Manual and Code to Compute Solutions and to Generate Figures for Flat Friedmann Differential Equation

Harri Ehtamo
Aalto University
harri.ehtamo@aalto.fi

Lauri Jokinen
Aalto University
lauri.jokinen@iki.fi

May 28, 2024

Contents

1 Introduction

This manual describes the code that can be used to compute solutions and to generate figures for flat Friedmann differential equation with various parameter combinations. The underlying mathematical model has been developed by Harri Ehtamo in [1] in detail. The manual and the code are designed by Harri Ehtamo and Lauri Jokinen, and they are available on GitHub [2]. We use Matlab R2021b.

The model includes a time-dependent density parameter arising as a consequence of a hypothetical radiation field. It is called Λ-radiation, and it is related to the cosmological constant term.

In Section 2 we define the Friedmann equation together with the Λ-radiation term. In Section 3 we define the parameters for the model and show how to compute unknown parameters from the flatness condition, together with other conditions. In particular, we fix the time unit of our model by defining a time unit for the Hubble constant appropriately. In Section 4 we discuss the computation of each of the 12 generated figures separately.

2 System of differential equations

Define, $a = a(t)$, and define T by $a(T) = 1$. The ΛR-model is defined with the following equations,

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{H_T^2}{a^3}\Omega^{B+D} + \frac{H_T^2}{a^3}\alpha\Omega_t^{\Lambda\mathrm{R}} + H_T^2\Omega^\Lambda, \tag{1}$$

$$\Omega_t^{\Lambda\mathrm{R}} \triangleq \Omega^{\Lambda\mathrm{R}}(t) = \frac{H_T\sqrt{\Omega^\Lambda}}{a(t)}\int_0^t a(s)\, e^{-s\, H_T\sqrt{\Omega^\Lambda}}\,\mathrm{d}s, \tag{2}$$

with the initial condition, $a(0) = a_0 > 0$. Here, $\Omega^{B+D} = \Omega^B + \Omega^D$, with $\Omega^B \geq 0$, $\Omega^D \geq 0$ constants; and $H_T > 0$, $\Omega^\Lambda \geq 0$, $\alpha \geq 0$ constants. (Pure) Friedmann model corresponds to the case $\alpha = 0$. Equation (1) is equivalent to,

$$\dot{a} = H_T\sqrt{\frac{1}{a}\Omega^{B+D} + \frac{1}{a}\alpha\Omega_t^{\Lambda\mathrm{R}} + a^2\Omega^\Lambda}.$$

Then, define the integral in Equation (2) as, $b = b(t)$, which results in another ODE,

$$b = \int_0^t a(s)\, e^{-s\, H_T\sqrt{\Omega^\Lambda}}\,\mathrm{d}s \quad \Longleftrightarrow \quad \begin{cases} \dot{b} = a\, e^{-t\, H_T\sqrt{\Omega^\Lambda}}, \\ b(0) = 0, \end{cases} \tag{3}$$

and,

$$\Omega_t^{\Lambda\mathrm{R}} = \frac{H_T\sqrt{\Omega^\Lambda}}{a(t)}b(t).$$

Now, we have an ODE system which can be solved with, e.g., Matlab's ODE solver,

$$\begin{aligned} &\dot{a} = H_T\sqrt{\frac{1}{a}\Omega^{B+D} + \frac{1}{a}\alpha\left(\frac{H_T\sqrt{\Omega^\Lambda}}{a}b\right) + a^2\Omega^\Lambda}; \\ &\dot{b} = a\, e^{-t\, H_T\sqrt{\Omega^\Lambda}}; \\ &a(0) = a_0,\ b(0) = 0,\ t \geq 0. \end{aligned} \tag{4}$$

In the figures, we shift the time axis, s.t., the present time is at zero. This is done by replacing the time variable, t, by $t' = t - T$, where $a(T) = 1$. But for the purpose of this manual, it is sufficient to use t instead of t'.

