{T9A13 = T9A^(1./3.), T9A56 = T9A^(5./6.)},
avec
      With[
avec
        {SVRev = 1.88*^18 * T9A56 / T932 * Exp[-35.829/T9A13] * 1.7232*^-1},
exponentielle
        With[{SVDDir = SVRev / 0.172255 * Exp[-60.5573/T9]},
avec exponentielle
          SVDDir]]]]];
AddReaction[reac, source, f, forward, True];
vrai

(* %TUNL&Cam08 !Camargo et al.Phys.Rev.C 78,034605 (2008) pour DC
constante C
!Tilley (TUNL) Table 9.5 pour la resonance a 87 keV (dominante) *)
table
source = "TUNL&Cam08";

```

```

    reac = "Li8 + p > Be9 + g ";
    f = 3.;
    forward[T9_] :=
      With[{T923 = T9 ^ (2 / 3), T913 = T9 ^ (1 / 3), T932 = T9 ^ (3 / 2)},
        _avec
        (3.516*^6 / T923 * Exp[-8.5155 / T913] +
          _exponentielle
          2.669*^4 / T932 * Exp[-1.010 / T9] ) ];
    AddReaction[reac, source, f, forward, True];
    _vrai

    source = "Wan91";
    reac = "B11 + a > N15 + g ";
    f = 3.;
    forward[T9_] :=
      With[{T932 = T9 ^ (3 / 2)}, (643. / T932 * Exp[-5.1526 / T9] ) ];
      _avec _exponentielle
    AddReaction[reac, source, f, forward, True];
    _vrai

  ];

  If[reslist[[2]] === {}, {}, reslist[[2, 1]]
_si

  (* The output is the list of reactions in standard
  format (Name,_liste initial,_liste List final,f factor) which is
  then used by the differential equation constructor *)

];

```

Collecting all reaction rates

We collect the description of all rates in a single table (ListReactions). This include the weak rate (1 reaction), all the reactions from the external file (generated by the function LoadRates), and the reactions which were given analytically by the function DefineAnalyticRates.

In[670]:=

```

ReactionPEN = {"nT0p", {"n"}, {"p"}, 0, "Companion Paper"};
(* Format is name, List of initial particles,
  [format] [liste]
  List of final particles, f factor for uncertainty*)
  [liste]

TabulatedReactions :=
  (Select[SafeImport[TabulatedReactionsFile],
    [sélectionne]
    (NumericQ[#[[1]]] || "*" == #[[1]] || StringMatchQ[#[[1]], "\\*%" ~~ __) &]);
    [expression numérique ?] [corresponds à chaîne de caractères ?]
  ReshapedTabulatedReactionsAll := TreatData[TabulatedReactions]

ReshapedTabulatedReactions = If[$ReducedNetwork,
  [si]
  Take[ReshapedTabulatedReactionsAll, SmallNuclearNetworkSize],
  [prends]
  ReshapedTabulatedReactionsAll];

```

In[674]:=

ReshapedTabulatedReactions

Out[674]=

```

{{{n, p, >, d, g, ;}, { ... 1 ... }, And06,
  {{0.001, 44140., 1.0045}, {0.002, 44065., 1.0045}, {0.003, 43990., 1.0045},
   {0.004, 43916., 1.0045}, {0.005, 43842., 1.0045}, {0.006, 43769., 1.0045},
   {0.007, 43696., 1.0045}, {0.008, 43624., 1.0045}, {0.009, 43552., 1.0045},
   {0.01, 43481., 1.0045}, {0.011, 43410., 1.0045}, {0.012, 43340., 1.0045},
   {0.013, 43270., 1.0045}, {0.014, 43200., 1.0045}, {0.015, 43131., 1.0045},
   {0.016, 43062., 1.0045}, {0.018, 42926., 1.0045}, {0.02, 42792., 1.0045},
   ... 24 ..., {0.7, 27560., 1.0044}, {0.8, 26962., 1.0044}, {0.9, 26504., 1.0044},
   {1., 26158., 1.0044}, {1.25, 25657., 1.0045}, {1.5, 25517., 1.0046}, {1.75, 25615., 1.0048},
   {2., 25873., 1.0049}, {2.5, 26696., 1.0053}, {3., 27756., 1.0056}, {3.5, 28937., 1.0058},
   {4., 30176., 1.0061}, {5., 32699., 1.0064}, {6., 35177., 1.0067}, {7., 37553., 1.0069},
   {8., 39810., 1.0071}, {9., 41943., 1.0072}, {10., 43957., 1.0074}}}, { ... 335 ... }, { ... 1 ... }}

```

Taille de la mémoire : 2.6 MB

+ Afficher plus

Afficher tout

Iconiser ▼



Stocker l'expression complète dans le notebook

In[675]:=

```

LoadRates := (
  (* The list of reactions which are tabulated in external .dat file*)
  ListReactionsFile = TreatReactionLine /@ ReshapedTabulatedReactions;

  (* The list of reactions defined analytically *)
  If[Not[$ReducedNetwork],
    _si _négation
    ExtraAnalyticReactions = DefineAnalyticRates;;
    ExtraAnalyticReactions = {};
  ];

  (* We concatenate the weak reactions,
  the tabulated rates and the analytic rates*)
  ListReactions =
    Join[{ReactionPEN}, ListReactionsFile, ExtraAnalyticReactions];
    _joins
  );

```

LoadRates is the function which does all that. Let us call it. It will stop and quit if one of the reactions is inconsistent (not conservation of N nor Z).

In[676]:=

```
LoadRates;
```

... **General** : Exp[-7261.2137] is too small to represent as a normalized machine number; precision may be lost. i

... **General** : Exp[-7191.8819] is too small to represent as a normalized machine number; precision may be lost. i

... **General** : Exp[-7123.2122] is too small to represent as a normalized machine number; precision may be lost. i

... **General** : Further output of `General::munfl` will be suppressed during this calculation. i

We now restrict the nuclides up to a maximum mass. Useless if MaximumNuclearMass has been set to Infinity.

In[677]:=

```

SpeciesUpToMaximumMass[A_Integer] :=
  [nombre entier]
  SpeciesUpToMaximumMass[A] = Union@@Table[NamesMassNumberAll[i], {i, A}];
  [union [table]
NonBaryonicParticle[key_] := key == "g" || key == "Bm" || key == "Bm";
ReactionUpToMaximumMass[A_Integer][Reaction_List] :=
  [nombre entier [liste]
  And@@((MemberQ[SpeciesUpToMaximumMass[A], #] || NonBaryonicParticle[#]) & /@
  [et [appartient ?]
  Flatten@Reaction[[2 ;; 3]]);
  [aplatis]
ListReactionsUpToMass[A_Integer, ListReactions_List] :=
  [nombre entier [liste]
  Select[ListReactions, ReactionUpToMaximumMass[A][#] &];
  [sélectionne]
ListReactionsUpToMass[Infinity, ListReactions_List] := ListReactions;
  [infini [liste]
ListReactionsUpToChosenMass :=
  ListReactionsUpToMass[MaximumNuclearMass, ListReactions];

```

For information let us print the list of reactions which are taken into account.

In[683]:=

```

ReactionWithArrow[name_String] := StringReplace[name, "T0" -> " -> "]
  [chaîne de caractères [remplace dans chaîne de caractères]
NiceDisplayReaction[reaction_List] :=
  [liste]
  Join[{ReactionWithArrow[First[reaction]]}, Rest[reaction]]
  [joins [premier [reste]
PadLeftNumberRow[tab_List] :=
  [liste]
  MapThread[Prepend[#2, #1] &, {Range@Length@tab, tab}]
  [applique en ... [ajoute en début [plage [longueur]
PadLeftNumberRowFromZero[tab_List] :=
  [liste]
  MapThread[Prepend[#2, #1] &, {Range@Length@tab - 1, tab}]
  [applique en ... [ajoute en début [plage [longueur]

```

In[687]:=