Above, $a(t)$, $t \geq 0$, is called the scale factor of the model. The parameters Ω^B, Ω^D and Ω^Λ are the density parameters for baryonic matter, cold dark matter, and cosmological constant; the cosmological constant is defined by $\Lambda/3 \triangleq H_T^2 \Omega^\Lambda$. The time-dependent density parameter $\Omega_t^{\Lambda\mathrm{R}}$ arises as a consequence of a radiation field and it is called Λ-radiation; the constant α is an extra parameter related to the amount of Λ-radiation. The constant H_T is called the Hubble constant, and the Hubble parameter entering, e.g., in Figure 12, is defined by $H_t \triangleq H(t) = \dot{a}(t)/a(t)$, $t \geq 0$. Note that, since $a(T) = 1$, then $H_T = \dot{a}(T)$.

2.1 Solving the ODE system

We shall next study the system with respect to the initial value $a_0 > 0$. Suppose we want to put $a_0 = 0$, and $b_0 \triangleq b(0) = 0$, but the equation for $\dot{a}$ is singular at $t = 0$, as in our case. Nevertheless, Matlab's stiff ODE solver `ode23s` is able to solve the ODE with very small a_0. We shall use $a_0 = 10^{-16}$ which provides enough accuracy for our purposes.

Let us study this approximation carefully, since the system *may* behave rather differently depending on the initial values for a_0, b_0, t_0. So, we simulate the system with randomly sampled initial values. In the Figure below, we have random initial values under different upper bounds. We see that the curves, including the black curve, seem to converge as we approach the initial value $a_0 = b_0 = t_0 = 0$. This indicates that a limit curve exists and the above approximation, $a_0 = 10^{-16}$, $b_0 = t_0 = 0$, should be adequate.

The solver `ode23s` is versatile and provides various useful options. We use these options to increase the precision of the solver, see the code for details. The function `ode23s` also provides a framework for finding user-defined *events* during integration. The solver then attempts to find these events with a required precision. As an example, define two such events.

Suppose first, we want to solve T from the equation $a(T) = 1$. This equation is trivial to define as an event. Then, in Figure 3, we want to find the maximum point of $\Omega_t^{\Lambda\mathrm{R}}$. This is found by solving the equation,

$$\frac{d}{dt}\Omega_t^{\Lambda\mathrm{R}} = 0 \iff \frac{d}{dt}\left(\frac{b}{a}\right) = 0 \iff a\,\dot{b} - b\,\dot{a} = 0, \tag{5}$$

which we define as an event.

3 Defining parameters

The ODE system has parameters H_T, Ω^B, Ω^D, Ω^Λ and α. Here, we provide different parameter definitions which are then later used in various Figures. To numerically solve equations in the code, we use Matlab's algorithm `fsolve`, and for unconstrained optimization we use `fminunc`. We increase the precision of these algorithms by using certain options, which are visible in the code.