```

MyGrid[Join[{"Reaction Number", "Reaction Name", "Initial species",
  [joins [nombre]
  "Final Species", "Reference"}], PadLeftNumberRowFromZero[
  Drop[#, {4}] & /@ (NiceDisplayReaction /@ ListReactionsUpToChosenMass)]]]
  [laisse tomber]

```

Out[687]=

Reaction Number	Reaction Name	Initial species	Final Species	Reference	
0	n -> p	{n}	{p}	Companion Paper	
1	np -> dg	{n, p}	{d, g}	And06	
2	dp -> He3g	{d, p}	{He3, g}	Moscoso2021	
3	dd -> He3n	{d, d}	{He3, n}	Gom17	

4	dd -> tp	{d, d}	{t, p}	Gom17	
5	tp -> ag	{t, p}	{a, g}	Ser04	
6	td -> an	{t, d}	{a, n}	deSouza19a	
7	ta -> Li7g	{t, a}	{Li7, g}	DAACV04	
8	He3n -> tp	{He3, n}	{t, p}	DAACV04	
9	He3d -> ap	{He3, d}	{a, p}	deSouza19b	
10	He3a -> Be7g	{He3, a}	{Be7, g}	Ili16	
11	Be7n -> Li7p	{Be7, n}	{Li7, p}	deSouza2020	
12	Li7p -> aa	{Li7, p}	{a, a}	DAACV04	
13	Li7p -> aag	{Li7, p}	{a, a, g}	NACRE	
14	Be7n -> aa	{Be7, n}	{a, a}	Bar16	
15	Be7d -> aap	{Be7, d}	{a, a, p}	Rij19	
16	da -> Li6g	{d, a}	{Li6, g}	Ham10	
17	Li6p -> Be7g	{Li6, p}	{Be7, g}	NACRE	
18	Li6p -> He3a	{Li6, p}	{He3, a}	NACRE	
19	Be9t -> B11n	{Be9, t}	{B11, n}	TALYS2	
20	O18n -> O19g	{O18, n}	{O19, g}	TALYS2	
21	Li9p -> He6a	{Li9, p}	{He6, a}	=li7pa	
22	Li9d -> Be10n	{Li9, d}	{Be10, n}	TALYS2	
23	Be10a -> C14g	{Be10, a}	{C14, g}	TALYS2	
24	N12n -> C12p	{N12, n}	{C12, p}	TALYS2	
25	Li9p -> Be9n	{Li9, p}	{Be9, n}	TALYS2	
26	Li9a -> B12n	{Li9, a}	{B12, n}	CGXSV12	
27	Li9p -> Be10g	{Li9, p}	{Be10, g}	TALYS2	
28	N13n -> N14g	{N13, n}	{N14, g}	TALYS2	
29	B10a -> N14g	{B10, a}	{N14, g}	TALYS2	
30	B8a -> N12g	{B8, a}	{N12, g}	TALYS2	
31	B12p -> Be9a	{B12, p}	{Be9, a}	TALYS2	
32	Be10p -> B11g	{Be10, p}	{B11, g}	TALYS2	
33	Be10p -> Li7a	{Be10, p}	{Li7, a}	TALYS2	
34	Be11p -> Li8a	{Be11, p}	{Li8, a}	TALYS2	
35	Be11p -> B11n	{Be11, p}	{B11, n}	TALYS2	
36	B8n -> aap	{B8, n}	{a, a, p}	TALYS2	
37	B10n -> B11g	{B10, n}	{B11, g}	TALYS2	
38	B10a -> C13p	{B10, a}	{C13, p}	TALYS2	
39	O17n -> O18g	{O17, n}	{O18, g}	TALYS2	
40	F17n -> O17p	{F17, n}	{O17, p}	TALYS2	
41	F18n -> O18p	{F18, n}	{O18, p}	TALYS2	
42	Be10a -> C13n	{Be10, a}	{C13, n}	TALYS2	
43	Be11a -> C14n	{Be11, a}	{C14, n}	TALYS2	
44	N14a -> F18g	{N14, a}	{F18, g}	ILCCF10	
45	N15a -> F19g	{N15, a}	{F19, g}	ILCCF10	
46	O15a -> Ne19g	{O15, a}	{Ne19, g}	ILCCF10	
47	O16p -> F17g	{O16, p}	{F17, g}	ILCCF10	
48	O16a -> Ne20g	{O16, a}	{Ne20, g}	ILCCF10	
49	O17p -> F18g	{O17, p}	{F18, g}	ILCCF10	
50	O18p -> F19g	{O18, p}	{F19, g}	ILCCF10	
51	O18a -> Ne22g	{O18, a}	{Ne22, g}	ILCCF10	
52	F17p -> Ne18g	{F17, p}	{Ne18, g}	ILCCF10	
53	F18p -> Ne19g	{F18, p}	{Ne19, g}	ILCCF10	
54	Ne19p -> Na20g	{Ne19, p}	{Na20, g}	ILCCF10	

55	017p -> N14a	{017, p}	{N14, a}	ILCCF10	
56	018p -> N15a	{018, p}	{N15, a}	ILCCF10	
57	F18p -> O15a	{F18, p}	{O15, a}	ILCCF10	
58	C14a -> O18g	{C14, a}	{O18, g}	ILCCF10	
59	C14p -> N15g	{C14, p}	{N15, g}	ILCCF10	
60	Be12p -> Li9a	{Be12, p}	{Li9, a}	TALYS2	
61	Li6He3 -> aap	{Li6, He3}	{a, a, p}	TALYS2	
62	Li6t -> Be9g	{Li6, t}	{Be9, g}	TALYS2	
63	Li6t -> aan	{Li6, t}	{a, a, n}	TALYS2	
64	Li6t -> Li8p	{Li6, t}	{Li8, p}	TALYS2	
65	Li7d -> Be9g	{Li7, d}	{Be9, g}	CGXSV12	
66	Li7He3 -> B10g	{Li7, He3}	{B10, g}	TALYS2	
67	Li7He3 -> Li6a	{Li7, He3}	{Li6, a}	TALYS2	
68	Li7t -> Be10g	{Li7, t}	{Be10, g}	TALYS2	
69	Li8a -> B12g	{Li8, a}	{B12, g}	TALYS2	
70	Li8a -> B11n	{Li8, a}	{B11, n}	CGXSV12	
71	Li8d -> Be10g	{Li8, d}	{Be10, g}	TALYS2	
72	Li8He3 -> B11g	{Li8, He3}	{B11, g}	TALYS2	
73	Li8He3 -> B10n	{Li8, He3}	{B10, n}	TALYS2	
74	Li8He3 -> Be10p	{Li8, He3}	{Be10, p}	TALYS2	
75	Li8He3 -> Li7a	{Li8, He3}	{Li7, a}	TALYS2	
76	Li8t -> Be11g	{Li8, t}	{Be11, g}	TALYS2	
77	Li8t -> Be10n	{Li8, t}	{Be10, n}	TALYS2	
78	Li9a -> B13g	{Li9, a}	{B13, g}	TALYS2	
79	Li9d -> Be11g	{Li9, d}	{Be11, g}	TALYS2	
80	Li9He3 -> B12g	{Li9, He3}	{B12, g}	TALYS2	
81	Li9He3 -> B11n	{Li9, He3}	{B11, n}	TALYS2	
82	Li9He3 -> Be11p	{Li9, He3}	{Be11, p}	TALYS2	
83	Li9He3 -> Li8a	{Li9, He3}	{Li8, a}	TALYS2	
84	Li9t -> Be12g	{Li9, t}	{Be12, g}	TALYS2	
85	Li9t -> Be11n	{Li9, t}	{Be11, n}	TALYS2	
86	Be7He3 -> C10g	{Be7, He3}	{C10, g}	TALYS2	
87	Be7t -> B10g	{Be7, t}	{B10, g}	TALYS2	
88	Be7t -> Be9p	{Be7, t}	{Be9, p}	CGXSV12	
89	Be7t -> Li6a	{Be7, t}	{Li6, a}	TALYS2	
90	Be9a -> C13g	{Be9, a}	{C13, g}	TALYS2	
91	Be9d -> B11g	{Be9, d}	{B11, g}	TALYS2	
92	Be9d -> B10n	{Be9, d}	{B10, n}	TALYS2	
93	Be9d -> Be10p	{Be9, d}	{Be10, p}	TALYS2	
94	Be9d -> Li7a	{Be9, d}	{Li7, a}	TALYS2	
95	Be9He3 -> C12g	{Be9, He3}	{C12, g}	TALYS2	
96	Be9He3 -> C11n	{Be9, He3}	{C11, n}	TALYS2	
97	Be9He3 -> B11p	{Be9, He3}	{B11, p}	TALYS2	
98	Be9He3 -> aaa	{Be9, He3}	{a, a, a}	TALYS2	
99	Be9t -> B12g	{Be9, t}	{B12, g}	TALYS2	
100	Be9t -> Li8a	{Be9, t}	{Li8, a}	TALYS2	
101	Be10d -> B12g	{Be10, d}	{B12, g}	TALYS2	
102	Be10d -> B11n	{Be10, d}	{B11, n}	TALYS2	
103	Be10d -> Li8a	{Be10, d}	{Li8, a}	TALYS2	

104	Be10He3 -> C13g	{Be10, He3}	{C13, g}	TALYS2	
105	Be10He3 -> C12n	{Be10, He3}	{C12, n}	TALYS2	
106	Be10He3 -> B12p	{Be10, He3}	{B12, p}	TALYS2	
107	Be10He3 -> Be9a	{Be10, He3}	{Be9, a}	TALYS2	
108	Be10t -> B13g	{Be10, t}	{B13, g}	TALYS2	
109	Be10t -> B12n	{Be10, t}	{B12, n}	TALYS2	
110	Be10t -> Li9a	{Be10, t}	{Li9, a}	TALYS2	
111	Be11a -> C15g	{Be11, a}	{C15, g}	TALYS2	
112	Be11d -> B13g	{Be11, d}	{B13, g}	TALYS2	
113	Be11d -> B12n	{Be11, d}	{B12, n}	TALYS2	
114	Be11d -> Be12p	{Be11, d}	{Be12, p}	TALYS2	
115	Be11d -> Li9a	{Be11, d}	{Li9, a}	TALYS2	
116	Be11He3 -> C14g	{Be11, He3}	{C14, g}	TALYS2	
117	Be11He3 -> C13n	{Be11, He3}	{C13, n}	TALYS2	
118	Be11He3 -> B13p	{Be11, He3}	{B13, p}	TALYS2	
119	Be11He3 -> Be10a	{Be11, He3}	{Be10, a}	TALYS2	
120	Be11p -> B12g	{Be11, p}	{B12, g}	TALYS2	
121	Be11t -> B14g	{Be11, t}	{B14, g}	TALYS2	
122	Be11t -> B13n	{Be11, t}	{B13, n}	TALYS2	
123	Be12a -> C16g	{Be12, a}	{C16, g}	TALYS2	
124	Be12a -> C15n	{Be12, a}	{C15, n}	TALYS2	
125	Be12d -> B14g	{Be12, d}	{B14, g}	TALYS2	
126	Be12d -> B13n	{Be12, d}	{B13, n}	TALYS2	
127	Be12He3 -> C15g	{Be12, He3}	{C15, g}	TALYS2	
128	Be12He3 -> C14n	{Be12, He3}	{C14, n}	TALYS2	
129	Be12He3 -> B14p	{Be12, He3}	{B14, p}	TALYS2	
130	Be12He3 -> Be11a	{Be12, He3}	{Be11, a}	TALYS2	
131	Be12p -> B13g	{Be12, p}	{B13, g}	TALYS2	
132	Be12p -> B12n	{Be12, p}	{B12, n}	TALYS2	
133	Be12t -> B15g	{Be12, t}	{B15, g}	TALYS2	
134	Be12t -> B14n	{Be12, t}	{B14, n}	TALYS2	
135	B8a -> C11p	{B8, a}	{C11, p}	TALYS2	
136	B8d -> C10g	{B8, d}	{C10, g}	TALYS2	
137	B8He3 -> C10p	{B8, He3}	{C10, p}	TALYS2	
138	B8t -> C11g	{B8, t}	{C11, g}	TALYS2	

139	B8t -> C10n	{B8, t}	{C10, n}	TALYS2	
140	B8t -> B10p	{B8, t}	{B10, p}	TALYS2	
141	B8t -> Be7a	{B8, t}	{Be7, a}	TALYS2	
142	B10d -> C12g	{B10, d}	{C12, g}	TALYS2	
143	B10d -> C11n	{B10, d}	{C11, n}	TALYS2	
144	B10d -> B11p	{B10, d}	{B11, p}	TALYS2	
145	B10d -> aaa	{B10, d}	{a, a, a}	TALYS2	
146	B10He3 -> N13g	{B10, He3}	{N13, g}	TALYS2	
147	B10He3 -> N12n	{B10, He3}	{N12, n}	TALYS2	
148	B10He3 -> C12p	{B10, He3}	{C12, p}	TALYS2	
149	B10n -> Be10p	{B10, n}	{Be10, p}	TALYS2	
150	B10t -> C13g	{B10, t}	{C13, g}	TALYS2	
151	B10t -> C12n	{B10, t}	{C12, n}	TALYS2	
152	B10t -> B12p	{B10, t}	{B12, p}	TALYS2	
153	B10t -> Be9a	{B10, t}	{Be9, a}	TALYS2	
154	B11d -> C13g	{B11, d}	{C13, g}	TALYS2	
155	B11d -> C12n	{B11, d}	{C12, n}	CGXSV12	
156	B11d -> B12p	{B11, d}	{B12, p}	CGXSV12	
157	B11d -> Be9a	{B11, d}	{Be9, a}	TALYS2	
158	B11He3 -> N14g	{B11, He3}	{N14, g}	TALYS2	
159	B11He3 -> N13n	{B11, He3}	{N13, n}	TALYS2	
160	B11He3 -> C13p	{B11, He3}	{C13, p}	TALYS2	
161	B11He3 -> B10a	{B11, He3}	{B10, a}	TALYS2	
162	B11t -> C14g	{B11, t}	{C14, g}	TALYS2	
163	B11t -> C13n	{B11, t}	{C13, n}	TALYS2	
164	B11t -> Be10a	{B11, t}	{Be10, a}	TALYS2	
165	B12a -> N16g	{B12, a}	{N16, g}	TALYS2	
166	B12p -> C12n	{B12, p}	{C12, n}	TALYS2	
167	B12a -> N15n	{B12, a}	{N15, n}	TALYS2	
168	B12d -> C14g	{B12, d}	{C14, g}	TALYS2	
169	B12d -> C13n	{B12, d}	{C13, n}	TALYS2	
170	B12d -> B13p	{B12, d}	{B13, p}	TALYS2	
171	B12d -> Be10a	{B12, d}	{Be10, a}	TALYS2	
172	B12He3 -> N15g	{B12, He3}	{N15, g}	TALYS2	
173	B12He3 -> N14n	{B12, He3}	{N14, n}	TALYS2	
174	B12He3 -> C14p	{B12, He3}	{C14, p}	TALYS2	
175	B12He3 -> B11a	{B12, He3}	{B11, a}	TALYS2	
176	B12n -> B13g	{B12, n}	{B13, g}	TALYS2	
177	B12p -> C13g	{B12, p}	{C13, g}	TALYS2	
178	B12t -> C15g	{B12, t}	{C15, g}	TALYS2	
179	B12t -> C14n	{B12, t}	{C14, n}	TALYS2	
180	B12t -> Be11a	{B12, t}	{Be11, a}	TALYS2	
181	C9a -> O13g	{C9, a}	{O13, g}	TALYS2	
182	C9d -> C10p	{C9, d}	{C10, p}	TALYS2	
183	C9n -> C10g	{C9, n}	{C10, g}	TALYS2	
184	C9t -> N12g	{C9, t}	{N12, g}	TALYS2	
185	C9t -> C11p	{C9, t}	{C11, p}	TALYS2	
186	C9t -> B8a	{C9, t}	{B8, a}	TALYS2	
187	C11d -> N13g	{C11, d}	{N13, g}	TALYS2	
188	C11d -> C12p	{C11, d}	{C12, p}	CGXSV12	
189	C11He3 -> O14g	{C11, He3}	{O14, g}	TALYS2	

190	C11He3 -> N13p	{C11, He3}	{N13, p}	TALYS2	
191	C11He3 -> C10a	{C11, He3}	{C10, a}	TALYS2	
192	C11t -> N14g	{C11, t}	{N14, g}	TALYS2	
193	C11t -> N13n	{C11, t}	{N13, n}	TALYS2	
194	C11t -> C13p	{C11, t}	{C13, p}	TALYS2	
195	C11t -> B10a	{C11, t}	{B10, a}	TALYS2	
196	C12a -> O16g	{C12, a}	{O16, g}	NACRE	
197	C12d -> N14g	{C12, d}	{N14, g}	TALYS2	
198	C12d -> C13p	{C12, d}	{C13, p}	TALYS2	
199	C12He3 -> O15g	{C12, He3}	{O15, g}	TALYS2	
200	C12He3 -> N14p	{C12, He3}	{N14, p}	TALYS2	
201	C11a -> O15g	{C11, a}	{O15, g}	TALYS2	
202	C12He3 -> C11a	{C12, He3}	{C11, a}	TALYS2	
203	C12n -> C13g	{C12, n}	{C13, g}	TALYS2	
204	C12p -> N13g	{C12, p}	{N13, g}	NACRE	
205	C12t -> N15g	{C12, t}	{N15, g}	TALYS2	
206	C12t -> N14n	{C12, t}	{N14, n}	TALYS2	
207	C12t -> C14p	{C12, t}	{C14, p}	TALYS2	
208	C12t -> B11a	{C12, t}	{B11, a}	TALYS2	
209	C13a -> O17g	{C13, a}	{O17, g}	TALYS2	
210	C13d -> N15g	{C13, d}	{N15, g}	TALYS2	
211	C13d -> N14n	{C13, d}	{N14, n}	TALYS2	
212	C13d -> C14p	{C13, d}	{C14, p}	TALYS2	
213	C13d -> B11a	{C13, d}	{B11, a}	TALYS2	
214	C13He3 -> O16g	{C13, He3}	{O16, g}	TALYS2	
215	C13He3 -> O15n	{C13, He3}	{O15, n}	TALYS2	
216	C13He3 -> N15p	{C13, He3}	{N15, p}	TALYS2	
217	C13He3 -> C12a	{C13, He3}	{C12, a}	TALYS2	
218	C13n -> C14g	{C13, n}	{C14, g}	TALYS2	
219	C13p -> N14g	{C13, p}	{N14, g}	NACRE	
220	C13t -> N16g	{C13, t}	{N16, g}	TALYS2	
221	C13t -> N15n	{C13, t}	{N15, n}	TALYS2	
222	C13t -> C15p	{C13, t}	{C15, p}	TALYS2	
223	C13t -> B12a	{C13, t}	{B12, a}	TALYS2	
224	C14d -> N16g	{C14, d}	{N16, g}	TALYS2	
225	C14d -> B12a	{C14, d}	{B12, a}	TALYS2	
226	C14He3 -> O17g	{C14, He3}	{O17, g}	TALYS2	
227	C14He3 -> O16n	{C14, He3}	{O16, n}	TALYS2	
228	C14He3 -> N16p	{C14, He3}	{N16, p}	TALYS2	
229	C14He3 -> C13a	{C14, He3}	{C13, a}	TALYS2	
230	C14t -> N17g	{C14, t}	{N17, g}	TALYS2	
231	C14t -> N16n	{C14, t}	{N16, n}	TALYS2	
232	C15a -> O19g	{C15, a}	{O19, g}	TALYS2	
233	C15a -> O18n	{C15, a}	{O18, n}	TALYS2	
234	C15n -> C16g	{C15, n}	{C16, g}	TALYS2	
235	C15p -> N16g	{C15, p}	{N16, g}	TALYS2	
236	C15p -> N15n	{C15, p}	{N15, n}	TALYS2	
237	C15p -> B12a	{C15, p}	{B12, a}	TALYS2	
238	N12a -> O15p	{N12, a}	{O15, p}	TALYS2	
239	N12n -> N13g	{N12, n}	{N13, g}	TALYS2	
240	N12p -> O13g	{N12, p}	{O13, g}	TALYS2	

241	N13a -> F17g	{N13, a}	{F17, g}	TALYS2	
242	N13n -> C13p	{N13, n}	{C13, p}	TALYS2	
243	N13p -> O14g	{N13, p}	{O14, g}	NACRE	
244	N14n -> N15g	{N14, n}	{N15, g}	TALYS2	
245	N14p -> O15g	{N14, p}	{O15, g}	NACRE	
246	N15n -> N16g	{N15, n}	{N16, g}	TALYS2	
247	N15p -> O16g	{N15, p}	{O16, g}	NACRE	
248	O14n -> O15g	{O14, n}	{O15, g}	TALYS2	
249	O14n -> C11a	{O14, n}	{C11, a}	TALYS2	
250	O15n -> O16g	{O15, n}	{O16, g}	TALYS2	
251	O15n -> N15p	{O15, n}	{N15, p}	TALYS2	
252	O15n -> C12a	{O15, n}	{C12, a}	TALYS2	
253	O17a -> Ne21g	{O17, a}	{Ne21, g}	CF88	
254	O17a -> Ne20n	{O17, a}	{Ne20, n}	NACRE	
255	O19a -> Ne23g	{O19, a}	{Ne23, g}	TALYS2	
256	O19a -> Ne22n	{O19, a}	{Ne22, n}	TALYS2	
257	O19n -> O20g	{O19, n}	{O20, g}	TALYS2	
258	O19p -> F20g	{O19, p}	{F20, g}	TALYS2	
259	O19p -> F19n	{O19, p}	{F19, n}	TALYS2	
260	O19p -> N16a	{O19, p}	{N16, a}	TALYS2	
261	F17a -> Na21g	{F17, a}	{Na21, g}	TALYS2	
262	F17a -> Ne20p	{F17, a}	{Ne20, p}	TALYS2	
263	F17n -> F18g	{F17, n}	{F18, g}	TALYS2	
264	F18a -> Na22g	{F18, a}	{Na22, g}	TALYS2	
265	F18a -> Ne21p	{F18, a}	{Ne21, p}	TALYS2	
266	F18n -> F19g	{F18, n}	{F19, g}	TALYS2	
267	F19a -> Na23g	{F19, a}	{Na23, g}	TALYS2	
268	F19a -> Ne22p	{F19, a}	{Ne22, p}	TALYS2	
269	F19n -> F20g	{F19, n}	{F20, g}	TALYS2	
270	F19p -> Ne20g	{F19, p}	{Ne20, g}	NACRE	
271	F19p -> O16a	{F19, p}	{O16, a}	NACRE	
272	B8n -> Li6He3	{B8, n}	{Li6, He3}	TALYS2	
273	Li9p -> Li7t	{Li9, p}	{Li7, t}	TALYS2	
274	B8n -> Be7d	{B8, n}	{Be7, d}	TALYS2	
275	C9n -> Be7He3	{C9, n}	{Be7, He3}	TALYS2	
276	B10n -> aat	{B10, n}	{a, a, t}	TALYS2	
277	Be10p -> aat	{Be10, p}	{a, a, t}	TALYS2	
278	Be11p -> Be9t	{Be11, p}	{Be9, t}	TALYS2	
279	Be11p -> Be10d	{Be11, p}	{Be10, d}	TALYS2	
280	Be12p -> Be10t	{Be12, p}	{Be10, t}	TALYS2	
281	C9n -> B8d	{C9, n}	{B8, d}	TALYS2	
282	N13n -> C12d	{N13, n}	{C12, d}	TALYS2	
283	B10a -> C12d	{B10, a}	{C12, d}	TALYS2	
284	O14n -> C12He3	{O14, n}	{C12, He3}	TALYS2	
285	C15p -> C14d	{C15, p}	{C14, d}	TALYS2	
286	Ne18n -> O15a	{Ne18, n}	{O15, a}	TALYS2	
287	Ne19n -> O16a	{Ne19, n}	{O16, a}	TALYS2	
288	Na20n -> F17a	{Na20, n}	{F17, a}	TALYS2	
289	Ne18n -> F18p	{Ne18, n}	{F18, p}	TALYS2	
290	Ne19n -> F19p	{Ne19, n}	{F19, p}	TALYS2	
291	Li7He3 -> Be9p	{Li7, He3}	{Be9, p}	TALYS2	

292	Li6t -> Li7d	{Li6, t}	{Li7, d}	TALYS2	
293	Li6He3 -> Be7d	{Li6, He3}	{Be7, d}	TALYS2	
294	Li7He3 -> aad	{Li7, He3}	{a, a, d}	TALYS2	
295	Li8He3 -> Be9d	{Li8, He3}	{Be9, d}	TALYS2	
296	Li8He3 -> aat	{Li8, He3}	{a, a, t}	TALYS2	
297	Li9d -> Li8t	{Li9, d}	{Li8, t}	TALYS2	
298	Li9He3 -> Be10d	{Li9, He3}	{Be10, d}	TALYS2	
299	Li9He3 -> Be9t	{Li9, He3}	{Be9, t}	TALYS2	
300	Be7t -> aad	{Be7, t}	{a, a, d}	TALYS2	
301	Be7t -> Li7He3	{Be7, t}	{Li7, He3}	TALYS2	
302	Be9d -> aat	{Be9, d}	{a, a, t}	TALYS2	
303	Be9t -> Be10d	{Be9, t}	{Be10, d}	TALYS2	
304	Be9He3 -> B10d	{Be9, He3}	{B10, d}	TALYS2	
305	Be10He3 -> B11d	{Be10, He3}	{B11, d}	TALYS2	
306	Be10He3 -> B10t	{Be10, He3}	{B10, t}	TALYS2	
307	B8d -> Be7He3	{B8, d}	{Be7, He3}	TALYS2	
308	B8t -> aaHe3	{B8, t}	{a, a, He3}	TALYS2	
309	B10p -> aaHe3	{B10, p}	{a, a, He3}	TALYS2	
310	B10t -> B11d	{B10, t}	{B11, d}	TALYS2	
311	B10He3 -> C11d	{B10, He3}	{C11, d}	TALYS2	
312	B11t -> B13p	{B11, t}	{B13, p}	TALYS2	
313	B11He3 -> C12d	{B11, He3}	{C12, d}	TALYS2	
314	N12n -> C11d	{N12, n}	{C11, d}	TALYS2	
315	C11t -> C12d	{C11, t}	{C12, d}	TALYS2	
316	C11t -> B11He3	{C11, t}	{B11, He3}	TALYS2	
317	Be7He3 -> ppaa	{Be7, He3}	{p, p, a, a}	TALYS2	
318	dd -> ag	{d, d}	{a, g}	NACRE	
319	He3He3 -> app	{He3, He3}	{a, p, p}	NACRE	
320	Li7a -> B11g	{Li7, a}	{B11, g}	NACRE	
321	Be7p -> B8g	{Be7, p}	{B8, g}	NACRE	
322	Be7a -> C11g	{Be7, a}	{C11, g}	NACRE	
323	Be9p -> B10g	{Be9, p}	{B10, g}	NACRE	
324	Be9p -> aad	{Be9, p}	{a, a, d}	NACRE	
325	Be9p -> Li6a	{Be9, p}	{Li6, a}	NACRE	
326	Be9a -> C12n	{Be9, a}	{C12, n}	NACRE	
327	B10p -> C11g	{B10, p}	{C11, g}	NACRE	
328	B10p -> Be7a	{B10, p}	{Be7, a}	NACRE	
329	B11p -> C12g	{B11, p}	{C12, g}	NACRE	
330	B11p -> aaa	{B11, p}	{a, a, a}	NACRE	
331	B11a -> N14n	{B11, a}	{N14, n}	NACRE	
332	C13a -> O16n	{C13, a}	{O16, n}	NACRE	
333	N15p -> C12a	{N15, p}	{C12, a}	NACRE	
334	Li7t -> Be9n	{Li7, t}	{Be9, n}	CGXSV12	
335	B11n -> B12g	{B11, n}	{B12, g}	CGXSV12	
336	C11n -> aaa	{C11, n}	{a, a, a}	CGXSV12	
337	Li7d -> aan	{Li7, d}	{a, a, n}	CGXSV12	
338	dn -> tg	{d, n}	{t, g}	Nag06	2.

339	tt -> ann	{t, t}	{a, n, n}	Nag06	3.
340	He3n -> ag	{He3, n}	{a, g}	Wag69	3.
341	He3t -> ad	{He3, t}	{a, d}	CF88	10.
342	He3t -> anp	{He3, t}	{a, n, p}	CF88	10.
343	aan -> Be9g	{a, a, n}	{Be9, g}	NACRE	1.25
344	Li7t -> aann	{Li7, t}	{a, a, n, n}	CF88&MF89	3.
345	Li7He3 -> aanp	{Li7, He3}	{a, a, n, p}	CF88&MF89	3.
346	Li8d -> Li9p	{Li8, d}	{Li9, p}	Bal95	3.
347	Li8d -> Li7t	{Li8, d}	{Li7, t}	Has09c	3.
348	Be7t -> aanp	{Be7, t}	{a, a, n, p}	CF88&MF89	3.
349	Be7He3 -> aapp	{Be7, He3}	{a, a, p, p}	CF88&MF89	3.
350	C9a -> N12p	{C9, a}	{N12, p}	Wie89	3.
351	Li6n -> ta	{Li6, n}	{t, a}	CF88	3.
352	He3t -> Li6g	{He3, t}	{Li6, g}	FK90	3.
353	anp -> Li6g	{a, n, p}	{Li6, g}	CF88	3.
354	Li6n -> Li7g	{Li6, n}	{Li7, g}	MF89	3.
355	Li6d -> Li7p	{Li6, d}	{Li7, p}	MF89	3.
356	Li6d -> Be7n	{Li6, d}	{Be7, n}	MF89	3.
357	Li6a -> B10g	{Li6, a}	{B10, g}	CF88	3.
358	Li7a -> B10n	{Li7, a}	{B10, n}	NACRE	1.08
359	Li7n -> Li8g	{Li7, n}	{Li8, g}	MF89&Hei98	3.
360	Li7d -> Li8p	{Li7, d}	{Li8, p}	MF89	3.
361	Li8n -> Li9g	{Li8, n}	{Li9, g}	Rau94	3.
362	Li8p -> aan	{Li8, p}	{a, a, n}	Men12	3.
363	Li8d -> Be9n	{Li8, d}	{Be9, n}	Bal95	3.
364	Be9n -> Be10g	{Be9, n}	{Be10, g}	Rau94	3.
365	Be9p -> aapn	{Be9, p}	{a, a, p, n}	NACRE	1.05
366	B11p -> C11n	{B11, p}	{C11, n}	NACRE	1.1
367	Be10n -> Be11g	{Be10, n}	{Be11, g}	Rau94	3.
368	Be11n -> Be12g	{Be11, n}	{Be12, g}	Rau94	3.
369	B8p -> C9g	{B8, p}	{C9, g}	Des99Bea01	3.
370	aaa -> C12gg	{a, a, a}	{C12, g, g}	NACRE	1.15
371	C11p -> N12g	{C11, p}	{N12, g}	Tang03	3.
372	B10a -> N13n	{B10, a}	{N13, n}	CF88	3.
373	B11a -> C14p	{B11, a}	{C14, p}	Wan91	3.
374	C11n -> C12g	{C11, n}	{C12, g}	Rau94	3.
375	He6 -> Li6Bm	{He6}	{Li6, Bm}	Aud03	1
376	Li8 -> aaBm	{Li8}	{a, a, Bm}	Aud03	1
377	Li9 -> Be9Bm	{Li9}	{Be9, Bm}	Aud03	1
378	Li9 -> aanBm	{Li9}	{a, a, n, Bm}	Aud03	1
379	Be11 -> B11Bm	{Be11}	{B11, Bm}	Aud03	1
380	Be12 -> B12Bm	{Be12}	{B12, Bm}	Aud03	1
381	B8 -> aaBp	{B8}	{a, a, Bp}	Aud03	1
382	B12 -> C12Bm	{B12}	{C12, Bm}	Aud03	1
383	B13 -> C13Bm	{B13}	{C13, Bm}	Aud03	1
384	B14 -> C14Bm	{B14}	{C14, Bm}	Aud03	1
385	B15 -> C15Bm	{B15}	{C15, Bm}	Aud03	1
386	C9 -> aapBp	{C9}	{a, a, p, Bp}	Aud03	1
387	C10 -> B10Bp	{C10}	{B10, Bp}	Aud03	1
388	C11 -> B11Bp	{C11}	{B11, Bp}	Aud03	1
389	C15 -> N15Bm	{C15}	{N15, Bm}	Aud03	1

390	C16 -> N16Bm	{C16}	{N16, Bm}	Aud03	1
391	N12 -> C12Bp	{N12}	{C12, Bp}	Aud03	1
392	N13 -> C13Bp	{N13}	{C13, Bp}	Aud03	1
393	N16 -> O16Bm	{N16}	{O16, Bm}	Aud03	1
394	N17 -> O16nBm	{N17}	{O16, n, Bm}	Aud03	1
395	O13 -> N13Bp	{O13}	{N13, Bp}	Aud03	1
396	O14 -> N14Bp	{O14}	{N14, Bp}	Aud03	1
397	O15 -> N15Bp	{O15}	{N15, Bp}	Aud03	1
398	O19 -> F19Bm	{O19}	{F19, Bm}	Aud03	1
399	O20 -> F20Bm	{O20}	{F20, Bm}	Aud03	1
400	F17 -> O17Bp	{F17}	{O17, Bp}	Aud03	1
401	F18 -> O18Bp	{F18}	{O18, Bp}	Aud03	1
402	F20 -> Ne20Bm	{F20}	{Ne20, Bm}	Aud03	1
403	Ne18 -> F18Bp	{Ne18}	{F18, Bp}	Aud03	1
404	Ne19 -> F19Bp	{Ne19}	{F19, Bp}	Aud03	1
405	Ne23 -> Na23Bm	{Ne23}	{Na23, Bm}	Aud03	1
406	Na20 -> Ne20Bp	{Na20}	{Ne20, Bp}	Aud03	1
407	Na21 -> Ne21Bp	{Na21}	{Ne21, Bp}	Aud03	1
408	ann -> He6g	{a, n, n}	{He6, g}	Efr96	3.
409	O16n -> O17g	{O16, n}	{O17, g}	Iga95	3.
410	N14n -> C14p	{N14, n}	{C14, p}	CF88	3.
411	O14n -> N14p	{O14, n}	{N14, p}	CF88	3.
412	O14a -> Ne18g	{O14, a}	{Ne18, g}	Wie87	3.
413	C11a -> N14p	{C11, a}	{N14, p}	NACRE	2.
414	O14a -> F17p	{O14, a}	{F17, p}	Bar97C	3.
415	O17n -> C14a	{O17, n}	{C14, a}	Koe91	3.
416	F17n -> N14a	{F17, n}	{N14, a}	NACRE	1.05
417	F18n -> N15a	{F18, n}	{N15, a}	CF88	3.
418	C14d -> N15n	{C14, d}	{N15, n}	Kaw91	3.
419	ppn -> dp	{p, p, n}	{d, p}	CF88	3.
420	C14n -> C15g	{C14, n}	{C15, g}	Kaw91	3.
421	O16p -> N13a	{O16, p}	{N13, a}	CF88	3.
422	Li8p -> Be9g	{Li8, p}	{Be9, g}	TUNL&Cam08	3.
423	B11a -> N15g	{B11, a}	{N15, g}	Wan91	3.

Constants for reverse reactions

In[688]:=