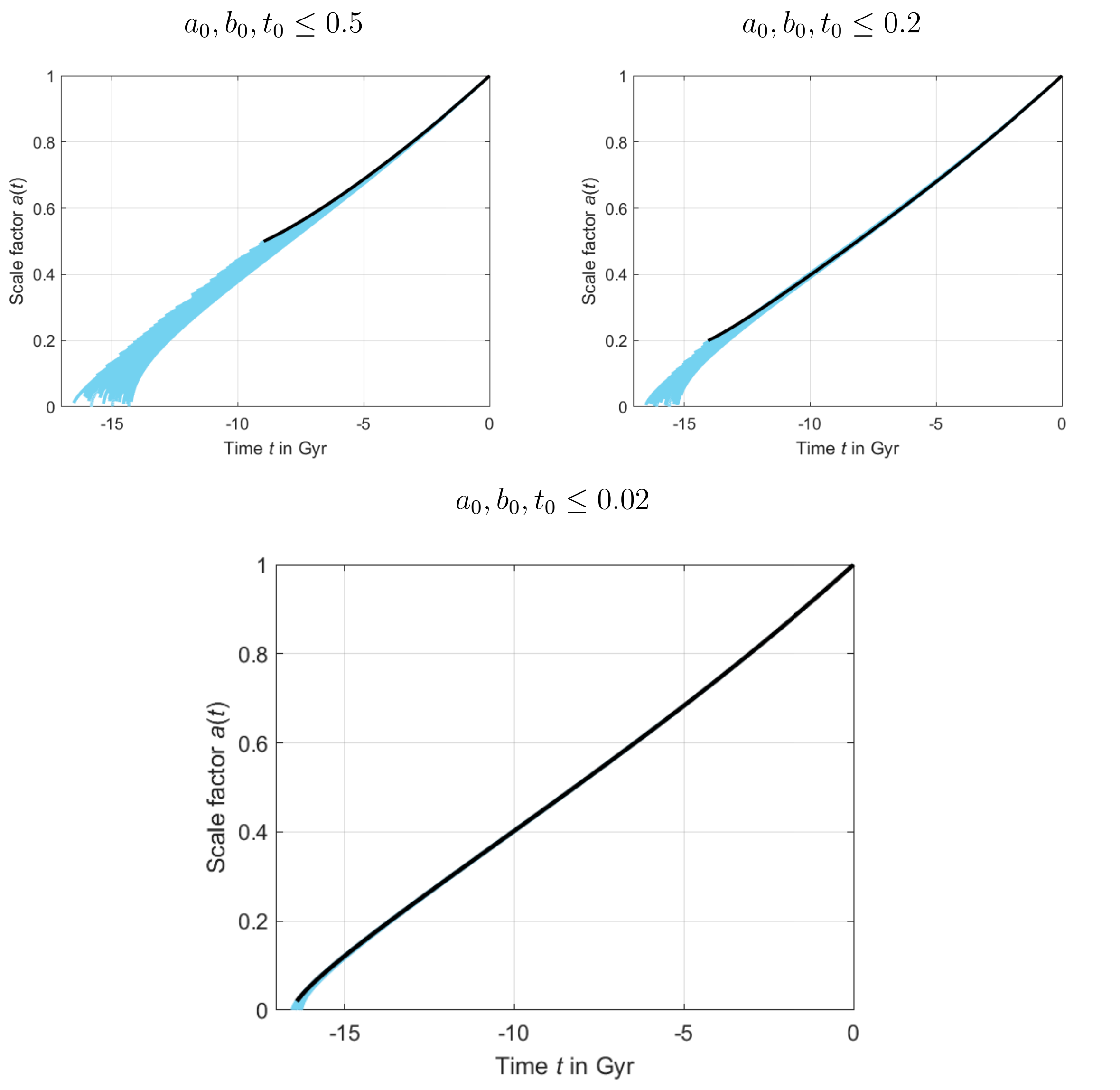

Figure. The ODE system solved with random positive initial conditions, namely, (a_0, b_0, t_0). The curve in black color denotes the largest a_0 allowed by the constraint, and $b_0 = t_0 = 0$. The Figures are produced with parameters, $\Omega^{B+D} = 0.049$, $\alpha = 1$, $\Omega^{\Lambda} = 0.6881$, and, $H_T = 0.0688$.

3.1 Unit conversion for H_T

The ODE system's only parameter whose unit depends on time is H_T, and in the figures, we use the time unit of Gyr, i.e., 10^9 years. Since the value of H_T is given in unit kms^{-1}Mpc^{-1}, we convert it to Gyr^{-1}, where Gyr is 10^9 years. With a (Julian) year of 365.25 days $= 3.15576 \cdot 10^7$ seconds, and using 2012 and 2015 IAU exact definitions for astronomical unit, 1 au $= 1.495978707 \cdot 10^{11}$ m, and for parcek, 1 pc $= 6.48/\pi \cdot 10^5$ au [3], we have an exactly defined conversion factor for the unit conversion (shown below). As an example, below we convert $H_T = 67.26$ kms^{-1}Mpc^{-1}:

```
convertion_factor = 0.001 * 3.15576 / (1.495978707 * 6.48 / pi);
%%%               = 0.001022712165045695
H_T = 67.26 * convertion_factor;
%%% = 0.06878762022097345 (in unit 1/Gyr)
```

This value for H_T is used in Figures 1-10. In Figures 11-12, we also use other values for H_T.