```
MyGrid[Join[{"Reaction Number", "Reaction Name",
  |joins |nombre
  "Q (MeV)", "Front Factor", "T9 power", "Q/kB/10^9"}],
  |face a... |factorise
  PadLeftNumberRow[Join[{ReactionWithArrow[#[[1]]], InfoReaction[#[[2]], #[[3]]]}] & /@
  |joins
  Rest@ListReactions]]]
|reste
```

Out[688]=

Reaction Number	Reaction Name	Q (MeV)	Front Factor	T9 power	Q/kB/10^9
1	np -> dg	2.224566	4.7161402×10^9	1.5	-25.81502
2	dp -> He3g	5.4934744	1.6335102×10^{10}	1.5	-63.749132
3	dd -> He3n	3.2689084	1.7318296	0.	-37.934112
4	dd -> tp	4.0326629	1.7349209	0.	-46.797116

5	tp -> ag	19.813865	2.610575×10^{10}	1.5	-229.93039
6	td -> an	17.589299	5.5354059	0.	-204.11537
7	ta -> Li7g	2.4676205	1.1132988×10^{10}	1.5	-28.635551
8	He3n -> tp	0.7637545	1.001785	0.	-8.8630042
9	He3d -> ap	18.353053	5.5452865	0.	-212.97837
10	He3a -> Be7g	1.5871335	1.1128943×10^{10}	1.5	-18.417922
11	Be7n -> Li7p	1.6442415	1.0021491	0.	-19.080633
12	Li7p -> aa	17.346244	4.6898011	0.	-201.29484
13	Li7p -> aag	17.346244	4.6898011	0.	-201.29484
14	Be7n -> aa	18.990486	4.6998798	0.	-220.37547
15	Be7d -> aap	16.76592	$9.9655219 \times 10^{-10}$	-1.5	-194.56045
16	da -> Li6g	1.4737584	1.5305257×10^{10}	1.5	-17.102259
17	Li6p -> Be7g	5.6068495	1.1877777×10^{10}	1.5	-65.064796
18	Li6p -> He3a	4.019716	1.067287	0.	-46.646874
19	Be9t -> B11n	9.5592358	3.8284908	0.	-110.93034
20	O18n -> O19g	3.9556015	3.0716041×10^9	1.5	-45.902856
21	Li9p -> He6a	12.226855	1.8556823	0.	-141.88678
22	Li9d -> Be10n	17.411815	14.484998	0.	-202.05575
23	Be10a -> C14g	12.012513	4.776756×10^{10}	1.5	-139.39944
24	N12n -> C12p	18.120447	3.0129978	0.	-210.27908
25	Li9p -> Be9n	12.824104	1.0004549	0.	-148.81756
26	Li9a -> B12n	5.9390985	3.4308601	0.	-68.920386
27	Li9p -> Be10g	19.636381	6.8313282×10^{10}	1.5	-227.87077
28	N13n -> N14g	10.55338	1.1930595×10^{10}	1.5	-122.46691
29	B10a -> N14g	11.612108	1.1144705×10^{11}	1.5	-134.75294
30	B8a -> N12g	8.0084156	7.1824536×10^{10}	1.5	-92.933817
31	B12p -> Be9a	6.885005	0.29160469	0.	-79.897177
32	Be10p -> B11g	11.228754	4.3271611×10^9	1.5	-130.30429
33	Be10p -> Li7a	2.56444	0.10767485	0.	-29.759095
34	Be11p -> Li8a	4.095425	0.16270048	0.	-47.525441
35	Be11p -> B11n	10.727117	0.49984572	0.	-124.48304
36	B8n -> aap	18.854115	$3.6007307 \times 10^{-10}$	-1.5	-218.79295
37	B10n -> B11g	11.454219	3.0347565×10^{10}	1.5	-132.92071
38	B10a -> C13p	4.0615452	9.3625853	0.	-47.132282
39	O17n -> O18g	8.0453692	1.1010405×10^{11}	1.5	-93.362646
40	F17n -> O17p	3.54281	1.002282	0.	-41.112609
41	F18n -> O18p	2.4382621	3.0065131	0.	-28.294861
42	Be10a -> C13n	3.8360797	1.3349807	0.	-44.515863
43	Be11a -> C14n	11.510876	5.5178002	0.	-133.57818
44	N14a -> F18g	4.4152323	5.4197205×10^{10}	1.5	-51.236651
45	N15a -> F19g	4.0137985	5.5418269×10^{10}	1.5	-46.578204
46	O15a -> Ne19g	3.5284656	5.5418798×10^{10}	1.5	-40.946149
47	O16p -> F17g	0.6002693	3.0344547×10^9	1.5	-6.965837
48	O16a -> Ne20g	4.7298448	5.6527354×10^{10}	1.5	-54.887578
49	O17p -> F18g	5.6071071	3.6621842×10^{10}	1.5	-65.067785
50	O18p -> F19g	7.9935992	9.1999488×10^9	1.5	-92.76188
51	O18a -> Ne22g	9.666819	5.8490805×10^{10}	1.5	-112.17879
52	F17p -> Ne18g	3.9230706	1.0985028×10^{11}	1.5	-45.52535

53	F18p -> Ne19g	6.4100206	2.759629×10^{10}	1.5	-74.385211
54	Ne19p -> Na20g	2.1904206	7.3872203×10^9	1.5	-25.418779
55	O17p -> N14a	1.1918748	0.67571458	0.	-13.831135
56	O18p -> N15a	3.9798007	0.16600931	0.	-46.183676
57	F18p -> O15a	2.881555	0.49795902	0.	-33.439062
58	C14a -> O18g	6.2276242	5.420692×10^{10}	1.5	-72.268588
59	C14p -> N15g	10.207425	8.9988536×10^9	1.5	-118.45226
60	Be12p -> Li9a	4.986955	0.097115699	0.	-57.871218
61	Li6He3 -> aap	16.879295	$7.2462507 \times 10^{-10}$	-1.5	-195.87611
62	Li6t -> Be9g	17.688239	4.2265088×10^{10}	1.5	-205.26352
63	Li6t -> aan	16.115541	$7.2333393 \times 10^{-10}$	-1.5	-187.01311
64	Li6t -> Li8p	0.8019182	2.0175581	0.	-9.3058756
65	Li7d -> Be9g	16.694377	5.8104618×10^{10}	1.5	-193.73023
66	Li7He3 -> B10g	17.787714	3.4631845×10^{10}	1.5	-206.41788
67	Li7He3 -> Li6a	13.326528	2.1970664	0.	-154.64796
68	Li7t -> Be10g	17.249425	2.4244984×10^{11}	1.5	-200.17129
69	Li8a -> B12g	10.001316	7.1839155×10^{10}	1.5	-116.06046
70	Li8a -> B11n	6.6316915	3.0721835	0.	-76.957595
71	Li8d -> Be10g	21.474032	3.0330295×10^{11}	1.5	-249.19583
72	Li8He3 -> B11g	27.209311	8.0344813×10^{10}	1.5	-315.75099
73	Li8He3 -> B10n	15.755092	2.6474879	0.	-182.83027
74	Li8He3 -> Be10p	15.980557	18.567558	0.	-185.44669
75	Li8He3 -> Li7a	18.544997	1.999259	0.	-215.20579
76	Li8t -> Be11g	15.71844	1.6045282×10^{11}	1.5	-182.40495
77	Li8t -> Be10n	15.216803	18.534474	0.	-176.58369
78	Li9a -> B13g	10.817916	4.5613941×10^{10}	1.5	-125.53672
79	Li9d -> Be11g	17.913452	1.2539653×10^{11}	1.5	-207.877
80	Li9He3 -> B12g	26.516718	8.972505×10^{10}	1.5	-307.71378
81	Li9He3 -> B11n	23.147094	3.8370693	0.	-268.61091
82	Li9He3 -> Be11p	12.419977	7.6765072	0.	-144.12787
83	Li9He3 -> Li8a	16.515402	1.2489714	0.	-191.65331
84	Li9t -> Be12g	14.82691	2.6881082×10^{11}	1.5	-172.05917
85	Li9t -> Be11n	11.656223	7.6628292	0.	-135.26487
86	Be7He3 -> C10g	15.001548	2.4232015×10^{11}	1.5	-174.08576
87	Be7t -> B10g	18.668201	3.4644432×10^{10}	1.5	-216.63551
88	Be7t -> Be9p	12.081389	3.5583331	0.	-140.19872
89	Be7t -> Li6a	14.207015	2.197865	0.	-164.86559
90	Be9a -> C13g	10.648357	9.1155448×10^{10}	1.5	-123.56907
91	Be9d -> B11g	15.816465	6.265042×10^{10}	1.5	-183.54248
92	Be9d -> B10n	4.3622456	2.0644299	0.	-50.621765
93	Be9d -> Be10p	4.5877111	14.478412	0.	-53.238184
94	Be9d -> Li7a	7.1521511	1.5589608	0.	-82.997279
95	Be9He3 -> C12g	26.279668	2.6899782×10^{11}	1.5	-304.96293
96	Be9He3 -> C11n	7.5589508	3.8265847	0.	-87.717994
97	Be9He3 -> B11p	10.32299	3.8353246	0.	-119.79335
98	Be9He3 -> aaa	19.004921	1.342731×10^{-9}	-1.5	-220.54298
99	Be9t -> B12g	12.92886	8.9524453×10^{10}	1.5	-150.03321
100	Be9t -> Li8a	2.9275443	1.2461791	0.	-33.972746

101	Be10d -> B12g	12.373812	2.1455094×10^{10}	1.5	-143.59214
102	Be10d -> B11n	9.0041876	0.91752172	0.	-104.48927
103	Be10d -> Li8a	2.3724961	0.2986546	0.	-27.531678
104	Be10He3 -> C13g	24.413699	3.4912881×10^{10}	1.5	-283.30925
105	Be10He3 -> C12n	19.467391	3.9395002	0.	-225.90972
106	Be10He3 -> B12p	6.8803373	1.3134349	0.	-79.84301
107	Be10He3 -> Be9a	13.765342	0.38300378	0.	-159.74019
108	Be10t -> B13g	10.9954	1.7431254×10^{10}	1.5	-127.59634
109	Be10t -> B12n	6.1165828	1.3110946	0.	-70.980006
110	Be10t -> Li9a	0.1774843	0.38214751	0.	-2.0596201
111	Be11a -> C15g	12.728986	4.9701689×10^{10}	1.5	-147.71377
112	Be11d -> B13g	16.750992	3.295019×10^{10}	1.5	-194.38721
113	Be11d -> B12n	11.872175	2.478354	0.	-137.77089
114	Be11d -> Be12p	0.9461211	7.4382519	0.	-10.979281
115	Be11d -> Li9a	5.9330761	0.72237104	0.	-68.850499
116	Be11He3 -> C14g	32.088495	1.4430343×10^{11}	1.5	-372.37157
117	Be11He3 -> C13n	23.912062	4.0329107	0.	-277.488
118	Be11He3 -> B13p	11.257517	2.0171401	0.	-130.63808
119	Be11He3 -> Be10a	20.075982	3.0209505	0.	-232.97213
120	Be11p -> B12g	14.096741	1.1688265×10^{10}	1.5	-163.58591
121	Be11t -> B14g	11.46298	2.8798746×10^{10}	1.5	-133.02238
122	Be11t -> B13n	10.493763	2.0135459	0.	-121.77508
123	Be12a -> C16g	13.808716	5.1410638×10^{10}	1.5	-160.24351
124	Be12a -> C15n	9.5582985	1.4168163	0.	-110.91946
125	Be12d -> B14g	14.549522	1.3434217×10^{10}	1.5	-168.84021
126	Be12d -> B13n	13.580305	0.93929136	0.	-157.59291
127	Be12He3 -> C15g	30.135918	3.7053075×10^{10}	1.5	-349.71286
128	Be12He3 -> C14n	28.917808	4.1135717	0.	-335.57727
129	Be12He3 -> B14p	9.0560473	0.82241401	0.	-105.09108
130	Be12He3 -> Be11a	17.406932	0.74550937	0.	-201.99909
131	Be12p -> B13g	15.804871	4.4298297×10^9	1.5	-183.40793
132	Be12p -> B12n	10.926053	0.33319038	0.	-126.7916
133	Be12t -> B15g	11.06961	1.849288×10^{10}	1.5	-128.45751
134	Be12t -> B14n	8.2922928	0.82094863	0.	-96.228076
135	B8a -> C11p	7.408145	3.078465	0.	-85.967965
136	B8d -> C10g	20.358652	3.032601×10^{11}	1.5	-236.25238
137	B8He3 -> C10p	14.865177	18.564934	0.	-172.50324
138	B8t -> C11g	27.22201	8.0365638×10^{10}	1.5	-315.89835
139	B8t -> C10n	14.101423	18.531855	0.	-163.64024
140	B8t -> B10p	18.53183	2.6542225	0.	-215.05299
141	B8t -> Be7a	19.677494	2.0000464	0.	-228.34787
142	B10d -> C12g	25.186331	4.5131916×10^{11}	1.5	-292.27527
143	B10d -> C11n	6.4656136	6.4201674	0.	-75.030341
144	B10d -> B11p	9.2296531	6.434831	0.	-107.10569
145	B10d -> aaa	17.911584	2.2528072×10^{-9}	-1.5	-207.85533
146	B10He3 -> N13g	21.636347	2.4429646×10^{11}	1.5	-251.07942
147	B10He3 -> N12n	1.5724098	9.1698685	0.	-18.247061
148	B10He3 -> C12p	19.692856	27.628793	0.	-228.52614
149	B10n -> Be10p	0.2254655	7.0132737	0.	-2.6164189

150	B10t -> C13g	23.87541	2.4441731×10^{11}	1.5	-277.06267
151	B10t -> C12n	18.929102	27.579564	0.	-219.66314
152	B10t -> B12p	6.3420483	9.1950656	0.	-73.596425
153	B10t -> Be9a	13.227053	2.6813243	0.	-153.4936
154	B11d -> C13g	18.67842	1.3179669×10^{11}	1.5	-216.75409
155	B11d -> C12n	13.732112	14.871676	0.	-159.35456
156	B11d -> B12p	1.1450581	4.9582379	0.	-13.287849
157	B11d -> Be9a	8.0300631	1.4458454	0.	-93.185026
158	B11He3 -> N14g	20.735508	9.6040724×10^{10}	1.5	-240.62562
159	B11He3 -> N13n	10.182128	8.0499526	0.	-118.1587
160	B11He3 -> C13p	13.184945	8.068311	0.	-153.00496
161	B11He3 -> B10a	9.1234003	0.86176102	0.	-105.87268
162	B11t -> C14g	20.597624	2.8818155×10^{11}	1.5	-239.02553
163	B11t -> C13n	12.421191	8.0539349	0.	-144.14196
164	B11t -> Be10a	8.5851113	6.0329971	0.	-99.626094
165	B12a -> N16g	10.110416	3.0822331×10^{10}	1.5	-117.32652
166	B12p -> C12n	12.587054	2.9993874	0.	-146.06671
167	B12a -> N15n	7.6215598	4.2481819	0.	-88.444542
168	B12d -> C14g	23.485229	2.0167334×10^{11}	1.5	-272.5348
169	B12d -> C13n	15.308796	5.6362523	0.	-177.65122
170	B12d -> B13p	2.6542511	2.8190831	0.	-30.801309
171	B12d -> Be10a	11.472716	4.2219728	0.	-133.13536
172	B12He3 -> N15g	28.199179	1.1109994×10^{11}	1.5	-327.23793
173	B12He3 -> N14n	17.365884	4.1071575	0.	-201.52275
174	B12He3 -> C14p	17.991754	12.34601	0.	-208.78567
175	B12He3 -> B11a	17.207995	1.1183986	0.	-199.69052
176	B12n -> B13g	4.8788171	1.3295191×10^{10}	1.5	-56.61633
177	B12p -> C13g	17.533362	2.6581356×10^{10}	1.5	-203.46624
178	B12t -> C15g	18.44611	1.1100877×10^{11}	1.5	-214.05825
179	B12t -> C14n	17.228	12.324012	0.	-199.92266
180	B12t -> Be11a	5.7171243	2.2335009	0.	-66.344482
181	C9a -> O13g	8.2209156	4.5605333×10^{10}	1.5	-95.399778
182	C9d -> C10p	19.059081	14.516402	0.	-221.17148
183	C9n -> C10g	21.283647	6.8461389×10^{10}	1.5	-246.9865
184	C9t -> N12g	26.52271	8.9753778×10^{10}	1.5	-307.78331
185	C9t -> C11p	25.922439	3.8469286	0.	-300.81746
186	C9t -> B8a	18.514294	1.2496256	0.	-214.84949
187	C11d -> N13g	18.439642	1.3179714×10^{11}	1.5	-213.98319
188	C11d -> C12p	16.496151	14.905643	0.	-191.42991
189	C11He3 -> O14g	17.572837	2.8803064×10^{11}	1.5	-203.92433
190	C11He3 -> N13p	12.946167	8.0683386	0.	-150.23405
191	C11He3 -> C10a	7.4570323	6.0305815	0.	-86.535279
192	C11t -> N14g	22.735793	9.6088563×10^{10}	1.5	-263.83796
193	C11t -> N13n	12.182413	8.0539624	0.	-141.37105
194	C11t -> C13p	15.18523	8.0723299	0.	-176.21731
195	C11t -> B10a	11.123685	0.86219027	0.	-129.08503
196	C12a -> O16g	7.1619169	5.1331446×10^{10}	1.5	-83.110606
197	C12d -> N14g	10.272305	2.236818×10^{10}	1.5	-119.20517
198	C12d -> C13p	2.7217423	1.8791345	0.	-31.584512

199	C12He3 -> O15g	12.075618	3.6955518×10^{10}	1.5	-140.13175
200	C12He3 -> N14p	4.7788306	1.3693321	0.	-55.456034
201	C11a -> O15g	10.218716	9.933585×10^{10}	1.5	-118.58329
202	C12He3 -> C11a	1.8569023	0.37202599	0.	-21.54846
203	C12n -> C13g	4.9463083	8.8622617×10^9	1.5	-57.399533
204	C12p -> N13g	1.9434906	8.8420967×10^9	1.5	-22.553275
205	C12t -> N15g	14.848371	3.6974876×10^{10}	1.5	-172.30822
206	C12t -> N14n	4.0150761	1.3668922	0.	-46.59303
207	C12t -> C14p	4.6409463	4.1088429	0.	-53.855953
208	C12t -> B11a	3.8571873	0.3722113	0.	-44.760806
209	C13a -> O17g	6.3586879	1.7616089×10^{10}	1.5	-73.78952
210	C13d -> N15g	16.159292	6.8274501×10^{10}	1.5	-187.52082
211	C13d -> N14n	5.3259967	2.523981	0.	-61.805634
212	C13d -> C14p	5.9518669	7.5870221	0.	-69.068557
213	C13d -> B11a	5.1681079	0.68729211	0.	-59.97341
214	C13He3 -> O16g	22.793228	1.5147802×10^{11}	1.5	-264.50447
215	C13He3 -> O15n	7.1293096	4.1699872	0.	-82.732214
216	C13He3 -> N15p	10.665817	4.1796188	0.	-123.77169
217	C13He3 -> C12a	15.631311	2.9509791	0.	-181.39386
218	C13n -> C14g	8.1764329	3.578146×10^{10}	1.5	-94.883578
219	C13p -> N14g	7.5505627	1.1903448×10^{10}	1.5	-87.620654
220	C13t -> N16g	12.390919	3.0270843×10^{10}	1.5	-143.79066
221	C13t -> N15n	9.9020629	4.1721715	0.	-114.90869
222	C13t -> C15p	0.9127481	4.176189	0.	-10.592003
223	C13t -> B12a	2.2805031	0.98210754	0.	-26.464143
224	C14d -> N16g	10.471715	1.384404×10^{10}	1.5	-121.51922
225	C14d -> B12a	0.3612991	0.44915617	0.	-4.1927026
226	C14He3 -> O17g	18.759874	1.2875442×10^{10}	1.5	-217.69933
227	C14He3 -> O16n	14.616795	4.2334221	0.	-169.62089
228	C14He3 -> N16p	4.9782403	0.84750249	0.	-57.770088
229	C14He3 -> C13a	12.401186	0.73089104	0.	-143.90981
230	C14t -> N17g	10.099703	3.8603633×10^{10}	1.5	-117.2022
231	C14t -> N16n	4.2144858	0.84599241	0.	-48.907084
232	C15a -> O19g	8.9651156	1.8484801×10^{10}	1.5	-104.03586
233	C15a -> O18n	5.0095141	6.0179635	0.	-58.133006
234	C15n -> C16g	4.2504171	3.6286028×10^{10}	1.5	-49.324049
235	C15p -> N16g	11.478171	7.248437×10^9	1.5	-133.19866
236	C15p -> N15n	8.9893148	0.99903799	0.	-104.31668
237	C15p -> B12a	1.367755	0.23516836	0.	-15.87214
238	N12a -> O15p	9.618445	4.2576249	0.	-111.61744
239	N12n -> N13g	20.063937	2.6641218×10^{10}	1.5	-232.83236
240	N12p -> O13g	1.5120706	1.326475×10^{10}	1.5	-17.546853
241	N13a -> F17g	5.8186956	1.7616065×10^{10}	1.5	-67.523168
242	N13n -> C13p	3.0028177	1.0022806	0.	-34.846257
243	N13p -> O14g	4.6266696	3.5698879×10^{10}	1.5	-53.690279
244	N14n -> N15g	10.833295	2.7050322×10^{10}	1.5	-125.71519
245	N14p -> O15g	7.2967873	2.6987987×10^{10}	1.5	-84.675713
246	N15n -> N16g	2.4888558	7.2554168×10^9	1.5	-28.881976

247	N15p -> O16g	12.127411	3.6242067×10^{10}	1.5	-140.73278
248	O14n -> O15g	13.223498	9.0194076×10^9	1.5	-153.45235
249	O14n -> C11a	3.0047825	0.090797105	0.	-34.869058
250	O15n -> O16g	15.663918	3.6325777×10^{10}	1.5	-181.77225
251	O15n -> N15p	3.5365078	1.0023097	0.	-41.039475
252	O15n -> C12a	8.5020015	0.70767101	0.	-98.661645
253	O17a -> Ne21g	7.3479321	8.6333599×10^{10}	1.5	-85.269224
254	O17a -> Ne20n	0.5867655	18.586092	0.	-6.8091319
255	O19a -> Ne23g	10.911866	5.9348191×10^{10}	1.5	-126.62696
256	O19a -> Ne22n	5.7112175	19.04243	0.	-66.275937
257	O19n -> O20g	7.6080171	1.110756×10^{11}	1.5	-88.287385
258	O19p -> F20g	10.639334	2.2176988×10^{10}	1.5	-123.46436
259	O19p -> F19n	4.0379977	2.995161	0.	-46.859024
260	O19p -> N16a	2.513055	0.39212956	0.	-29.162797
261	F17a -> Na21g	6.5612456	8.6331889×10^{10}	1.5	-76.140104
262	F17a -> Ne20p	4.1295755	18.628505	0.	-47.921741
263	F17n -> F18g	9.1499171	3.6705411×10^{10}	1.5	-106.18039
264	F18a -> Na22g	8.4795256	2.5065774×10^{10}	1.5	-98.400823
265	F18a -> Ne21p	1.740825	2.3574347	0.	-20.201438
266	F18n -> F19g	10.431861	2.7659767×10^{10}	1.5	-121.05674
267	F19a -> Na23g	10.467324	2.9670844×10^{10}	1.5	-121.46827
268	F19a -> Ne22p	1.6732198	6.3577315	0.	-19.416912
269	F19n -> F20g	6.6013359	7.4042723×10^9	1.5	-76.605333
270	F19p -> Ne20g	12.843457	3.6967378×10^{10}	1.5	-149.04215
271	F19p -> O16a	8.1136121	0.65397327	0.	-94.154572
272	B8n -> Li6He3	1.9748203	0.49690948	0.	-22.916841
273	Li9p -> Li7t	2.3869557	0.28176254	0.	-27.699475
274	B8n -> Be7d	2.0881954	0.36131883	0.	-24.232505
275	C9n -> Be7He3	6.2820992	0.28252455	0.	-72.900745
276	B10n -> aat	0.322285	$1.3566047 \times 10^{-10}$	-1.5	-3.7399627
277	Be10p -> aat	0.0968195	$1.9343387 \times 10^{-11}$	-1.5	-1.1235438
278	Be11p -> Be9t	1.1678807	0.13055947	0.	-13.552695
279	Be11p -> Be10d	1.7229289	0.54477808	0.	-19.993763
280	Be12p -> Be10t	4.8094707	0.25413145	0.	-55.811598
281	C9n -> B8d	0.9249954	0.22575139	0.	-10.734127
282	N13n -> C12d	0.2810754	0.53337351	0.	-3.261745
283	B10a -> C12d	1.3398029	4.9823924	0.	-15.547769
284	O14n -> C12He3	1.1478802	0.24406119	0.	-13.320599
285	C15p -> C14d	1.0064559	0.52357817	0.	-11.679437
286	Ne18n -> O15a	8.1084015	0.16638821	0.	-94.094106
287	Ne19n -> O16a	12.135453	0.65547753	0.	-140.8261
288	Na20n -> F17a	10.545302	0.26925106	0.	-122.37316
289	Ne18n -> F18p	5.2268465	0.33414036	0.	-60.655044
290	Ne19n -> F19p	4.0218407	1.0023002	0.	-46.67153
291	Li7He3 -> Be9p	11.200902	3.5570403	0.	-129.98109
292	Li6t -> Li7d	0.9938621	0.72739637	0.	-11.533292
293	Li6He3 -> Be7d	0.1133751	0.72713209	0.	-1.3156636
294	Li7He3 -> aad	11.85277	$2.8709958 \times 10^{-10}$	-1.5	-137.5457
295	Li8He3 -> Be9d	11.392846	1.2824306	0.	-132.20851

296	Li8He3 -> aat	16.077377	$3.5915945 \times 10^{-10}$	-1.5	-186.57024
297	Li9d -> Li8t	2.1950118	0.78151655	0.	-25.472058
298	Li9He3 -> Be10d	14.142906	4.1819929	0.	-164.12164
299	Li9He3 -> Be9t	13.587858	1.0022407	0.	-157.68057
300	Be7t -> aad	12.733257	$2.8720393 \times 10^{-10}$	-1.5	-147.76333
301	Be7t -> Li7He3	0.880487	1.0003635	0.	-10.217629
302	Be9d -> aat	4.6845306	$2.8006152 \times 10^{-10}$	-1.5	-54.361728
303	Be9t -> Be10d	0.5550482	4.1726432	0.	-6.4410678
304	Be9He3 -> B10d	1.0933372	0.59602569	0.	-12.687653
305	Be10He3 -> B11d	5.7352792	0.26489954	0.	-66.555161
306	Be10He3 -> B10t	0.538289	0.14284128	0.	-6.2465854
307	B8d -> Be7He3	5.3571038	1.2514853	0.	-62.166617
308	B8t -> aaHe3	18.090361	$3.5943149 \times 10^{-10}$	-1.5	-209.92995
309	B10p -> aaHe3	-0.4414695	$1.3541875 \times 10^{-10}$	-1.5	5.1230416
310	B10t -> B11d	5.1969902	1.8545027	0.	-60.308576
311	B10He3 -> C11d	3.1967052	1.8535794	0.	-37.096229
312	B11t -> B13p	-0.2333537	4.0283348	0.	2.7079576
313	B11He3 -> C12d	10.463203	4.2936315	0.	-121.42045
314	N12n -> C11d	1.6242954	0.20213806	0.	-18.849168
315	C11t -> C12d	12.463488	4.2957702	0.	-144.6328
316	C11t -> B11He3	2.000285	1.0004981	0.	-23.212347
317	Be7He3 -> ppaa	11.272446	$1.2201358 \times 10^{-19}$	-3.	-130.81132
318	dd -> ag	23.846528	4.5291411×10^{10}	1.5	-276.7275
319	He3He3 -> app	12.859579	$3.3947057 \times 10^{-10}$	-1.5	-149.22924
320	Li7a -> B11g	8.6643136	4.0187296×10^{10}	1.5	-100.5452
321	Be7p -> B8g	0.1363706	1.3052572×10^{10}	1.5	-1.5825153
322	Be7a -> C11g	7.5445156	4.0181887×10^{10}	1.5	-87.550481
323	Be9p -> B10g	6.5868116	9.7361407×10^9	1.5	-76.436786
324	Be9p -> aad	0.6518677	$8.0713053 \times 10^{-11}$	-1.5	-7.5646116
325	Be9p -> Li6a	2.1256261	0.61766701	0.	-24.66687
326	Be9a -> C12n	5.7020485	10.2858	0.	-66.169535
327	B10p -> C11g	8.6901796	3.027841×10^{10}	1.5	-100.84536
328	B10p -> Be7a	1.145664	0.75353379	0.	-13.294881
329	B11p -> C12g	15.956678	7.013691×10^{10}	1.5	-185.16958
330	B11p -> aaa	8.6819308	$3.5009579 \times 10^{-10}$	-1.5	-100.74964
331	B11a -> N14n	0.1578888	3.6723556	0.	-1.8322237
332	C13a -> O16n	2.2156086	5.7921384	0.	-25.711074
333	N15p -> C12a	4.9654937	0.70604024	0.	-57.62217
334	Li7t -> Be9n	10.437148	3.5507024	0.	-121.11809
335	B11n -> B12g	3.3696241	2.3383745×10^{10}	1.5	-39.10287
336	C11n -> aaa	11.44597	$3.5089541 \times 10^{-10}$	-1.5	-132.82499
337	Li7d -> aan	15.121678	9.944151×10^{-10}	-1.5	-175.47982
338	dn -> tg	6.2572289	1.636426×10^{10}	1.5	-72.612137
339	tt -> ann	11.33207	$3.3826191 \times 10^{-10}$	-1.5	-131.50323
340	He3n -> ag	20.577619	2.6152349×10^{10}	1.5	-238.79339
341	He3t -> ad	14.32039	1.5981381	0.	-166.18126
342	He3t -> anp	12.095825	3.388657×10^{-10}	-1.5	-140.36623
343	aan -> Be9g	1.5726983	5.8430948×10^{19}	3.	-18.250409

344	Li7t -> aann	8.8644495	$1.2153499 \times 10^{-19}$	-3.	-102.86768
345	Li7He3 -> aanp	9.628204	$6.0875964 \times 10^{-20}$	-3.	-111.73068
346	Li8d -> Li9p	1.8376511	4.4398826	0.	-21.325059
347	Li8d -> Li7t	4.2246068	1.2509926	0.	-49.024533
348	Be7t -> aanp	10.508691	6.089809×10^{-20}	-3.	-121.94831
349	Be7He3 -> aapp	11.272446	$1.2201358 \times 10^{-19}$	-3.	-130.81132
350	C9a -> N12p	6.708845	3.4380846	0.	-77.852925
351	Li6n -> ta	4.7834705	1.0691921	0.	-55.509878
352	He3t -> Li6g	15.794149	2.4459915×10^{10}	1.5	-183.28351
353	anp -> Li6g	3.6983244	7.2181738×10^{19}	3.	-42.917279
354	Li6n -> Li7g	7.251091	1.1903303×10^{10}	1.5	-84.145429
355	Li6d -> Li7p	5.026525	2.5239503	0.	-58.330409
356	Li6d -> Be7n	3.3822835	2.5185377	0.	-39.249776
357	Li6a -> B10g	4.4611855	1.5762766×10^{10}	1.5	-51.769915
358	Li7a -> B10n	-2.7899055	1.3242346	0.	32.375514
359	Li7n -> Li8g	2.0326221	1.3081021×10^{10}	1.5	-23.587603
360	Li7d -> Li8p	-0.1919439	2.7736709	0.	2.2274168
361	Li8n -> Li9g	4.0622171	2.0939109×10^{10}	1.5	-47.140079
362	Li8p -> aan	15.313622	3.585195×10^{-10}	-1.5	-177.70723
363	Li8d -> Be9n	14.661755	4.4419024	0.	-170.14262
364	Be9n -> Be10g	6.8122771	6.828222×10^{10}	1.5	-79.053205
365	Be9p -> aapn	-1.5726983	$1.7114218 \times 10^{-20}$	-3.	18.250409
366	B11p -> C11n	-2.7640395	0.99772122	0.	32.075351
367	Be10n -> Be11g	0.5016371	8.6569933×10^9	1.5	-5.8212577
368	Be11n -> Be12g	3.1706871	3.5079839×10^{10}	1.5	-36.794301
369	B8p -> C9g	1.2995706	2.0890858×10^{10}	1.5	-15.080893
370	aaa -> C12gg	7.2747468	2.0033634×10^{20}	3.	-84.419943
371	C11p -> N12g	0.6002706	2.3331283×10^{10}	1.5	-6.9658521
372	B10a -> N13n	1.0587275	9.3412819	0.	-12.286024
373	B11a -> C14p	0.783759	11.039007	0.	-9.0951468
374	C11n -> C12g	18.720717	7.0297102×10^{10}	1.5	-217.24493
375	He6 -> Li6Bm	2.7228746	0.33369189	0.	-31.597652
376	Li8 -> aaBm	15.313622	$3.5926105 \times 10^{-10}$	-1.5	-177.70723
377	Li9 -> Be9Bm	12.824104	1.0025242	0.	-148.81756
378	Li9 -> aanBm	11.251405	$1.7157418 \times 10^{-20}$	-3.	-130.56715
379	Be11 -> B11Bm	10.727117	0.50087959	0.	-124.48304
380	Be12 -> B12Bm	10.926053	0.33387954	0.	-126.7916
381	B8 -> aaBp	18.854115	$3.5932985 \times 10^{-10}$	-1.5	-218.79295
382	B12 -> C12Bm	12.587054	3.0055912	0.	-146.06671
383	B13 -> C13Bm	12.654545	2.0034564	0.	-146.84991
384	B14 -> C14Bm	19.861761	5.0121716	0.	-230.48619
385	B15 -> C15Bm	18.302553	2.0042068	0.	-212.39234
386	C9 -> aapBp	17.554545	1.720034×10^{-20}	-3.	-203.71206
387	C10 -> B10Bp	4.4304075	0.14292924	0.	-51.412752
388	C11 -> B11Bp	2.7640395	1.0002152	0.	-32.075351
389	C15 -> N15Bm	8.9893148	1.0011044	0.	-104.31668
390	C16 -> N16Bm	7.2277535	0.20017152	0.	-83.874609
391	N12 -> C12Bp	18.120447	3.0067786	0.	-210.27908

392	N13 -> C13Bp	3.0028177	1.0002118	0.	-34.846257
393	N16 -> O16Bm	9.6385548	5.0055055	0.	-111.8508
394	N17 -> O16nBm	3.7533377	$1.0969485 \times 10^{-10}$	-1.5	-43.555682
395	O13 -> N13Bp	18.551867	2.0042765	0.	-215.2855
396	O14 -> N14Bp	5.9267108	0.33351101	0.	-68.776633
397	O15 -> N15Bp	3.5365078	1.0002409	0.	-41.039475
398	O19 -> F19Bm	4.0379977	3.0013561	0.	-46.859024
399	O20 -> F20Bm	3.0313165	0.20006966	0.	-35.176972
400	F17 -> O17Bp	3.54281	1.0002132	0.	-41.112609
401	F18 -> O18Bp	2.4382621	3.0003074	0.	-28.294861
402	F20 -> Ne20Bm	6.242121	5.0030359	0.	-72.436817
403	Ne18 -> F18Bp	5.2268465	0.33345067	0.	-60.655044
404	Ne19 -> F19Bp	4.0218407	1.0002313	0.	-46.67153
405	Ne23 -> Na23Bm	3.593456	1.5005136	0.	-41.700331
406	Na20 -> Ne20Bp	14.674877	5.0053916	0.	-170.2949
407	Na21 -> Ne21Bp	4.3294965	1.000233	0.	-50.241728
408	ann -> He6g	0.9754498	1.0837997×10^{20}	3.	-11.319627
409	O16n -> O17g	4.1430793	3.0413792×10^9	1.5	-48.078446
410	N14n -> C14p	0.6258702	3.0059743	0.	-7.2629231
411	O14n -> N14p	5.9267108	0.33420083	0.	-68.776633
412	O14a -> Ne18g	5.1150966	5.4207012×10^{10}	1.5	-59.35824
413	C11a -> N14p	2.9219283	3.6807432	0.	-33.907575
414	O14a -> F17p	1.192026	0.49346269	0.	-13.832889
415	O17n -> C14a	1.817745	2.0311806	0.	-21.094058
416	F17n -> N14a	4.7346848	0.67725653	0.	-54.943744
417	F18n -> N15a	6.4180628	0.49910918	0.	-74.478537
418	C14d -> N15n	7.9828589	1.9080971	0.	-92.637244
419	ppn -> dp	2.224566	2.3580701×10^9	1.5	-25.81502
420	C14n -> C15g	1.2181101	9.0075189×10^9	1.5	-14.135583
421	O16p -> N13a	-5.2184263	0.17225497	0.	60.557331
422	Li8p -> Be9g	16.886321	2.0948634×10^{10}	1.5	-195.95764
423	B11a -> N15g	10.991184	9.9338403×10^{10}	1.5	-127.54741

List of reactions

In[689]:=

```
ListReactionsNames = ListReactionsUpToChosenMass[All, 1];
```

In[690]:=

```
ListNuclearReactionsNames = Select[ListReactionsNames, # != "nT0p" &];
```

We can define an association (a dictionary) between reaction names and number

In[691]:=

```

KeyNuclearReaction =
  Association[# → Position[ListNuclearReactionsNames, #] [[1, 1]] & /@
    ListNuclearReactionsNames];
KeyReaction = Association[
  # → Position[ListReactionsNames, #] [[1, 1]] & /@ ListReactionsNames];

```

Let us collect all the species involved in the reactions. These are the species whose abundance we are going to solve numerically.

In[693]:=

```

VariablesInEquations =
  Union@Flatten[Join[RemoveNonNuclear[#[[2]]], RemoveNonNuclear[#[[3]]]] & /@
    ListReactionsUpToChosenMass]

```

Out[693]=

```

{a, B10, B11, B12, B13, B14, B15, B8, Be10, Be11, Be12, Be7, Be9, C10, C11,
 C12, C13, C14, C15, C16, C9, d, F17, F18, F19, F20, He3, He6, Li6, Li7,
 Li8, Li9, n, N12, N13, N14, N15, N16, N17, Na20, Na21, Na22, Na23, Ne18,
 Ne19, Ne20, Ne21, Ne22, Ne23, O13, O14, O15, O16, O17, O18, O19, O20, p, t}

```

We can check the number of species (59 for the full network of 423 equations).

In[694]:=

```

NumberVariable = Length@VariablesInEquations

```

Out[694]=

59

List of chemical species names used

Let us collect all the species used together with their weights in terms of neutrons and protons

In[695]:=

```
NamesWithWeights =
  Select[NamesWithWeightsAll, MemberQ[VariablesInEquations, #[[1]]] &]
  [sélectionne] [appartient ?]
```

Out[695]=

```
{ {n, {1, 0}}, {p, {0, 1}}, {d, {1, 1}}, {t, {2, 1}}, {He3, {1, 2}}, {a, {2, 2}},
  {He6, {4, 2}}, {Li6, {3, 3}}, {Li7, {4, 3}}, {Li8, {5, 3}}, {Li9, {6, 3}},
  {Be7, {3, 4}}, {Be9, {5, 4}}, {Be10, {6, 4}}, {Be11, {7, 4}}, {Be12, {8, 4}},
  {B8, {3, 5}}, {B10, {5, 5}}, {B11, {6, 5}}, {B12, {7, 5}}, {B13, {8, 5}},
  {B14, {9, 5}}, {B15, {10, 5}}, {C9, {3, 6}}, {C10, {4, 6}}, {C11, {5, 6}},
  {C12, {6, 6}}, {C13, {7, 6}}, {C14, {8, 6}}, {C15, {9, 6}}, {C16, {10, 6}},
  {N12, {5, 7}}, {N13, {6, 7}}, {N14, {7, 7}}, {N15, {8, 7}}, {N16, {9, 7}},
  {N17, {10, 7}}, {O13, {5, 8}}, {O14, {6, 8}}, {O15, {7, 8}}, {O16, {8, 8}},
  {O17, {9, 8}}, {O18, {10, 8}}, {O19, {11, 8}}, {O20, {12, 8}}, {F17, {8, 9}},
  {F18, {9, 9}}, {F19, {10, 9}}, {F20, {11, 9}}, {Ne18, {8, 10}}, {Ne19, {9, 10}},
  {Ne20, {10, 10}}, {Ne21, {11, 10}}, {Ne22, {12, 10}}, {Ne23, {13, 10}},
  {Na20, {9, 11}}, {Na21, {10, 11}}, {Na22, {11, 11}}, {Na23, {12, 11}} }
```

WeightsNuclear is the list of nuclear weights for the variables (their A in nuclear physics notation). It is obtained by summing the (n,p) numbers

In[696]:=

```
WeightsNuclear = (Plus@@ (#[[2]])) & /@ NamesWithWeights
  [plus]
```

Out[696]=

```
{1, 1, 2, 3, 3, 4, 6, 6, 7, 8, 9, 7, 9, 10, 11, 12, 8, 10, 11, 12, 13, 14,
  15, 9, 10, 11, 12, 13, 14, 15, 16, 12, 13, 14, 15, 16, 17, 13, 14, 15, 16,
  17, 18, 19, 20, 17, 18, 19, 20, 18, 19, 20, 21, 22, 23, 20, 21, 22, 23}
```

We build a function which selects all species having the same mass number A or Z number

In[697]:=

```
NamesMassNumber[A_] :=
  Select[NamesWithWeights, ((Plus@@ (#[[2]])) == A) &] [[All, 1]]
  [sélectionne] [plus] [tout]
NamesAtomicNumber[Z_] := Select[NamesWithWeights, (#[[2, 2]] == Z) &] [[All, 1]]
  [sélectionne] [tout]
```

Shorthand notation for abundances of species

We use two ways of noting the abundance of a species. One notation is Yn_ip_j where i is the number of neutrons and j the number of protons. For instance Yn₂p₂ is He₄.

But we also want to use short names. For instance He₄ for Helium₄. So we want to relate automatically YHe₄ to Yn₂p₂.

When the function ShortString is evaluated, it defines the relation between the Yshortname and the Yn_ip_j notation

In[699]:=

```
StackY[name_] := "Y" <> ToString[name];
               | en chaîne de caractères
```

In[700]:=

```
YName[PostString_][n_, p_] :=
  ToExpression@StackY["n" <> ToString[n] <> "p" <> ToString[p] <> PostString];
  | en expression          | en chaîne de caractères   | en chaîne de caractères
ShortString[nameshort_, np_List] :=
  | liste
  (Evaluate@ToExpression["Y" <> nameshort] := YName[""] @@ np;);
  | évalue          | en expression
```

In[702]:=

```
ShortString@@@NamesWithWeights;
```

Some examples to see how these short names have been related ot the correct names

In[703]:=

```
Ya
Yp
Yt
YLi7
```

Out[703]=

```
Yn2p2
```

Out[704]=

```
Yn0p1
```

Out[705]=

```
Yn2p1
```

Out[706]=

```
Yn4p3
```

The list of all shortnames available is ShortNames, and we associate a number through a dictionary KeyVal

In[707]:=

```
ShortNames = NamesWithWeights[All, 1]
                                   |tout
KeyVal = Association[# → Position[ShortNames, #] [1, 1] & /@ ShortNames]
               |association      |position
```

Out[707]=

```
{n, p, d, t, He3, a, He6, Li6, Li7, Li8, Li9, Be7, Be9, Be10, Be11, Be12,
 B8, B10, B11, B12, B13, B14, B15, C9, C10, C11, C12, C13, C14, C15, C16,
 N12, N13, N14, N15, N16, N17, O13, O14, O15, O16, O17, O18, O19, O20, F17,
 F18, F19, F20, Ne18, Ne19, Ne20, Ne21, Ne22, Ne23, Na20, Na21, Na22, Na23}
```

Out[708]=

```
<| n → 1, p → 2, d → 3, t → 4, He3 → 5, a → 6, He6 → 7, Li6 → 8, Li7 → 9, Li8 → 10, Li9 → 11,
 Be7 → 12, Be9 → 13, Be10 → 14, Be11 → 15, Be12 → 16, B8 → 17, B10 → 18, B11 → 19,
 B12 → 20, B13 → 21, B14 → 22, B15 → 23, C9 → 24, C10 → 25, C11 → 26, C12 → 27, C13 → 28,
 C14 → 29, C15 → 30, C16 → 31, N12 → 32, N13 → 33, N14 → 34, N15 → 35, N16 → 36,
 N17 → 37, O13 → 38, O14 → 39, O15 → 40, O16 → 41, O17 → 42, O18 → 43, O19 → 44,
 O20 → 45, F17 → 46, F18 → 47, F19 → 48, F20 → 49, Ne18 → 50, Ne19 → 51, Ne20 → 52,
 Ne21 → 53, Ne22 → 54, Ne23 → 55, Na20 → 56, Na21 → 57, Na22 → 58, Na23 → 59 |>
```

The list of variable in standard Ynipj forms are

In[709]:=

```
VarList = ToExpression /@ (StackY /@ ShortNames)
               |len expression
```

Out[709]=

```
{Yn1p0, Yn0p1, Yn1p1, Yn2p1, Yn1p2, Yn2p2, Yn4p2, Yn3p3, Yn4p3, Yn5p3, Yn6p3,
 Yn3p4, Yn5p4, Yn6p4, Yn7p4, Yn8p4, Yn3p5, Yn5p5, Yn6p5, Yn7p5, Yn8p5, Yn9p5,
 Yn10p5, Yn3p6, Yn4p6, Yn5p6, Yn6p6, Yn7p6, Yn8p6, Yn9p6, Yn10p6, Yn5p7,
 Yn6p7, Yn7p7, Yn8p7, Yn9p7, Yn10p7, Yn5p8, Yn6p8, Yn7p8, Yn8p8, Yn9p8,
 Yn10p8, Yn11p8, Yn12p8, Yn8p9, Yn9p9, Yn10p9, Yn11p9, Yn8p10, Yn9p10,
 Yn10p10, Yn11p10, Yn12p10, Yn13p10, Yn9p11, Yn10p11, Yn11p11, Yn12p11}
```

Abstract abundance functions

We build list of abundance function and abundance function derivatives which are used later to build the system of equations. These stay at an abstract level and they are used only to build the differential system of equations which is later solved by NDSolve.

In[710]:=

```
SetTimeDependence[list_List, tv_] := #[tv] & /@ list;
               |liste
```

In[711]:=

```
FunList[tv_] = SetTimeDependence[VarList, tv]
FunPrimeList[tv_] = FunList'[tv]
```

Out[711]=