3.2 Flatness equation

This section describes the function `flatness_solve_Omega_L` in the code. Since for given H_T the equation $H_T = \dot{a}(T)$, or $a(T) = 1$, should hold, then the flatness equation,

$$\Omega^{B+D} + \Omega^{\Lambda} + \alpha \Omega_T^{\Lambda\mathrm{R}} = 1, \tag{6}$$

should hold, that for given Ω^{B+D}, α only depends on the parameter Ω^{Λ}. This equation for Ω^{Λ} is solved iteratively by using a suitable iteration method. Note, that at every iteration step, the equations (4) are solved with the current value of Ω^{Λ} to get $\Omega_T^{\Lambda\mathrm{R}} \triangleq \Omega^{\Lambda\mathrm{R}}(T)$, and the left-hand side of (6).

E.g., for parameter values $\alpha = 1$, $\Omega^B = 0.049$ and $\Omega^D = 0$, the flatness equation implies, $\Omega^{\Lambda} \approx 0.6881$, $\Omega_T^{\Lambda\mathrm{R}} \approx 0.2629$.

3.3 Benchmark model's parameters

In the benchmark F-model, in the figures F-model b, the parameters are,

$$\Omega^B = 0.049, \quad \Omega^D = 0.268, \quad \Omega^{\Lambda} = 0.683, \quad \alpha = 0.$$

In other (pure) F-models, i.e., in models with $\alpha = 0$, also other Ω^B, Ω^D and Ω^{Λ} parameters are used. Note, that for $t = T$, $a(T) = 1$, $H_T = \dot{a}(T)$, the flatness condition $\Omega^{B+D} + \Omega^{\Lambda} = 1$ should hold for the parameters. This model implies $T^b \approx 13.800$ Gyrs.

3.4 Optimal Ω^B

This section describes the function `optimal_Omega_B`. Fix $\alpha = 1$ and $\Omega^D = 1$, which implies the following flatness equation, $\Omega^B + \Omega^{\Lambda} + \Omega_t^{\Lambda\mathrm{R}} = 1$. We then solve the following maximization problem,

$$\max_{\Omega^B} \Omega_T^{\Lambda\mathrm{R}},$$
$$\text{s.t.} \qquad \Omega^B + \Omega^{\Lambda} + \Omega_T^{\Lambda\mathrm{R}} = 1,$$

as in Figure 6. We use Matlab's unconstrained optimization algorithm, so the constraint needs to be satisfied at each iteration, i.e., we solve the flatness equation (6) as in 3.2 at each iteration step of the optimization method. This solving method is quite slow due to the layered iterations, but it is sufficient and accurate for the purpose.

The solution to this problem is $\Omega_{\mathrm{opt}}^B \approx 0.0458$, and $\Omega_{T,\mathrm{opt}}^{\Lambda\mathrm{R}} \approx 0.2629$.

3.5 Age-optimal Ω^D and α: case 1

E.g., in Figure 8, we have two curves with $\alpha \Omega_T^{\Lambda\mathrm{R}} = c$, $c = 0.06$, or 0.16 and the age of the universe is matched to the benchmark model's age, namely, $T^b \approx 13.800$ Gyrs, see definition in 3.3. In addition, $\Omega^B = 0.049$ is fixed. In mathematical terms, we find the parameters Ω^D, α, and Ω^Λ so that the following system of equations,

$$\begin{aligned}\alpha\Omega_T^{\Lambda\mathrm{R}} &= c,\\ T &= T^b,\\ \Omega^{B+D} + \Omega^\Lambda + \alpha\Omega_T^{\Lambda\mathrm{R}} &= 1,\end{aligned}$$

is satisfied.