```
{Yn1p0[tv], Yn0p1[tv], Yn1p1[tv], Yn2p1[tv], Yn1p2[tv], Yn2p2[tv], Yn4p2[tv],
Yn3p3[tv], Yn4p3[tv], Yn5p3[tv], Yn6p3[tv], Yn3p4[tv], Yn5p4[tv], Yn6p4[tv],
Yn7p4[tv], Yn8p4[tv], Yn3p5[tv], Yn5p5[tv], Yn6p5[tv], Yn7p5[tv], Yn8p5[tv],
Yn9p5[tv], Yn10p5[tv], Yn3p6[tv], Yn4p6[tv], Yn5p6[tv], Yn6p6[tv], Yn7p6[tv],
Yn8p6[tv], Yn9p6[tv], Yn10p6[tv], Yn5p7[tv], Yn6p7[tv], Yn7p7[tv], Yn8p7[tv],
Yn9p7[tv], Yn10p7[tv], Yn5p8[tv], Yn6p8[tv], Yn7p8[tv], Yn8p8[tv],
Yn9p8[tv], Yn10p8[tv], Yn11p8[tv], Yn12p8[tv], Yn8p9[tv], Yn9p9[tv],
Yn10p9[tv], Yn11p9[tv], Yn8p10[tv], Yn9p10[tv], Yn10p10[tv], Yn11p10[tv],
Yn12p10[tv], Yn13p10[tv], Yn9p11[tv], Yn10p11[tv], Yn11p11[tv], Yn12p11[tv]}
```

Out[712]=

```
{Yn1p0'[tv], Yn0p1'[tv], Yn1p1'[tv], Yn2p1'[tv], Yn1p2'[tv], Yn2p2'[tv], Yn4p2'[tv],
Yn3p3'[tv], Yn4p3'[tv], Yn5p3'[tv], Yn6p3'[tv], Yn3p4'[tv], Yn5p4'[tv],
Yn6p4'[tv], Yn7p4'[tv], Yn8p4'[tv], Yn3p5'[tv], Yn5p5'[tv], Yn6p5'[tv],
Yn7p5'[tv], Yn8p5'[tv], Yn9p5'[tv], Yn10p5'[tv], Yn3p6'[tv], Yn4p6'[tv],
Yn5p6'[tv], Yn6p6'[tv], Yn7p6'[tv], Yn8p6'[tv], Yn9p6'[tv], Yn10p6'[tv],
Yn5p7'[tv], Yn6p7'[tv], Yn7p7'[tv], Yn8p7'[tv], Yn9p7'[tv], Yn10p7'[tv],
Yn5p8'[tv], Yn6p8'[tv], Yn7p8'[tv], Yn8p8'[tv], Yn9p8'[tv], Yn10p8'[tv],
Yn11p8'[tv], Yn12p8'[tv], Yn8p9'[tv], Yn9p9'[tv], Yn10p9'[tv], Yn11p9'[tv],
Yn8p10'[tv], Yn9p10'[tv], Yn10p10'[tv], Yn11p10'[tv], Yn12p10'[tv],
Yn13p10'[tv], Yn9p11'[tv], Yn10p11'[tv], Yn11p11'[tv], Yn12p11'[tv]}
```

Numerical abundance functions

We also need the list of functions to store the results of the numerical integrations.

We have divided our BBN period in 3 eras.

High temperature (HT),

Middle temperature (MT)

Low temperature (LT).

So for instance the abundance of a species, say Lithium7 whose shortname is 'Li7', during the HT period at a given cosmological time t , is

$\text{YHT}[\text{"Li7"}][t]$.

Eventually we are interested in the values $\text{YLT}[\text{"species"}][\text{tend}]$ where tend is the final time of our numerical integration.

In[713]:=

```

KeyQ[key_] := MemberQ[NamesWithWeightsAll[All, 1], key];
               |appartient ?           |tout

YHT[key_?KeyQ][t_] := Y["HT"][key][t];
YMT[key_?KeyQ][t_] := Y["MT"][key][t];
YLT[key_?KeyQ][t_] := Y["LT"][key][t];

```

We make a list of the Y_i for a given period at a given time. This is typically used for initial conditions in Differential Solver.

In[717]:=

```

YPeriodTime[period_String][tv_] := Y[period][#][tv] & /@ ShortNames;
               |chaîne de caractères

```

We also define a function which can choose between previously numerically solved function in a given era, or equilibrium values for a list of species

In[718]:=

```

NumericalValueOrThermalEquilibriumValue[
  period_, ListThermal_, name_, Tv_, tv_] :=
  Module[
    |module
    {Yv = Y[period][name][tv], Yn = Y[period]["n"][tv], Yp = Y[period]["p"][tv]},
    If[ MemberQ[ListThermal, name], YNSE[name, Yn, Yp, Tv], Yv]];
    |si |appartient ?

```

We make a list of the Y_i for a given period at a given time. If there is no known numerical value it takes the thermostatistical equilibrium.

This is only used for the initial conditions in the middle era, where all species considered start at thermodynamical equilibrium or not very far from it except for neutrons and protons which are computed numerically from the high temperature era.

In[719]:=

```

YPeriodTimeOrStateEquilibrium[period_String, ListThermal_List][Tv_, tv_] :=
               |chaîne de caractères |liste
  NumericalValueOrThermalEquilibriumValue[period, ListThermal, #, Tv, tv] & /@
  ShortNames;

```

CNO

List of CNO nuclei. This includes all N and O, but only C with $A \geq 12$. C11 decays into B11 and C10 into B10 and C9 into $\alpha + 2p$.

In[720]:=

```

CNONuclei = KeyNucleons /@ Join[Table[{i, 6}, {i, 6, 10}],
               |joins |table
  Table[{i, 7}, {i, 5, 10}], Table[{i, 8}, {i, 5, 12}]]
               |table           |table

```

Out[720]=

```

{C12, C13, C14, C15, C16, N12, N13, N14,
 N15, N16, N17, O13, O14, O15, O16, O17, O18, O19, O20}

```


In[721]:=

```
YLT["CNO"][t_] := Plus@@ ((YLT[#][t] &) /@ CNONuclei)
      |plus
```

Coupled systems of differential reactions

Formal construction of r.h.s

This is a function which takes the list of reactions, with the specification of initial and final particles and their multiplicity, and builds the rhs of the differential system.

It builds the rhs of Eq 138 in companion paper.

In[722]:=

```

FillReactionMatrix[listreac_List] :=
  Module[{
    Tab, i, j, nvar, TreatReaction, FactorInitialElements,
    nvar = Length@VarList;
    Tab = Table[0, {i, 1, nvar}];
    FactorInitialElements[el_List] :=
      Times@@ (ApB / DensityUnit Y[KeyVal[#[[1]]]] ^#[[2]] / (#[[2]]!) & /@el);
    TreatReaction[reaction_List] := Module[{
      InitialParticles = Tally[RemoveNonNuclear@reaction[[2]],
      FinalParticles = Tally[RemoveNonNuclear@reaction[[3]],
      ReactionForward, ReactionBackward,
      FactorInitialForward, FactorInitialBackward},

      ReactionForward = L[KeyReaction[reaction[[1]]]];
      ReactionBackward = Lbar[KeyReaction[reaction[[1]]]];
      FactorInitialForward = FactorInitialElements@InitialParticles;
      (* This computes the product  $Y_i^{n_i}/n_i!$  for initial particles*)
      FactorInitialBackward = FactorInitialElements@FinalParticles;

      (Tab[[KeyVal[#[[1]]]]] = Tab[[KeyVal[#[[1]]]]] - ReactionForward
        FactorInitialForward#[[2]] / ApB * DensityUnit) & /@InitialParticles;
      (Tab[[KeyVal[#[[1]]]]] = Tab[[KeyVal[#[[1]]]]] + ReactionForward
        FactorInitialForward#[[2]] / ApB * DensityUnit) & /@FinalParticles;
      (Tab[[KeyVal[#[[1]]]]] = Tab[[KeyVal[#[[1]]]]] - ReactionBackward
        FactorInitialBackward#[[2]] / ApB * DensityUnit) & /@FinalParticles;
      (Tab[[KeyVal[#[[1]]]]] = Tab[[KeyVal[#[[1]]]]] + ReactionBackward
        FactorInitialBackward#[[2]] / ApB * DensityUnit) & /@InitialParticles;
    ];
    TreatReaction /@ listreac;
    Tab
  ]

```

To check the list of reactions used at this stage, just evaluate the next cell after uncommenting it.

In[723]:=

```

(*Print[ListReactionsUpToChosenMass];*)

```

FormalReactions is the list of rhs of differential equation in an abstract level.

-The species are given by Y[1], Y[2],Y[3], and we need the dictionary KeyVal to know to which species it corresponds.

-The reactions are L[1], L[2],L[3] and we also need a dictionary (KeyReaction) to know to which

reaction it corresponds.

In[724]:=

```
FormalReactions = FillReactionMatrix@ListReactionsUpToChosenMass;
```

To see how this works we look at a few elements of FormalReactions. For instance the source of Li7 due to nuclear reactions is one element of FormalReactions. It is

In[725]:=

```
FormalReactions[[KeyVal["Li7"]]]
```

Out[725]=

$$\begin{aligned}
 & A\rho B L[8] \times Y[4] \times Y[6] + \frac{1}{2} A\rho B Lbar[13] Y[6]^2 + \frac{1}{2} A\rho B Lbar[14] Y[6]^2 + \\
 & \frac{1}{2} A\rho B^2 Lbar[338] \times Y[1] Y[6]^2 + \frac{1}{4} A\rho B^3 Lbar[345] Y[1]^2 Y[6]^2 + \\
 & \frac{1}{2} A\rho B^3 Lbar[346] \times Y[1] \times Y[2] Y[6]^2 + \frac{1}{2} A\rho B^2 Lbar[295] \times Y[3] Y[6]^2 + \\
 & A\rho B L[355] \times Y[1] \times Y[8] + A\rho B L[356] \times Y[3] \times Y[8] + A\rho B L[293] \times Y[4] \times Y[8] + \\
 & A\rho B Lbar[68] \times Y[6] \times Y[8] - Lbar[8] \times Y[9] - Lbar[355] \times Y[9] - \\
 & A\rho B L[360] \times Y[1] \times Y[9] - A\rho B L[13] \times Y[2] \times Y[9] - A\rho B L[14] \times Y[2] \times Y[9] - \\
 & A\rho B Lbar[12] \times Y[2] \times Y[9] - A\rho B Lbar[356] \times Y[2] \times Y[9] - A\rho B L[66] \times Y[3] \times Y[9] - \\
 & A\rho B L[338] \times Y[3] \times Y[9] - A\rho B L[361] \times Y[3] \times Y[9] - A\rho B Lbar[293] \times Y[3] \times Y[9] - \\
 & A\rho B L[69] \times Y[4] \times Y[9] - A\rho B L[335] \times Y[4] \times Y[9] - A\rho B L[345] \times Y[4] \times Y[9] - \\
 & A\rho B Lbar[274] \times Y[4] \times Y[9] - A\rho B Lbar[348] \times Y[4] \times Y[9] - A\rho B L[67] \times Y[5] \times Y[9] - \\
 & A\rho B L[68] \times Y[5] \times Y[9] - A\rho B L[292] \times Y[5] \times Y[9] - A\rho B L[295] \times Y[5] \times Y[9] - \\
 & A\rho B L[346] \times Y[5] \times Y[9] - A\rho B Lbar[302] \times Y[5] \times Y[9] - A\rho B L[321] \times Y[6] \times Y[9] - \\
 & A\rho B L[359] \times Y[6] \times Y[9] - A\rho B Lbar[34] \times Y[6] \times Y[9] - A\rho B Lbar[76] \times Y[6] \times Y[9] - \\
 & A\rho B Lbar[95] \times Y[6] \times Y[9] + Lbar[360] \times Y[10] + A\rho B Lbar[361] \times Y[2] \times Y[10] + \\
 & A\rho B L[348] \times Y[3] \times Y[10] + A\rho B L[76] \times Y[5] \times Y[10] + A\rho B L[274] \times Y[2] \times Y[11] + \\
 & A\rho B L[12] \times Y[1] \times Y[12] + A\rho B L[302] \times Y[4] \times Y[12] + Lbar[66] \times Y[13] + \\
 & A\rho B Lbar[335] \times Y[1] \times Y[13] + A\rho B Lbar[292] \times Y[2] \times Y[13] + \\
 & A\rho B L[95] \times Y[3] \times Y[13] + Lbar[69] \times Y[14] + A\rho B L[34] \times Y[2] \times Y[14] + \\
 & Lbar[67] \times Y[18] + A\rho B Lbar[359] \times Y[1] \times Y[18] + Lbar[321] \times Y[19]
 \end{aligned}$$

* $A\rho B$ is an abstract variable which needs to be replaced with the appropriate ρB (taking into account that this is not exactly ρ_B but $n_B m_a$, see appendix of companion paper).

* The reactions $L[i]$ and $Lbar[i]$ are also the reactions, with i being the i -th line of ListReactions. For instance reaction $L[8]$ is

In[726]:=

```
NiceDisplayReaction[ListReactions[[8]]]
```

Out[726]=

```
{ta -> Li7g, {t, a}, {Li7, g}, 0, DAACV04}
```

The $Y[i]$ are the abstract abundances, corresponding to the i -th elements in VariableList. For instance 9 corresponds to Li7 and so $Y[9]$ is the abstract abundance of Li7.

```
In[727]:=
KeyVal["Li7"]
ShortNames[[9]]

Out[727]=
9

Out[728]=
Li7
```

We also build formally a small network of reaction which is used in the middle temperature era. It is the same thing but with a reduced number of equations.

```
In[729]:=
NReactionsSmallNetwork = If[$ReducedNetwork, SmallNuclearNetworkSize + 1, 18];
FormalReactions18 = FillReactionMatrix@Take[ListReactionsUpToChosenMass,
Min[NReactionsSmallNetwork, Length@ListReactionsUpToChosenMass]];
WeightsNuclear.FormalReactions // Simplify
```

We check that formally nucleons are conserved. So we compute formally the $\sum_i A_i dY_i/dt$ and check that it is 0.

```
In[731]:=
WeightsNuclear.FormalReactions // Simplify

Out[731]=
0
```

We also build a differential system with just neutrons and protons and the weak interactions for the high temperature era

```
In[732]:=
FormalReactionsOnlyPEN = FillReactionMatrix[{ReactionPEN}];
```

Actual r.h.s of differential system

Now we can build the rhs of the differential equation. This is obtained from its formal expression “FormalReactions” in which time dependence is added thanks to some replacement rules.

First replacement rules for the abstract abundances.

```
In[733]:=
Yi := Y[KeyVal[#]] & /@ ShortNames;

RulesY[tv_] := Thread[Rule[Yi, FunList[tv]]];
```

Let us visualize few of these rules to understand

In[735]:=

```
Take[RulesY[TV], 8]
|prends
```

Out[735]=

```
{Y[1] → Yn1p0[TV], Y[2] → Yn0p1[TV], Y[3] → Yn1p1[TV], Y[4] → Yn2p1[TV],
Y[5] → Yn1p2[TV], Y[6] → Yn2p2[TV], Y[7] → Yn4p2[TV], Y[8] → Yn3p3[TV]}
```

Then for the abstract reactions, we also define rules to replace the abstract reaction $L[i]$ or its reverse $Lbar[i]$ by the actual rate.

In[736]:=

```
Li = L[KeyReaction[##]] & /@ ListReactionsNames;
Lbari = Lbar[KeyReaction[##]] & /@ ListReactionsNames;
```

In[738]:=

```
RulesλRHS[TV_] := Symbol["L" <> #][TV] & /@ ListReactionsNames;
|_symbole

RulesλRHS[n_, TV_] := Symbol["L" <> #][TV] & /@ Take[ListReactionsNames, n];
|_symbole |prends

RulesλbarRHS[TV_] := Symbol["Lbar" <> #][TV] & /@ ListReactionsNames;
|_symbole

RulesλbarRHS[n_, TV_] :=
  Symbol["Lbar" <> #][TV] & /@ Take[ListReactionsNames, n];
|_symbole |prends

Rulesλ[TV_] := Thread[Rule[Li, RulesλRHS[TV]]];
|_enfile |règle





Rulesλbar[TV_] := Thread[Rule[Lbari, RulesλbarRHS[TV]]];
|_enfile |règle
```

Let us visualize a few of these rules.

In[744]:=

```
Take[Rulesλ[TV], 4]
|prends
```

Out[744]=

```
{L[1] → 0.001137139 InterpolatingFunction[ Domain: {{1. × 107, 1. × 1012}}] [TV],
L[2] → InterpolatingFunction[ Domain: {{1. × 106, 1. × 1010}}] [TV],
L[3] → InterpolatingFunction[ Domain: {{1. × 106, 1. × 1010}}] [TV],
L[4] → InterpolatingFunction[ Domain: {{1. × 106, 1. × 1010}}] [TV]}
```

Let us apply these rules to form the r.h.s

Here is the r.h.s of the differential system for nuclear reactions. We distinguish between high,

middle and low temperature system of equations.

In[745]:=

```
DYOnlyPEN[Temp_, ρB_, time_] :=
  (FormalReactionsOnlyPEN) /. Dispatch@Rulesλ[Temp] /.
    _affecte
    Dispatch@Rulesλbar[Temp] /. Dispatch@RulesY[time] /. AρB → ρB;
    _affecte          _affecte

DY18[Temp_, ρB_, time_] :=
  (FormalReactions18) /. Dispatch@Rulesλ[Temp] /. Dispatch@Rulesλbar[Temp] /.
    _affecte          _affecte
    Dispatch@RulesY[time] /. AρB → ρB;
    _affecte

DY[Temp_, ρB_, time_] :=
  (FormalReactions) /. Dispatch@Rulesλ[Temp] /. Dispatch@Rulesλbar[Temp] /.
    _affecte          _affecte
    Dispatch@RulesY[time] /. AρB → ρB;
    _affecte
```

In[748]:=

```
(*DY[Tv,rv,tv][[5]];//Timing*)
    _chronométrage
```

We define compiled version which are used if the \$CompileNDSolve option is set to True. This is much faster.

Otherwise if \$CompileNDSolve=False, the DY, DY18 and DYPEN are called in DefineEquations further below when it is associated with the l.h.s to form the differential system.

In[749]:=

```
CompileFromFormal[FormalReactions_List] := ReleaseHold[
    _liste          _maintiens déverrouillage
    Hold[Compile[{{AρB, _Real}, {L, _Real, 1}, {Lbar, _Real, 1}, {Y, _Real, 1}},
    _main... _compile          _nombre réel          _nombre réel          _nombre réel          _nombre réel
      inside, CompilationTarget → "C", "RuntimeOptions" → "Speed",
        _cible de compilation          _con... _options de durée d'exécution
      CompilationOptions → {"InlineExternalDefinitions" → True}]] //.
    _options de compilation          _vrai
    {inside → FormalReactions, Y[m_] => Y[[m]], L[m_] => L[[m]], Lbar[m_] => Lbar[[m]]}]
```

In[750]:=

```
Timing[If[$CompileNDSolve,
  chronos[si
    DYC = CompileFromFormal[FormalReactions];
    DY18C = CompileFromFormal[FormalReactions18];
    DYCN[AρB_?NumericQ, L_, Lbar_, Y_] := DYC[AρB, L, Lbar, Y];
    DY18CN[AρB_?NumericQ, L_, Lbar_, Y_] := DY18C[AρB, L, Lbar, Y];
  ];
```

Out[750]=

```
{1.064887, Null}
```

Time integration of Cosmology and BBN

Friedmann equation

The Friedmann equation gives the Hubble expansion rate.

For completeness we put baryons and CDM even though it makes a difference of order 10^{-5} which is below what we can achieve anyway with homogeneous computations.

$$H^2 = \frac{8\pi G\rho}{3} = \frac{\rho}{\rho_{\text{crit}}}$$

so we build first the energy density.

We select the neutrino energy density depending on options for decoupling

In[751]:=

```
ρν[T_] := If[$IncompleteNeutrinoDecoupling,
  si
  ρνIncompleteDecoupling[a[T]], ρνDecoupling[T]];
```

Two methods to get the total energy density. Either from energy density of neutrinos or from temperature. This should be (and it is...) totally equivalent.

In[752]:=

```
ρtot1[T_] := (aBB (kB T)^4 (1 + DρT[T]) + ρν[T]
  + nbaryons0 (mbaryon0 * (1 + h2Ωc0 / h2Ωb0) + 3/2 (kB T)) / (c light)^2 / (a[T])^3);

ρtot2[T_] := (aBB (kB T)^4 (1 + 7/8 Nneu (TvAverageoverT[T])^4 + DρT[T])
  + nbaryons0 (mbaryon0 * (1 + h2Ωc0 / h2Ωb0) + 3/2 (kB T)) / (c light)^2 / (a[T])^3);
```

In[754]:=

```

ρtot[T_] := ρtot2[T]; (* If $TaverageN=True,
                        |si |vrai
ρtot1 and ρtot2 are equivalent. In general, we prefer ρtot2
                        |dans
which uses the effective temperatures (better consistency).*)

```

In[755]:=

```

ρtot2[10^10]
ρtot1[10^10]

```

Out[755]=

442 826.24

Out[756]=

442 829.05

We check that we have the correct asymptotic behaviours for the degrees of freedom (Beware that QED corrections alter a bit the results)

In[757]:=

```

ρtot[T] / (aBB (kB T)^4) /. T -> 10^12 // NP
(2 + 4 * 7 / 8 + 3 * 2 * 7 / 8) / 2 // N // NP
                        |valeur numérique

```

Out[757]//NumberForm=

5.3683834

Out[758]//NumberForm=

5.375

In[759]:=

```

ρtot[T] / (aBB (kB T)^4) /. T -> 10^7.5 // NP
(2 + 3 * 2 * 7 / 8 * (FourOverEleven)^(4/3)) / 2 // N // NP
                        |valeur numérique

```

Out[759]//NumberForm=

1.6918036

Out[760]//NumberForm=

1.6835124

We plot the energy density as a function of temperature.

In[761]:=

```

If[$PaperPlots, ListLogLogPlot[
|si |tracé log-log de liste
  Table[{T, 1 / T^4 ρtot[T]}, {T, ListT}], Frame -> True, Joined -> True,
  |table |cadre |vrai |joint |vrai
  PlotStyle -> Black, FrameLabel -> {"T (K)", "T^-4 ρ(T) g cm^-3 K^-4"}]]
|style de tracé |noir |étiquette de cadre
If[$PaperPlots, Export["Plots/PlotρT.pdf", Style[%, Magnification -> 1], "PDF"];]
|si |exporte |style |agrandissement |fonction de

```

Hubble function from Friedmann equation in flat FL space-time is then immediate

In[763]:=

$$H[a_] := \left(\frac{8 \pi G N}{3} \rho_{\text{tot}}[T_{\text{ofa}}[a]] \right)^{(1/2)};$$

Time and scale factor

This is the solver for dt/da giving $t(a)$. Direct integration of the Friedmann equation

In[764]:=

```
Computetofa :=
  (tofa = NDSolveValue[{tv'[av] ==  $\frac{1}{(av H[av])}$ , tv[a[Ti]] ==  $\frac{1}{(2 H[a[Ti]])}$ },
    [valeur donnée par solution numérique de l'équation différentielle]
    tv, {av, a[Ti], a[Tf]}, PrecisionGoal → 8, AccuracyGoal → 10];)
    [objectif de précision] [objectif d'exactitude]
```

We check that it is quick enough.

In[765]:=

```
Timing@Computetofa
[chronométrage]
```

Out[765]=

```
{0.010257, Null}
```

This is the solver for the inverse function, that is for da/dt , giving $a(t)$

In[766]:=

```
Computeaoft :=
  (aoft = NDSolveValue[{av'[tv] ==  $\frac{1}{tofa'[av[tv]]}$ , av[tofa@a[Ti]] == a[Ti]}, av,
    [valeur donnée par solution numérique de l'équation différentielle]
    {tv, tofa@a[Ti], tofa@a[Tf]}, PrecisionGoal → 75, AccuracyGoal → 20];)
    [objectif de précision] [objectif d'exactitude]
```

We check that it is quick enough.

In[767]:=

```
Timing@Computeaoft
[chronométrage]
```

Out[767]=

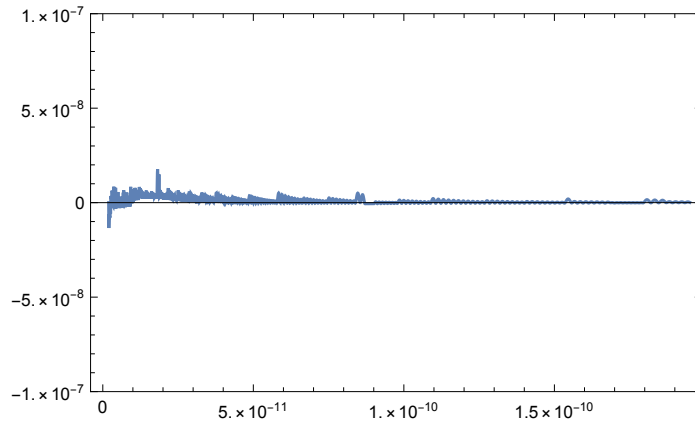
```
{0.010461, Null}
```

We also check that computing $a(t(a))$ gives negligible error to identity. It is of order 10^{-9} so we are totally OK.

In[768]:=

```
Plot[1 / av * aoft@tofa@av - 1, {av, a[Ti], a[Ti] * 100},
  _tracé
  PlotRange → {-10^-7, 10^-7}, Frame → True]
  _zone de tracé _cadre _vrai
```

Out[768]=



We define the temperature as a function of time. Since we have $T(a)$ (conserved entropy) and $a(t)$ (Friedman equation), we just combine both.

In case of incomplete decoupling of neutrinos, we could not use conservation of entropy to get $T(a)$ but we used the fit for the heating rate which allowed us to obtain numerically $T(a)$ and $a(T)$.

In[769]:=

```
Toft[tv_] := Tofa[aoft[tv]];
```

Initial nuclear conditions

We use equilibrium solution for the initial condition. This equation A15 of companion paper.

In[770]:=

```
Yni[Tv_] := 1 / (1 + (1 - 3/2 * Q/mn) Exp[Q/(kB Tv)]);
Ypi[Tv_] := 1 - Yni[Tv];
```

Other initial conditions based on the equation and the rates. It is better when including corrections because these are the correct initial conditions whatever the corrections.

In[772]:=

```
Yn2i[Tv_] := LpT0n[Tv] / (LpT0n[Tv] + LnT0p[Tv]);
Yp2i[Tv_] := 1 - Yn2i[Tv];
```

We see that the difference is very small

```
In[774]:=
Yni[10^11.5]
Yn2i[10^11.5]
```

```
Out[774]=
0.48865342
```

```
Out[775]=
0.4889842
```

Actual list of initial conditions. Vanishing for everything except protons and neutrons

```
In[776]:=
CIList[Tv_] :=
  Table[Which[i == 1, Yni[Tv], i == 2, Ypi[Tv], i ≥ 3, 0], {i, 1, NumberVariable}]


|  |  |
|--|--|
|  |  |
|--|--|


```

Construction of differential equations

The function DefineEquations wraps the definition of differential equations. This must be called any time we regenerate the probabilities on the reaction rates when including uncertainties on reaction rates in a Monte-Carlo analysis.

```
In[777]:=
DefineEquations := (

  (*We build the differential system for the High temperatures *)
  (* So we associate the r.h.s which is constructed thanks to DYOnlyPEN,
  with the l.h.s made of abundances derivatives *)
  SystemEquationsHT[tv_] = Thread[Equal[FunPrimeList[tv],


|  |  |
|--|--|
|  |  |
|--|--|


    (DYOnlyPEN[Tv, ρBv, tv] (*Dispatch@ReactionProbabilities*))]] /.


|  |  |
|--|--|
|  |  |
|--|--|


    {Tv → Toft@tv, ρBv → ρBForBBN@a@Toft@tv};

  (* For middle and low temperature we distinguish
  

|  |  |
|--|--|
|  |  |
|--|--|


  between the compiled and the uncompiled method *)

  If[$CompileNDSolve,


|  |  |
|--|--|
|  |  |
|--|--|



    (* If $CompileNDSolve=True,


|  |  |
|--|--|
|  |  |
|--|--|


    we reinterpolate the rates for the middle and the low temperatures. *)
    (* The system is a matrix system in
    this case and it is built directly in NDSolve below *)


|  |  |
|--|--|
|  |  |
|--|--|



    RulesλRHSI = MyInterpolationRate@
      Table[{Tv, MyChop@RulesλRHS[Tv]}, {Tv, ListTRange[Tf, T18]}};


|  |  |
|--|--|
|  |  |
|--|--|



    RulesλbarRHSI = MyInterpolationRate@
```

```

Table[{Tv, MyChop@RulesλbarRHS[Tv]}, {Tv, ListTRange[Tf, T18]}}];
table
RulesλRHS18I =
MyInterpolationRate@Table[{Tv, RulesλRHS[NReactionsSmallNetwork, Tv]},
table
{Tv, ListTRange[T18, TMiddle]}}];
RulesλbarRHS18I =
MyInterpolationRate@Table[{Tv, RulesλbarRHS[NReactionsSmallNetwork, Tv]},
table
{Tv, ListTRange[T18, TMiddle]}}];

(* If $CompileNDSolve=False, we (re-)define the
si faux
systems of equations for Middle and Low temperatures*)
(* We associate the r.h.s formed thanks to DY18 and DY,
with the l.h.s made of derivatives *)
SystemEquationsMT[tv_] =
Thread[Equal[FunPrimeList[tv], (DY18[Tv, ρBv, tv])]] /.
lenfile égal
{Tv → Toft@tv, ρBv → ρBForBBN@a@Toft@tv};
SystemEquationsLT[tv_] =
Thread[Equal[FunPrimeList[tv], (DY[Tv, ρBv, tv])]] /.
lenfile égal
{Tv → Toft@tv, ρBv → ρBForBBN@a@Toft@tv};
]
)

```

In[778]:=

DefineEquations; // Timing

_{chronométrage}

... General : Exp[−2581.5] is too small to represent as a normalized machine number; precision may be lost. i

... General : Exp[−6375.18] is too small to represent as a normalized machine number; precision may be lost. i

... General : Exp[−3793.41] is too small to represent as a normalized machine number; precision may be lost. i

... General : Further output of General::munfl will be suppressed during this calculation. i

Out[778]=

{2.463647, Null}

Time delimitation of low, middle and high temperature eras

The time delimitation corresponding to Temperature delimitations of high, middle and low temperature eras.

In[779]:=

```

t0 := tofa@a[Tstart];
tmiddle := tofa@a[TMiddle];
t18 := tofa@a[T18];
tend := tofa@a[Tend];

```

Let us check the values in seconds or $t_0 < t_{\text{middle}} < t_{18} < t_{\text{end}}$

In[783]:=

```
{t0, tmiddle, t18, tend}
```

Out[783]=

```
{0.0099473654, 1.0068624, 99.705829, 49 227.574}
```

High temperature integration (n and p only)

The initial conditions at high temperature are found from thermal and chemical equilibrium. This is used to integrate from 10^{11} K to 10^{10} K. We keep track only of neutrons and protons.

In[784]:=

```

HoldYNames[period_String] :=
  chaîne de caractères
  ToExpression /@ ("Hold@Y[" <> period <> "]" ["[" <> # <> "]" & /@ ShortNames) ;
  en expression      maintiens

```

Initial conditions

In[785]:=

```

InitialConditionsHT[tv_] := Thread[Equal[FunList[tv], CIList[Toft[tv]]]];
  enfile      égal

```

Actual solver. It solves the system and affects the results to the functions YHT["n"] (neutrons) and YHT["p"] (protons) which are functions of time.

In[786]:=

```

SolveValueHighTemperatures := (Thread[MySet[Evaluate[HoldYNames["HT"]],
  enfile      évalue
  NDSolveValue[
    valeur donnée par solution numérique d'équation différentielle
    Flatten@Join[SystemEquationsHT[tv], InitialConditionsHT[t0]],
    aplatis    joins
    VarList, {tv, t0, tmiddle},
    PrecisionGoal → 8 + PrecisionNDSolve,
    objectif de précision
    AccuracyGoal → 11, InterpolationOrder → InterpOrder]]];
    objectif d'exactitude      ordre d'interpolation
    tHT = Y["HT"] ["n"] [[3, 1]];
  );

```

This period is very quick to solve as can be checked. The variable tHT stores the time steps used by the solver in case we are interested.

```
In[787]:= AbsoluteTiming[SolveValueHighTemperatures;]
           [durée absolue
```

```
Out[787]= {0.035907, Null}
```

We can also extend this integration with only weak interactions to much later times to see what would happen without nuclear reactions.

This is only to perform plots in the paper.

```
In[788]:= SolveValueHighTemperaturesYnOnly :=
  (Thread[MySet[Evaluate[HoldYNAMES["WeakInteractionsOnly"]],
    [enfile [évalue
    NDSolveValue[
    [valeur donnée par solution numérique d'équation différentielle
    Flatten@Join[SystemEquationsHT[tv], InitialConditionsHT[t0]],
    [aplatis [joins
    VarList,
    {tv, t0, tend},
    PrecisionGoal → 8 + PrecisionNDSolve, AccuracyGoal → 11 (* 11*),
    [objectif de précision [objectif d'exactitude
    InterpolationOrder → InterpOrder]]];
    [ordre d'interpolation
  );
```

```
In[789]:= If[$ResultsPlots, SolveValueHighTemperaturesYnOnly];]
           [si
```

Middle temperature integration (n,p,d,t,He3,He4,Be7,Li7,Li6)

The end values for neutrons and protons at high temperatures are used as initial conditions for the middle temperature era.

We use thermostatistical equilibrium for the species defined in the list ListThermalValuesMT below, and 0 otherwise.

```
In[790]:= ListThermalValuesMT = If[$ReducedNetwork, {"d", "t", "He3", "a", "Be7", "Li7"},
           [si
           {"d", "t", "He3", "a", "Be7", "Li7", "Li6"}];

ListThermalValuesUsedMT =
  Intersection[VariablesInEquations, ListThermalValuesMT]
           [intersection
```

```
Out[791]= {a, Be7, d, He3, Li6, Li7, t}
```

We check the initial value used for the middle temperature era.

In[794]:=

```

SolveValueMiddleTemperatures := (If[$CompileNDSolve,
    (* Compiled version. *)
    resMT = NDSolveValue[
        {Ytab'[tv] == DY18CN[ρBForBBN@a@Toft@tv,
            RulesλRHS18I[Toft@tv], RulesλbarRHS18I[Toft@tv], Ytab[tv]],
        Ytab[tmiddle] == YPeriodTimeOrStateEquilibrium["HT",
            ListThermalValuesUsedMT][TMiddle, tmiddle]},
        Ytab, {tv, tmiddle, t18},
        Method → {"BDF", "MaxDifferenceOrder" → $BDFOrder},
        PrecisionGoal → 7 + PrecisionNDSolve, AccuracyGoal → 11,
        InterpolationOrder → InterpOrder, Compiled → Automatic];
    Y["MT"][key_][tv_?NumericQ] := resMT[tv][KeyVal[key]];
    tMT = resMT[[3, 1]];

    (* Non compiled version. Slightly slower *)
    Thread[MySet[Evaluate[HoldYNames["MT"]], NDSolveValue[
        Flatten@
        Join[SystemEquationsMT[tv], InitialConditionsMT[TMiddle, tmiddle]],
        VarList, {tv, tmiddle, t18},
        Method → {"BDF", "MaxDifferenceOrder" → $BDFOrder},
        PrecisionGoal → 7 + PrecisionNDSolve, AccuracyGoal → 11,
        InterpolationOrder → InterpOrder, Compiled → False]]];
    tMT = Y["MT"]["n"][[3, 1]];
]);

```

NB: tMT stores the time steps used. Can be used to check the behaviour of the integrator.

In[795]:=

```

AbsoluteTiming[SolveValueMiddleTemperatures;]

```

Out[795]=

```
{0.232891, Null}
```

For information we plot the result of the integration

In[796]:=

```

If[$ResultsPlots, LogLogPlot[Evaluate[YPeriodTime["MT"][tv]], {tv, tmiddle, t18},
    Frame → True, PlotRange → {10^-40, 10}, FrameLabel → {"t(s)", "Yi"}]]

```


Low temperature integration (All 59 isotopes)

!With the reduce network the list of nuclides is the same as in middle temperature range!

The end values at middle temperatures are then used as initial conditions for the low temperature era.

Now the full system of equations is used.

In[797]:=

```
InitialConditionsLT[tv_] := Thread[Equal[FunList[tv], YPeriodTime["MT"][tv]]];
                               lenfile légal
```

In[798]:=

```

SolveValueLowTemperatures := (If[$CompileNDSolve,
                                si
                                (* Compiled version *)
                                compilé
                                resLT = NDSolveValue[
                                    valeur donnée par solution numérique d'équation différentielle
                                    {Ytab'[tv] == DYCN[ρBForBBN@a@Toft@tv,
                                        RulesλRHSI[Toft@tv], RulesλbarRHSI[Toft@tv], Ytab[tv]],
                                      Ytab[t18] == YPeriodTime["MT"][t18]},
                                    Ytab, {tv, t18, tend},
                                    Method → {"BDF", "MaxDifferenceOrder" → $BDFOrder},
                                    méthode
                                    PrecisionGoal → 5 + PrecisionNDSolve, AccuracyGoal → AccuracyNDSolve,
                                    objectif de précision objectif d'exactitude
                                    InterpolationOrder → InterpOrder,
                                    ordre d'interpolation
                                    StartingStepSize → 10^-4, MaxStepSize → 500];
                                Y["LT"][key_][tv_?NumericQ] := resLT[tv][[KeyVal[key]]];
                                expression numérique ?
                                tLT = resLT[[3, 1]];

                                (* Uncompiled version. Slower. *)
                                Thread[MySet[Evaluate[HoldYNAMES["LT"]], NDSolveValue[
                                    enfile évalue valeur donnée par solution numérique d'équation diffi
                                    Flatten@Join[SystemEquationsLT[tv], InitialConditionsLT[t18]],
                                    aplatis joins
                                    VarList, {tv, t18, tend},
                                    Method → {"BDF", "MaxDifferenceOrder" → $BDFOrder},
                                    méthode
                                    "EquationSimplification" → "Solve"},
                                    résous
                                    PrecisionGoal → 5 + PrecisionNDSolve, AccuracyGoal → AccuracyNDSolve,
                                    objectif de précision objectif d'exactitude
                                    InterpolationOrder → InterpOrder, StartingStepSize → 10^-4]]];
                                ordre d'interpolation taille d'étape initiale
                                tLT = Y["LT"]["n"][[3, 1]];
                                ];)

```

In[799]:=

```

AbsoluteTiming[SolveValueLowTemperatures;]
durée absolue

```

Out[799]=

```
{13.435408, Null}
```

We can plot the results

In[800]:=

```

If[$ResultsPlots, LogLogPlot[Evaluate[YPeriodTime["LT"][tv]], {tv, t18, tend},
                                si tracé log-log évalue
                                Frame → True, PlotRange → {10^-40, 10}, FrameLabel → {"t(s)", "Yi"}]]
                                cadre vrai zone de tracé étiquette de cadre

```

Gathering integrations on all periods in one function

We define an interpolation of the results. We join the results from high, middle and low temperature eras. This is joined in the function

YI[“key”][time] where key is the name of the nuclide (e.g. “a” for He4, “t” for tritium and “d” for deuterium but otherwise “Li7”, “C12” etc...).

In[801]:=

```
InterpolateResults = (
  Clear[Yall, YI];
  |efface
  Yall[key_?KeyQ] := Yall[key] = Function[{tv},
    |fonction
    Piecewise[{{Y["HT"][key][tv], tv < tmiddle}, {Y["MT"][key][tv],
      |fonction par morceaux
      tv < t18 && tv ≥ tmiddle}, {Y["LT"][key][tv], tv ≤ tend && tv ≥ t18}}]];

  YI[key_?KeyQ] := YI[key] = Interpolation[Table[{tv, Yall[key][tv]},
    |interpolation |table
    {tv, Join[tHT, Rest@tMT, Rest@tLT]}], InterpolationOrder → 1];
    |joins |reste |reste |ordre d'interpolation
```

In[802]:=

InterpolateResults;

The function RunNumericalIntegrals below performs the integrations of incomplete neutrino decoupling

and then the Friedmann equation integration.

Then it defines the nuclear reactions, possibly having introduced uncertainty on rates depending on options,

and solves for the high, middle and low temperature era.

This is the Driver of PRIMAT which needs to be called whenever we rerun PRIMAT with new parameters (e.g. exploring dependence in baryons or neutrinos).

In[803]:=

```
RunNumericalIntegralsNuclearReactions := (
  (* Middle temperature integration *)
  SolveValueMiddleTemperatures;

  (* Low temperature integration *)
  SolveValueLowTemperatures;
);
```

In[804]:=

```

RunNumericalIntegrals := (

  (* In case of incomplete neutrino decoupling, we recompute
     dans
     all the integrations a(T) then inversion T(a), then  $\rho_\nu(a)$  *)
  If[$IncompleteNeutrinoDecoupling,
     si
     RecomputeIncompleteNeutrinoDecoupling;];

  (*In case the plasma conditions have changed in a MC exploration,
     dans
     we recompute the inversion of a[T]*)
  (* This is needed if we have recomputed the neutrino decoupling,
     but I am wondering if this is always needed. *)
     unité imaginaire
  InvertaOFT;

  (* scale factor integration from Friedmann equation. *)
  Computetofa;
  Computeaoft;

  (* Build equations. Needed since rate are
     modified randomly by the f factor of each reaction*)
  LoadRates;
  DefineEquations;

  (* High temperature integration with only PEN reactions *)
  SolveValueHighTemperatures;

  (* Middle and Low temperature WITH nuclear reactions*)
  RunNumericalIntegralsNuclearReactions;

  InterpolateResults;
);

```

Gathering the numerical results

We define a pseudo-mass fraction as a function of time using the interpolated results

In[805]:=

```

XI[key_?KeyQ][t_] := Ai[key] × YI[key][t]
XI["CNO"][t_] := Plus@@ ((Ai[#] × YI[#][t] &) /@ CNONuclei)
                    plus

```

Final abundances are obtained by evaluation at $t = t_{\text{end}}$.

We define a shorthand for the final abundances (Yf) and pseudo mass fraction (Xf).

In[807]:=

```
Yf[key_] := YLT[key][tend]
Xf[key_] := Ai[key] × Yf[key]

Xf["CNO"] := Plus@@ (Xf[#] &) /@ CNO nuclei
```

And a shorthand notation for Y_i / H

In[810]:=

```
YfH[key_] := Yf[key] / Yf["p"]
```

Results and plots

Checks of the conservation of total number of baryons

The total number MUST be conserved. We build it from the nuclear weights of species.

At high temperatures we check visually. By construction the quantity $Y_{tot} =$

$\sum_i A_i Y_i$ should be 1 and conserved. We define it and plot the difference with unity.

In[811]:=

```
Ytot[period_String] :=
  chaîne de caractères

Function[{tv}, Plus@@ (WeightsNuclear * YPeriodTime[period][tv])];
```

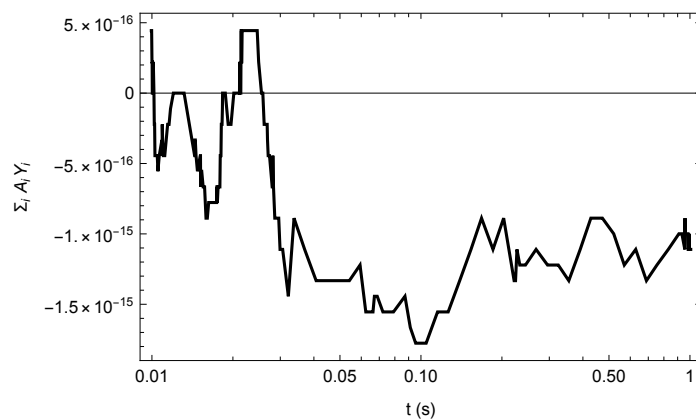
High temperature era

In[812]:=

```
LogLinearPlot[Ytot["HT"][tv] - 1, {tv, t0, tmiddle},
  tracé log-linéaire

Frame → True, FrameLabel → {"t (s)", "Σi AiYi"}, PlotStyle → Black]
cadre vrai étiquette de cadre style de tracé noir
```

Out[812]:=

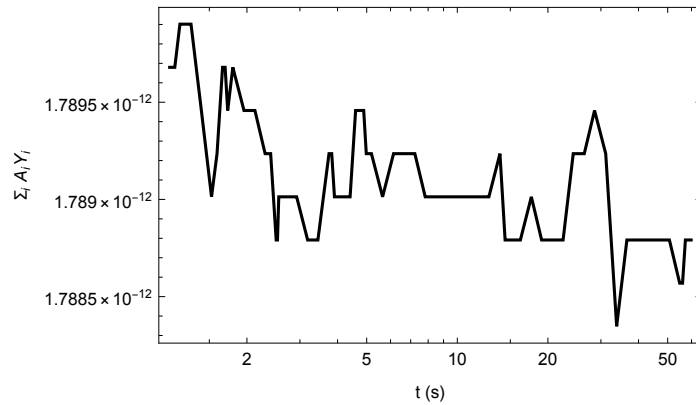


Middle temperature era

In[813]:=

```
LogLinearPlot[Ytot["MT"][tv] - 1, {tv, tmiddle * 1.1, 0.6 t18},
  _tracé log-linéaire
  Frame → True, FrameLabel → {"t (s)", "Σi AiYi"}, PlotStyle → Black]
  _cadre _vrai _étiquette de cadre _style de tracé _noir
```

Out[813]=

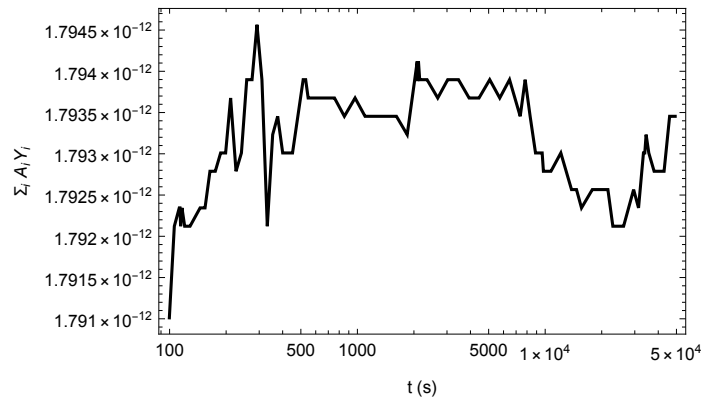


Low temperature era with the full network.

In[814]:=

```
LogLinearPlot[Ytot["LT"][t] - 1, {t, t18, tend},
  _tracé log-linéaire
  Frame → True, FrameLabel → {"t (s)", "Σi AiYi"}, PlotStyle → Black]
  _cadre _vrai _étiquette de cadre _style de tracé _noir
```

Out[814]=



Time evolution of abundances

Early thermodynamical equilibrium

Estimate of T_{nuc} (see companion paper)

In[815]:=

```
TFreeze = 0.8 MeV / kB;
tFreeze = tofa@a@TFreeze
YnF[tv_] := 1 / (1 + Exp[Q / kB / TFreeze]) Exp[-(tv - tFreeze) / τneutron];
  _exponentielle _exponentielle
YpF[tv_] := 1 - YnF[tv];
```

Out[816]=

1.1704954

```
In[819]:=
  tnuc = FindRoot[YNSE["d", YnF[tv], YpF[tv], Toft[tv]] == YnF[tv], {tv, 100}][[1, 2]]
  Tnuc = Toft[tnuc]
```

```
Out[819]=
  296.70458
```

```
Out[820]=
  7.6959596 × 108
```

Abundance of neutrons at T_{nuc} and T_{nuc} in MeV

```
In[821]:=
  YnF[tnuc]
  kB * Tnuc / MeV
```

```
Out[821]=
  0.11838073
```

```
Out[822]=
  0.066318639
```

In[823]:=

```

PlotDeutEq =
Show[LogLogPlot[{YnF[tofa@a[10^8 Tv]], YNSE["d", YnF[tofa@a[10^8 Tv]],
mon... tracé log-log

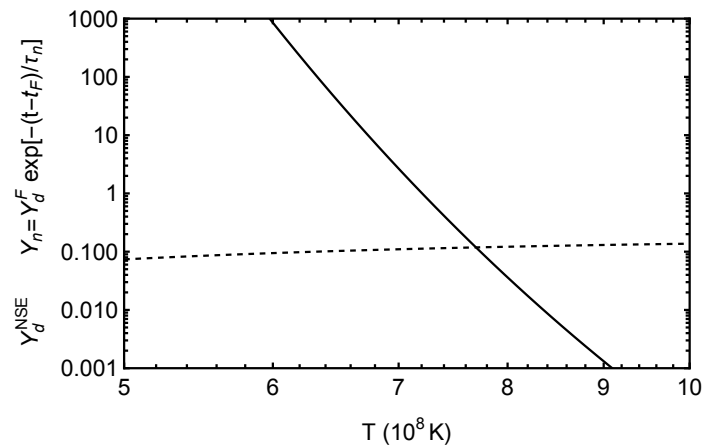
    YpF[tofa@a[10^8 Tv]], 10^8 Tv]], {Tv, 0.05 * 10^2, 0.1 * 10^2},
    GridLines -> {{{Tnuc, {Gray, Thickness[0.005]}}}, {}}, Frame -> True,
    lignes de grille      gris      épaisseur      cadre      vrai
    PlotRange -> {{0.05 * 10^2, 0.1 * 10^2}, {10^-3, 10^3}},
    zone de tracé
    PlotRangePadding -> None, FrameStyle -> Thickness[0.004],
    garnissage de zone de tr... aucun style de cadre épaisseur
    PlotStyle -> {{Black, Thickness[0.004], Dashing[0.01]},
    style de tracé      noir      épaisseur      style de tirets
    {Black, Thickness[0.004]}}, PlotRange -> {10^-3, 1000},
    noir      épaisseur      zone de tracé
    FrameLabel -> {"T (10^8 K)", "Y_d^NSE      Y_n=Y_d^F exp[-(t-t_F)/tau_n]"},
    étiquette de cadre
    LabelStyle -> {FontSize -> 12}],
    style d'étiquette      taille de police de caractères

    Graphics[{Rotate[Text[Style["0.066 MeV", FontSize -> 10, Black],
    graphique      fais pivo... texte      style      taille de police de ... noir
    {Log@Tnuc - 0.015, 2}], 90 Degree]]}]
    logarithme      degré

If[$ResultsPlots,
si
    Export["Plots/PlotDeutEq.pdf", Style[PlotDeutEq, Magnification -> 1], "PDF"];]
    exporte      style      agrandissement      fonction de

```

Out[823]=



Checks of thermo equilibrium at early times.

We check the accuracy of thermal equilibrium for “d” “t” “a” “Li7”.

Most important is deuterium because it determines the final Helium abundance.

In[825]:=

```
LogLogPlot[{YI["d"][tv] / YNSE["d"], YI["n"][tv], YI["p"][tv], Toft[tv]}],
  _tracé log-log
```

```
{tv, tmiddle*1.1, 50}, Frame → True, FrameLabel → {"t (s)", "Yi"},
  _cadre _vrai _étiquette de cadre
```

```
PlotStyle → {Black, {Black, Dashed}},
  _style de tracé _noir _noir _en tirets
```

```
ImagePadding → {{50, 10}, {40, 25}}, PlotRange → {0.999, 1.001}
  _garnissage d'image _zone de tracé
```

```
LogLogPlot[{YI["t"][tv] / YNSE["t"], YI["n"][tv], YI["p"][tv], Toft[tv]}],
  _tracé log-log
```

```
{tv, tmiddle*1.1, 10}, Frame → True, FrameLabel → {"t (s)", "Yi"},
  _cadre _vrai _étiquette de cadre
```

```
PlotStyle → {Black, {Black, Dashed}},
  _style de tracé _noir _noir _en tirets
```

```
ImagePadding → {{50, 10}, {40, 25}}, PlotRange → {0.999, 1.001}
  _garnissage d'image _zone de tracé
```

```
(*LogLogPlot[{YI["a"][tv] / YNSE["a"], YI["n"][tv], YI["p"][tv], Toft[tv]}],
  _tracé log-log
```

```
{tv, tmiddle*1.1, 1.5}, Frame → True, FrameLabel → {"t (s)", "Yi"},
  _cadre _vrai _étiquette de cadre
```

```
PlotStyle → {Black, {Black, Dashed}}, FrameTicks → MyTicks,
  _style de tracé _noir _noir _en tirets _graduations de cadre
```

```
ImagePadding → {{50, 10}, {40, 25}}, PlotRange → {0.999, 1.001}
  _garnissage d'image _zone de tracé
```

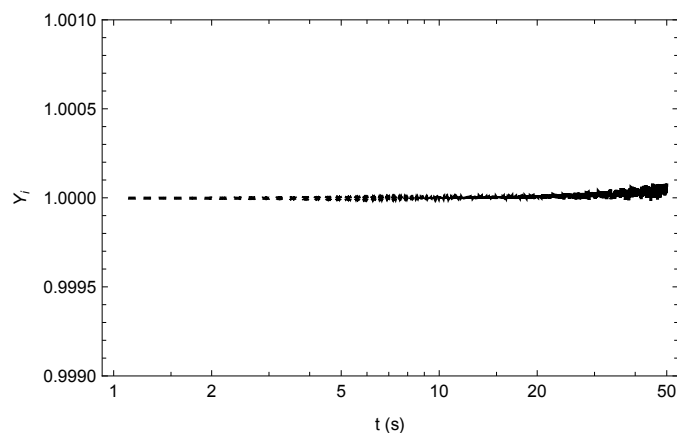
```
LogLogPlot[{YI["Li7"][tv] / YNSE["Li7"], YI["n"][tv], YI["p"][tv], Toft[tv]}],
  _tracé log-log
```

```
{tv, tmiddle*1.1, 1.5}, Frame → True, FrameLabel → {"t (s)", "Yi"},
  _cadre _vrai _étiquette de cadre
```

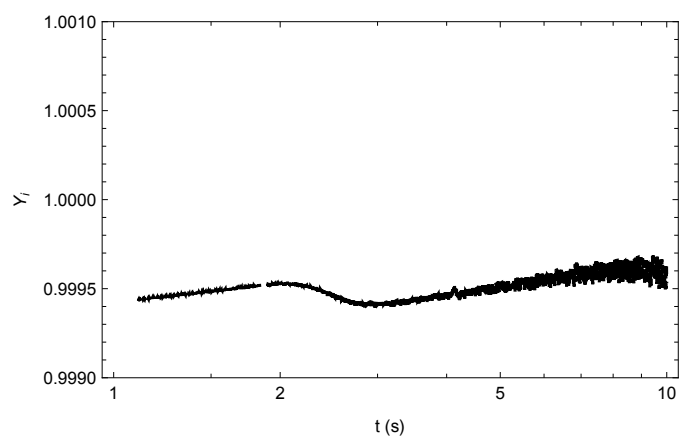
```
PlotStyle → {Black, {Black, Dashed}}, FrameTicks → MyTicks,
  _style de tracé _noir _noir _en tirets _graduations de cadre
```

```
ImagePadding → {{50, 10}, {40, 25}}, PlotRange → {0.999, 1.001} *)
  _garnissage d'image _zone de tracé
```

Out[825]=



Out[826]=



We plot early values together with thermal equilibrium in dashes

In[827]:=

```
MyTickst = {{Automatic, Automatic},
             _automatique _automatique
             {Automatic, {{tofa@a[10^11], "1011K"}, {tofa@a[10^10.5], "1010.5K"},
             _automatique
             {tofa@a[10^10], "1010K"}, {tofa@a[10^9.5], "109.5K"}, {tofa@a[10^9],
             "109K"}, {tofa@a[10^8.5], "108.5K"}, {tofa@a[10^8], "108K"}}}};
```

In[828]:=

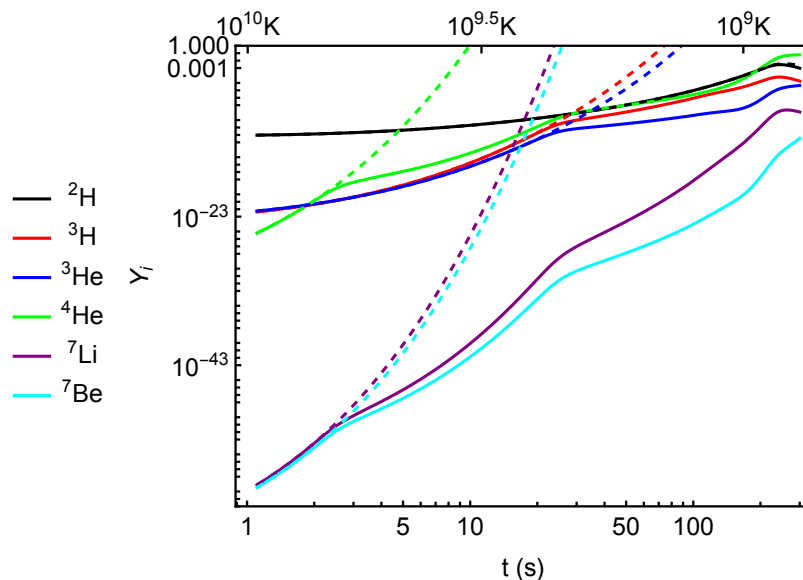
```

PL1 = LogLogPlot[{YI["d"][tv], YI["t"][tv], YI["He3"][tv], YI["a"][tv],
  YI["Li7"][tv], YI["Be7"][tv], YNSE["d", YI["n"][tv], YI["p"][tv], Toft[tv]],
  YNSE["t", YI["n"][tv], YI["p"][tv], Toft[tv]],
  YNSE["He3", YI["n"][tv], YI["p"][tv], Toft[tv]],
  YNSE["a", YI["n"][tv], YI["p"][tv], Toft[tv]],
  YNSE["Li7", YI["n"][tv], YI["p"][tv], Toft[tv]],
  YNSE["Be7", YI["n"][tv], YI["p"][tv], Toft[tv]]},
{tv, tmiddle * 1.1, 300}, Frame → True, FrameLabel → {"t (s)", "Yi"},
  Cadre → True, Vrai → True, Étiquette de cadre → True,
  FrameTicks → MyTickst, LabelStyle → {FontSize → 13},
  graduations de cadre → True, style d'étiquette → True, taille de police de caractères → True,
  FrameStyle → Thickness[0.004], PlotRange → {10-62, 1}, AspectRatio → .8,
  style de cadre → True, épaisseur → True, zone de tracé → True, rapport d'aspect → True,
  PlotStyle → {Black, Red, Blue, Green, Purple, Cyan, {Black, Dashed},
  style de tracé → True, noir → True, rouge → True, bleu → True, vert → True, violet → True, cyan → True, noir → True, en tirets → True,
  {Red, Dashed}, {Blue, Dashed}, {Green, Dashed}, {Purple, Dashed},
  rouge → True, en tirets → True, bleu → True, en tirets → True, vert → True, en tirets → True, violet → True, en tirets → True,
  {Cyan, Dashed}}, (*ImagePadding→{{50,10},{40,25}},*)
  cyan → True, en tirets → True, garnissage d'image → True,
  PlotLegends → Placed[LineLegend[{"2H", "3H", "3He", "4He", "7Li", "7Be"},
  légendes de tracé → True, placé → True, légende de ligne → True,
  LegendLayout → (Grid[#, Frame → None] &)], Left]]
  disposition de légende → True, grille → True, cadre → True, aucun → True, gauche → True,

If[$ResultsPlots,
  si
  Export["Plots/PlotEarlyEquilibrium.pdf", Style[PL1, Magnification → 1], "PDF"];]
  exporte → True, style → True, agrandissement → True, fonction de → True

```

Out[828]=



Neutrons only evolution

In[830]:=

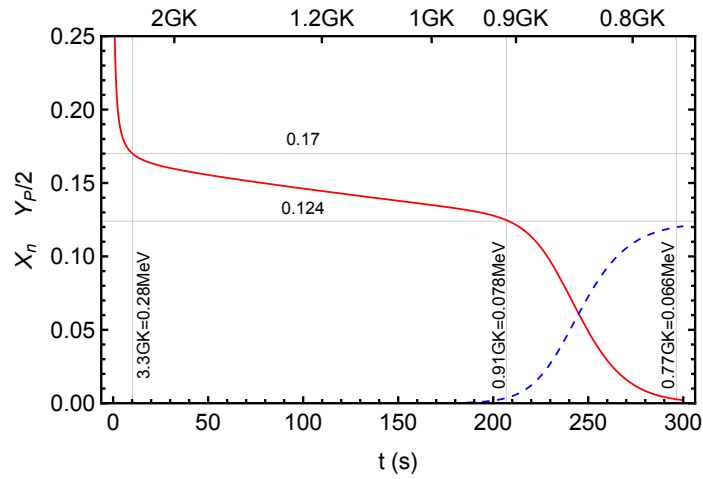
```

YnCoc =
Show[Plot[{YI["n"][tv], Y["WeakInteractionsOnly"]["n"][tv], YI["a"][tv] * 2
mon... tracé
(*, 1/(1+Exp[Q/kB/Toft[tv]]*)}, {tv, t0, 300}, Frame → True,
exponentielle cadre vrai
FrameTicks → {{Automatic, Automatic}, {Automatic, {{tofa@a[10^9], "1GK"},
graduations de cadre automatique automatique automatique
{tofa@a[.8 * 10^9], "0.8GK"}, {tofa@a[.9 * 10^9], "0.9GK"},
{tofa@a[1.2 * 10^9], "1.2GK"}, {tofa@a[2 * 10^9], "2GK"}}}},
FrameStyle → Thickness[0.004], FrameLabel → {"t (s)", "Xn Yp/2"},
style de cadre épaisseur étiquette de cadre
LabelStyle → {FontSize → 12},
style d'étiquette taille de police de caractères
PlotStyle → {{Red, Thickness[0.003]}, {Red, Thickness[0.003], Dotted},
style de tracé rouge épaisseur rouge épaisseur en pointillé
{Blue, Thickness[0.003], Dashed}}, PlotRange → {0, 0.25},
bleu épaisseur en tirets zone de tracé
GridLines → {{{tofa@a[3.3 Giga Kelvin], {Gray, Thickness[0.003]}},
lignes de grille gris épaisseur
{tofa@a[0.91 Giga Kelvin], {Gray, Thickness[0.003]}},
gris épaisseur
{tofa@a[0.77 * 10^9], {Gray, Thickness[0.003]}},
gris épaisseur
{{0.124, {Gray, Thickness[0.003]}}, {0.17, {Gray, Thickness[0.003]}},
gris épaisseur gris épaisseur
Graphics[{Rotate[Text[Style["3.3GK=0.28MeV", FontSize → 9, Black], {16, 0.06}],
graphique fais pivo... texte style taille de police de... noir
90 Degree], Rotate[Text[Style["0.91GK=0.078MeV", FontSize → 9, Black],
degré fais pivo... texte style taille de police de... noir
{203, 0.06}], 90 Degree],
degré
Rotate[Text[
fais pivo... texte
Style["0.77GK=0.066MeV", FontSize → 9, Black], {292, 0.06}], 90 Degree],
style taille de police de... noir degré
Rotate[Text[Style["0.17", FontSize → 9, Black], {100, 0.18}], 0 Degree],
fais pivo... texte style taille de police de... noir degré
Rotate[Text[Style["0.124", FontSize → 9, Black], {100, 0.133}], 0 Degree]]]
fais pivo... texte style taille de police de... noir degré

If[$ResultsPlots,
si
Export["Plots/PlotYnCoc.pdf", Style[YnCoc, Magnification → 1], "PDF"];
exporte style aggrandissement fonction de densité de pri

```

Out[830]=



Main species of the small network

The plots assume that we have used the full nuclear network and not the reduced one. Otherwise some nuclides might not exist and Mathematica should complain ...

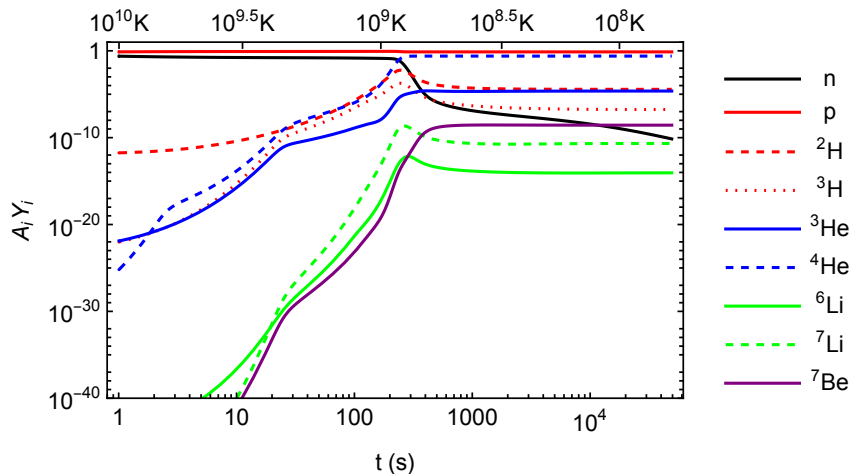
In[832]:=

```

BBNsmall = LogLogPlot[ {YI["n"] [tv], YI["p"] [tv], 2 YI["d"] [tv], 3 YI["t"] [tv],
  tracé log-log
  3 YI["He3"] [tv], 4 YI["a"] [tv], YI["Li6"] [tv], YI["Li7"] [tv], 7 YI["Be7"] [tv] },
  {tv, tmiddle, tend}, Frame → True, FrameStyle → Thickness[0.003],
  cadre vrai style de cadre épaisseur
  FrameLabel → {"t (s)", "AiYi"}, LabelStyle → {FontSize → 12},
  étiquette de cadre style d'étiquette taille de police de caractères
  PlotRange → {10-40, 10}, PlotStyle → {Black, Red, {Red, Dashed},
  zone de tracé style de tracé noir rouge rouge en tirets
    {Red, Dotted}, Blue, {Blue, Dashed}, Green, {Green, Dashed}, Purple},
    rouge en pointillé bleu bleu en tirets vert vert en tirets violet
  FrameTicks → MyTickst, ImagePadding → {{50, 10}, {40, 25}}, PlotLegends →
  graduations de cadre garnissage d'image légendes de tracé
  Placed[LineLegend[{"n", "p", "2H", "3H", "3He", "4He", "6Li", "7Li", "7Be"},
  placé légende de ligne
    LegendLayout → (Grid[#, Frame → None] &)], Right]]
    disposition de légende grille cadre aucun droite

```

Out[832]=



In[833]:=

```

If[$ResultsPlots, Export["Plots/BBNsmall.pdf", BBNsmall, "PDF"];]
si exporte fonction de

```

From Hydrogen to Borron

Custom colors for plots of a given element

In[834]:=

```

ListColor[Color_, n_] := Take[Join[{{Color, Thickness[0.004]}},
  prends joins épaisseur
  Table[{Color, Thickness[0.004], Dashing[{i] 0.01}], {i, 1, 3}],
  table épaisseur style de tirets
  Table[{Color, Thickness[0.004],
  table épaisseur
    Dashing[{0, 0.015 * i, {i] 0.015, i 0.015}]], {i, 1, 4}]], n]
    style de tirets

```

In[835]:=

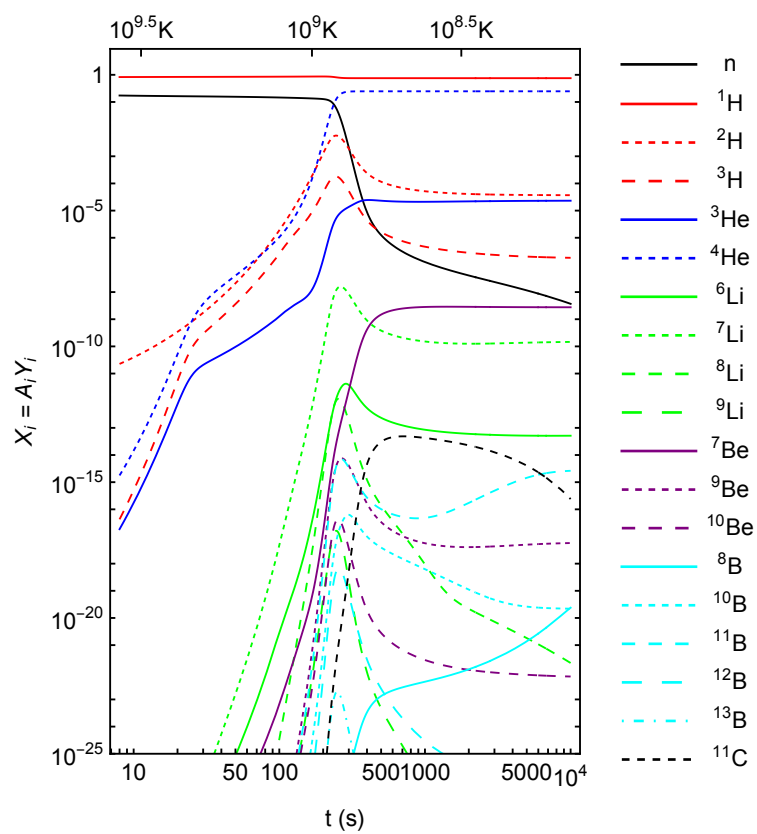
```

BBNsmall2 =
LogLogPlot[{XI["n"][tv], XI["p"][tv], XI["d"][tv], XI["t"][tv], XI["He3"][tv],
i = AiYi"}, LabelStyle → {FontSize → 12},
_cadre _vrai _étiquette de cadre _style d'étiquette _taille de police de caractères
FrameStyle → Thickness[0.004], PlotRange → {10^-25, 9},
_style de cadre _épaisseur _zone de tracé
PlotStyle → Join[ListColor[Black, 1], ListColor[Red, 3],
_style de tracé _joins _noir _rouge
    ListColor[Blue, 2], ListColor[Green, 4], ListColor[Purple, 3],
_bleu _vert _violet
    ListColor[Cyan, 5], {{Black, Thickness[0.004], Dashed}}},
_cyan _noir _épaisseur _en tirets
AspectRatio → 1.5, FrameTicks → MyTickst, ImagePadding → {{50, 10}, {40, 25}},
_rapport d'aspect _graduations de cadre _garnissage d'image
PlotLegends → Placed[LineLegend[{"n", "1H", "2H", "3H", "3He", "4He", "6Li",
_légendes de tracé _placé _légende de ligne
    "7Li", "8Li", "9Li", "7Be", "9Be", "10Be", "8B", "10B", "11B", "12B",
    "13B", "11C"}, LegendLayout → (Grid[#, Frame → None] &)], Right]]
_con... _disposition de légende _grille _cadre _aucun _droite

If[$ResultsPlots,
_si
Export["Plots/PlotBBNLight.pdf", Style[BBNsmall2, Magnification → 1], "PDF"];]
_exporte _style _agrandissement _fonction de

```

Out[835]=



From Carbon to Oxygen

In[837]:=

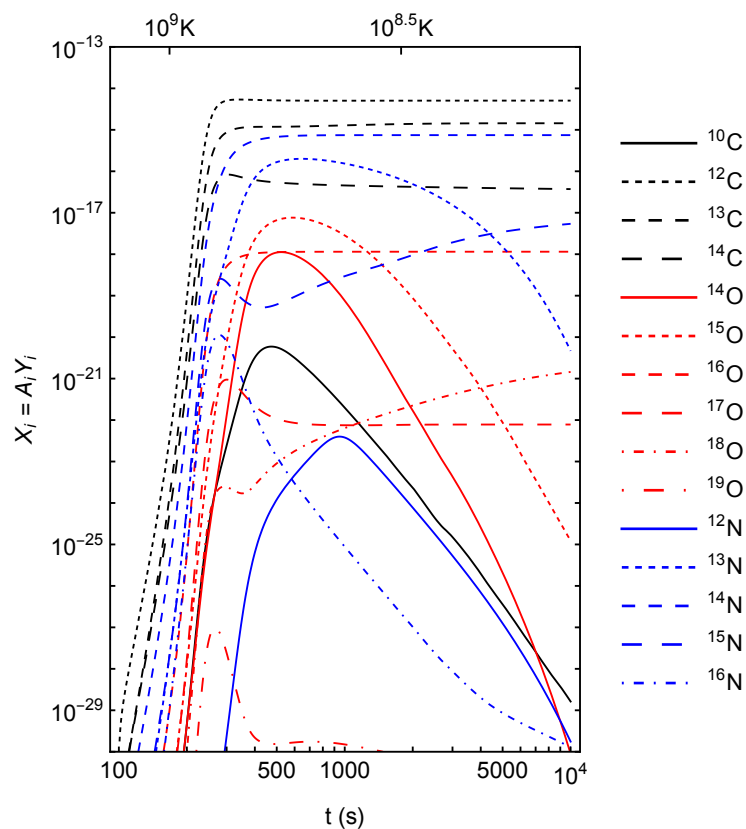
```

If[Not[$ReducedNetwork],
  si [négation]
  BBNCNO = LogLogPlot[{XI["C10"][tv], XI["C12"][tv], XI["C13"][tv],
    [tracé log-log]
    XI["C14"][tv], XI["O14"][tv], XI["O15"][tv], XI["O16"][tv],
    XI["O17"][tv], XI["O18"][tv], XI["O19"][tv], XI["N12"][tv],
    XI["N13"][tv], XI["N14"][tv], XI["N15"][tv], XI["N16"][tv]},
    {tv, 1.01 t18, 10^4}, Frame → True, FrameLabel → {"t (s)", "Xi = AiYi"},
    [cadre] [vrai] [étiquette de cadre]
    LabelStyle → {FontSize → 12}, FrameStyle → Thickness[0.004],
    [style d'étiquette] [taille de police de ca...] [style de cadre] [épaisseur]
    PlotRange → {10^-30, 10^-13}, PlotStyle → Join[ListColor[Black, 4],
    [zone de tracé] [style de tracé] [joins] [noir]
    ListColor[Red, 6], ListColor[Blue, 5], ListColor[Purple, 1]],
    [rouge] [bleu] [violet]
    AspectRatio → 1.5, FrameTicks → MyTickst, ImagePadding → {{50, 10}, {40, 25}},
    [rapport d'aspect] [graduations de cadre] [garnissage d'image]
    PlotLegends → Placed[LineLegend[{10C", 12C", 13C", 14C", 14O",
    [légendes de tracé] [placé] [légende de ligne] [consta... [consta... [consta... [consta... [notation O
    15O", 16O", 17O", 18O", 19O", 12N", 13N", 14N", 15N", 16N"}],
    [notati... [notati... [notati... [notati... [notati... [valeur... [valeur... [valeur... [valeur... [valeur numérique]
    LegendLayout → (Grid[#, Frame → None] &)], Right]]
    [disposition de légende] [grille] [cadre] [aucun] [droite]
  ]

If[$ResultsPlots && Not[$ReducedNetwork],
  si [négation]
  Export["Plots/PlotBBNHeavy.pdf", Style[BBNCNO, Magnification → 1], "PDF"];
  [exporte] [style] [agrandissement] [fonction de de

```

Out[837]=



Final abundances

Main results

Final time at $T = 4 \times 10^7$ K in seconds.

In[839]:=

```
tofa[a[4 * 10^7]]
```

Out[839]=

```
110 753.3
```

Standard abundances as reported in most BBN papers.

Note the definition $Y_P = 4 Y_{\text{He4}}$. Since the atomic mass

of Helium is not exactly 4 this is not exactly Helium mass abundance

In[840]:=

```
MyGrid@Transpose[
  _transpose
  {{"H", "YP=4YHe", "D/H x 105", "3He/H x 105", "T/H x 108", "(3He+T)/H x 105",
    _dérivée d
    "7Li/H x 1011", "7Be/H x 1010", "(7Li+7Be)/H x 1010", "6Li/H x 1014",
    "9Be/H x 1019", "10B/H x 1021", "11B/H x 1016", "CNO/H x 1016", "Neff"},
  {Yf["p"], 4 Yf["a"], YfH["d"] 105, YfH["He3"] 105, YfH["t"] 108,
    (YfH["t"] + YfH["He3"]) 105, YfH["Li7"] 1011, YfH["Be7"] 1010,
    (YfH["Li7"] + YfH["Be7"]) 1010, YfH["Li6"] 1014, YfH["Be9"] 1019,
    YfH["B10"] 1021, YfH["B11"] 1016, YfH["CNO"] 1016, Neff}}]
```

Out[840]=

H	0.75272872
Y _P =4Y _{He}	0.24721111
D/H x 10 ⁵	2.4376915
³ He/H x 10 ⁵	1.0315675
T/H x 10 ⁸	7.7610476
(³ He+T)/H x 10 ⁵	1.0393285
⁷ Li/H x 10 ¹¹	2.8453517
⁷ Be/H x 10 ¹⁰	5.2245122
(⁷ Li+ ⁷ Be)/H x 10 ¹⁰	5.5090474
⁶ Li/H x 10 ¹⁴	1.1826908
⁹ Be/H x 10 ¹⁹	8.7824993
¹⁰ B/H x 10 ²¹	2.8722753
¹¹ B/H x 10 ¹⁶	3.2546945
CNO/H x 10 ¹⁶	7.8286982
N _{eff}	3.0439773

All final abundances

Final number fractions

In[841]:=

```
MyGrid[{#, Yf[#]} & /@ ShortNames]
```

Out[841]=

n	$7.1701401 \times 10^{-11}$
p	0.75272872
d	0.000018349204
t	5.8419634×10^{-8}
He3	7.7649048×10^{-6}
a	0.061802778
He6	$4.5136388 \times 10^{-44}$
Li6	$8.9024536 \times 10^{-15}$
Li7	$2.1417779 \times 10^{-11}$
Li8	$5.0026583 \times 10^{-26}$
Li9	$3.3456891 \times 10^{-41}$
Be7	$3.9326404 \times 10^{-10}$
Be9	$6.6108394 \times 10^{-19}$

Be10	$6.4630222 \times 10^{-24}$
Be11	$2.4624758 \times 10^{-38}$
Be12	$1.0425426 \times 10^{-55}$
B8	1.028768×10^{-23}
B10	$2.1620441 \times 10^{-21}$
B11	2.449902×10^{-16}
B12	$1.8082691 \times 10^{-32}$
B13	$1.5965455 \times 10^{-48}$
B14	$2.5432041 \times 10^{-63}$
B15	$1.4381908 \times 10^{-81}$
C9	$2.0549347 \times 10^{-40}$
C10	$1.0668262 \times 10^{-36}$
C11	$4.6798357 \times 10^{-27}$
C12	$4.2218219 \times 10^{-16}$
C13	$1.1078801 \times 10^{-16}$
C14	$2.4272139 \times 10^{-18}$
C15	$5.3413991 \times 10^{-34}$
C16	$2.0248583 \times 10^{-49}$
N12	$1.5584583 \times 10^{-45}$
N13	$1.4566448 \times 10^{-28}$
N14	$5.3228791 \times 10^{-17}$
N15	$5.9046213 \times 10^{-19}$
N16	$3.1830736 \times 10^{-34}$
N17	$2.5253827 \times 10^{-44}$
O13	$-1.0768582 \times 10^{-58}$
O14	$3.2282925 \times 10^{-43}$
O15	$6.6103205 \times 10^{-32}$
O16	$7.1794239 \times 10^{-20}$
O17	4.591403×10^{-24}
O18	$1.3352048 \times 10^{-22}$
O19	$6.6872229 \times 10^{-36}$
O20	$6.3313609 \times 10^{-49}$
F17	$5.5372415 \times 10^{-36}$
F18	$8.6892316 \times 10^{-25}$
F19	$6.3544502 \times 10^{-26}$
F20	$3.8592489 \times 10^{-38}$
Ne18	$3.7380244 \times 10^{-49}$
Ne19	$1.3098503 \times 10^{-41}$
Ne20	$6.8383734 \times 10^{-28}$
Ne21	$5.3400298 \times 10^{-30}$
Ne22	8.856846×10^{-32}
Ne23	$-2.9924608 \times 10^{-74}$
Na20	$8.5616556 \times 10^{-64}$
Na21	$-8.6817923 \times 10^{-78}$
Na22	$1.7735319 \times 10^{-36}$

Table in mass fraction instead of number fraction. So we use $X_i = A_i Y_i$.

In[842]:=

```
TableXMass = {#, Xf[#]} & /@ Join[ShortNames, {"CNO"}];
```

[Joins](#)

```
MyGrid[TableXMass]
```

Out[843]=

n	$7.1701401 \times 10^{-11}$
p	0.75272872
d	0.000036698408
t	1.752589×10^{-7}
He3	0.000023294714
a	0.24721111
He6	$2.7081833 \times 10^{-43}$
Li6	$5.3414721 \times 10^{-14}$
Li7	$1.4992446 \times 10^{-10}$
Li8	$4.0021266 \times 10^{-25}$
Li9	$3.0111202 \times 10^{-40}$
Be7	2.7528482×10^{-9}
Be9	$5.9497555 \times 10^{-18}$
Be10	$6.4630222 \times 10^{-23}$
Be11	$2.7087234 \times 10^{-37}$
Be12	$1.2510511 \times 10^{-54}$
B8	$8.2301442 \times 10^{-23}$
B10	$2.1620441 \times 10^{-20}$
B11	$2.6948922 \times 10^{-15}$
B12	$2.1699229 \times 10^{-31}$
B13	$2.0755091 \times 10^{-47}$
B14	$3.5604858 \times 10^{-62}$
B15	$2.1572863 \times 10^{-80}$
C9	$1.8494413 \times 10^{-39}$
C10	$1.0668262 \times 10^{-35}$
C11	$5.1478193 \times 10^{-26}$
C12	$5.0661863 \times 10^{-15}$
C13	$1.4402441 \times 10^{-15}$
C14	$3.3980995 \times 10^{-17}$
C15	$8.0120986 \times 10^{-33}$
C16	$3.2397733 \times 10^{-48}$
N12	1.87015×10^{-44}
N13	$1.8936382 \times 10^{-27}$
N14	$7.4520307 \times 10^{-16}$
N15	8.856932×10^{-18}
N16	$5.0929178 \times 10^{-33}$
N17	$4.2931506 \times 10^{-43}$
O13	$-1.3999156 \times 10^{-57}$
O14	$4.5196095 \times 10^{-42}$
O15	$9.9154807 \times 10^{-31}$

O16	$1.1487078 \times 10^{-18}$
O17	7.805385×10^{-23}
O18	$2.4033686 \times 10^{-21}$
O19	$1.2705724 \times 10^{-34}$
O20	$1.2662722 \times 10^{-47}$
F17	$9.4133105 \times 10^{-35}$
F18	$1.5640617 \times 10^{-23}$
F19	$1.2073455 \times 10^{-24}$
F20	$7.7184977 \times 10^{-37}$
Ne18	6.728444×10^{-48}
Ne19	$2.4887155 \times 10^{-40}$
Ne20	$1.3676747 \times 10^{-26}$
Ne21	$1.1214063 \times 10^{-28}$
Ne22	$1.9485061 \times 10^{-30}$
Ne23	$-6.8826598 \times 10^{-73}$
Na20	$1.7123311 \times 10^{-62}$
Na21	$-1.8231764 \times 10^{-76}$
Na22	$3.9017702 \times 10^{-35}$
Na23	$2.5033817 \times 10^{-36}$
CNO	$7.2956226 \times 10^{-15}$

Uncomment if you want to output results in external file

In[844]:=

```
(*SetDirectory[NotebookDirectory[]]
  |alloue répertoire |répertoire de notebook
Export["TableXMass.dat",TableXMass,"TSV"]*)
|exporte
```

In[845]:=

```
(*TableXMassT8={#,XI[#] [tofa[a[10^8]]]}&/@Join[ShortNames,{"CNO"}]*)
|joins
```

In[846]:=

```
(*SetDirectory[NotebookDirectory[]]
  |alloue répertoire |répertoire de notebook
Export["TableXMassT8.dat",TableXMassT8,"TSV"]*)
|exporte
```

Tools for Monte-Carlo on nuclear rates

We gather some tools for Monte-Carlo estimation of uncertainties from nuclear rates.

Some Examples are provided in the Example folder.

There are 3 booleans which control what variables are varied randomly.

Nuclear rates

Neutron lifetime

And possibly baryons abundance according to [Planck 2015] results when this is interesting to vary it as well.

In[847]:=

```

$ParallelBool = True;
               _vrai
$Randomneutron = True;
               _vrai
$Randomh2Ob = True;
               _vrai

```

Initialize Kernels (if parallelization this is called. It just distributes definitions)

In[850]:=

```

InitializeKernels := (
  LaunchKernels[];
  _lance noyaux
  Print["Number of Kernels ", $KernelCount];
  _imprime _nombre _noyaux _compte noyaux
  DistributeDefinitions[ReshapedTabulatedReactions,
  _distribue définitions
    ListReactionsUpToChosenMass, LoadRates, DefineEquations,
    SystemEquationsHT, SystemEquationsMT, SystemEquationsLT,
    LoadRates, DY, DY18, DYOnlyPEN, LbarnT0p, LnT0p];
);

```

We define a function which launches the Monte-Carlo on subKernels and collects the results

In[851]:=

```

RunPRIMATMonteCarlo[number_] := Module[{res, time, Abundances,
  _module
  mytabfunctions, sss, RandomVariables, CosmoParametersList},

  Print["Reaction file used is : ", TabulatedReactionsFile];
  _imprime
  Print[StringJoin@@ {# <> " " & /@ SafeImport[TabulatedReactionsFile][[1]]}];
  _imprime _joins chaînes de caractères

  Print[MyGrid[Join[{"Reaction Number", "Reaction Name",
  _imprime _joins _nombre
    "Initial species", "Final Species", "Reference"}},
    PadLeftNumberRowFromZero[Drop[#, {4}] & /@
    _laisse tomber
    (NiceDisplayReaction /@ ListReactionsUpToChosenMass)]]]];

  If[number > 1, Print["Running a Monte-Carlo with ", number, " points."];];
  _si _imprime
  Off[CompiledFunction::cfta];
  _dé... _fonction compilée
  mytabfunctions = If[$ParallelBool, ParallelTable, Table];
  _si _table en parallèle _table
  If[$ParallelBool, InitializeKernels;
  _si
    ParallelEvaluate[$HistoryLength = 0;]];
  _évalue en parallèle _longueur d'historique

```

```

(* We always use the same seed so that we always use the same
sequence of random number as advocated in [Cyburt et al. 2015].*)

res = mytabfunctions[
  $Seed := i;
  (* We use a different seed so that for each MC
point we have a different sequence of reaction rates *)
  InitializeRandom[$Seed];
  (* We restart our random list from the beginning *)

  h2Ωb0 = Meanh2Ωb0 + If[$Randomh2Ωb, σh2Ωb0 NormalRealisation, 0];
  τneutron =
    Meanτneutron + If[$Randomτneutron, στneutron NormalRealisation, 0];

  CosmoParametersList = {h2Ωb0, τneutron};

  time = AbsoluteTiming[RunNumericalIntegrals][[1]];

  RandomVariables = Rest@ListReactionsUpToChosenMass[All, 4];

  Share[];

  Print["Iteration ", i, " Memory usage = ",
    MemoryInUse[], " time = ", time, " Kernel : ", $KernelID];

  Abundances = YPeriodTime["LT"][tend];
  ClearSystemCache[];

  If[$Verbose, Print[Abundances(*, " ", RandomVariables*)]];

  {Abundances, RandomVariables, CosmoParametersList}, {i, 1, number}];

If[$ParallelBool, CloseKernels[]];

h2Ωb0 = Meanh2Ωb0;
τneutron = Meanτneutron;
MC = res[[All, 1]];

RV = res[[All, 2]];

Cosmo = res[[All, 3]];

res];

RunPRIMAT := RunPRIMATMonteCarlo[1];

```

The results of RunPRIMAT are gathered in the files MC (abundances) RV (rates variations) Cosmo (List of cosmological parameters, limited to baryons abundances and neutron lifetime)

We define functions to dump the results of a Monte-Carlo in a file and also the converse to load the result so as to analyze and use it.

In[853]:=

```
Clear[LoadMC, DumpMC]
_efface
```

In[854]:=

```
DumpMC[File_String] := (
_chaîne de caractères
  Print["Exporting ", "MonteCarlo/MC" <> File <> ".dat"];
_imprime _fichier
  Export["MonteCarlo/MC" <> File <> ".dat", MC];
_exporte _fichier

  Print["Exporting ", "MonteCarlo/RV" <> File <> ".dat"];
_imprime _fichier
  Export["MonteCarlo/RV" <> File <> ".dat", RV];
_exporte _fichier

  Print["Exporting ", "MonteCarlo/Cosmo" <> File <> ".dat"];
_imprime _fichier
  Export["MonteCarlo/Cosmo" <> File <> ".dat", Cosmo]);
_exporte _fichier

LoadMC[File_String] := (
_chaîne de caractères
  MCfile = "MonteCarlo/MC" <> File <> ".dat";
_fichier
  RVfile = "MonteCarlo/RV" <> File <> ".dat";
_fichier
  Cosmofile = "MonteCarlo/Cosmo" <> File <> ".dat";
_fichier

  MC = Import[MCfile];
_importe
  RV = Import[RVfile];
_importe
  Cosmo = Import[Cosmofile];
_importe
  TMC = Transpose[MC];
_transpose
);
```

Whenever a Monte-Carlo is finished, or loaded, we can obtain the table of values for a given element, or a given reaction

In[856]:=

```
ElementColumn[el_] := MC[All, KeyVal[el]];
                                     |tout
ReactionColumn[el_] := RV[All, KeyNuclearReaction[el]];
                                     |tout
h2Ωb0List := Cosmo[All, 1];
                                     |tout
τneutronList := Cosmo[All, 2];
                                     |tout
```