3.6 Age-optimal Ω^D and α: case 2

This case describes, e.g., the optimal ΛR-model in Figure 9, and the function `optimal_Omega_D_and_alpha` in the code. We aim to find a model with the following property,

$$\frac{\Omega^D + \alpha\Omega_T^{\Lambda\mathrm{R}}}{\Omega^B} = \frac{\Omega_{T,\mathrm{opt}}^{\Lambda\mathrm{R}}}{\Omega_{\mathrm{opt}}^B} \approx 5.7380,$$

where $\Omega_{T,\mathrm{opt}}^{\Lambda\mathrm{R}}$, Ω_{opt}^B are as in 3.4. As described above, we match the age of the universe to the benchmark model's age and fix $\Omega^B = 0.049$. In mathematical terms, we solve the parameters Ω^D and α from the following system,

$$\begin{aligned}\frac{\Omega^D + \alpha\Omega_T^{\Lambda\mathrm{R}}}{\Omega^B} &= \frac{\Omega_{T,\mathrm{opt}}^{\Lambda\mathrm{R}}}{\Omega_{\mathrm{opt}}^B},\\ T &= T^b,\\ \Omega^{B+D} + \Omega^\Lambda + \alpha\Omega_T^{\Lambda\mathrm{R}} &= 1,\end{aligned}$$

when $a(T) = 1$. We further replace the third equation by combining the first and third equations. This yields an equivalent system,

$$\begin{aligned}\frac{\Omega^D + \alpha\Omega_T^{\Lambda\mathrm{R}}}{\Omega^B} &= \frac{\Omega_{T,\mathrm{opt}}^{\Lambda\mathrm{R}}}{\Omega_{\mathrm{opt}}^B},\\ T &= T^b,\\ \Omega^\Lambda &= 1 - \left(\frac{\Omega_{\mathrm{opt}}^{\Lambda\mathrm{R}}}{\Omega_{\mathrm{opt}}^B} + 1\right)\Omega^B.\end{aligned}$$

With this system, we no longer need to solve the flatness equation iteratively, since the value of Ω^Λ is fixed. The solution to this system is , $\Omega^D \approx 0.2589$, $\alpha \approx 0.0832$, and $\alpha\Omega_T^{\Lambda\mathrm{R}} \approx 0.0223$.

4 Figures

Here, we describe the contents of each figure separately. The figures are given in the Appendix.

4.1 Figures 1 and 2

In general, to compute the solution trajectories, we use the parameter values shown in the text boxes in the figures. The computation of the ΛR-model is discussed in 3.2.

4.2 Figure 3

The function $\Omega^{\Lambda\mathrm{R}}(t)$ is defined by equation (2). The blue dotted graph is the ΛR-model in Figures 1 and 2 but integrated up to 50 Gyr. The red graph is computed by putting $\Omega^{\Lambda} \approx 0.6881$ in 2, and $a(t)$ the benchmark model trajectory shown in Figure 2. To find the maximum value of $\Omega^{\Lambda\mathrm{R}}(t)$ we use (5).

4.3 Figure 4

We have three cases: $\Omega^{B+D} \in \{0.2,\ 0.27,\ 0.3\}$. In all cases, we have $\Omega^{B} = 0.049$ and $\Omega^{\Lambda} \approx 0.6881$. The parameter α is solved from the flatness equation.

Note that in the pure F-model, i.e., $\alpha = 0$, we now have $\Omega^{B} = 0.049$, $\Omega^{D} = 0.2629$, $\Omega^{\Lambda} = 0.6881$.

4.4 Figure 5

In each graph, we have $\Omega^{D} = 0$, $\alpha = 1$ and Ω^{Λ} is solved from the flatness equation. We let Ω^{B} take a few certain values including $\Omega^{B}_{\mathrm{opt}}$, see 3.4.

4.5 Figures 6 and 7

We have $\Omega^{D} = 0$ and $\alpha = 1$, and we let Ω^{B} vary to produce the curves. We solve the flatness equation at each point on the curve and record the $\Omega^{\Lambda\mathrm{R}}_{T}$-values. In the linear approximation curve, we use $a(t) = H_T\, t$, $t \geq 0$; and integrate (2) analytically to the upper bound $t = T = 1/H_T$, which gives the flatness equation in the following form:

$$\Omega^{B} + \frac{1 - \left(1 + \sqrt{\Omega^{\Lambda}}\right)\exp\left(-\sqrt{\Omega^{\Lambda}}\right)}{\sqrt{\Omega^{\Lambda}}} + \Omega^{\Lambda} = 1.$$

The curves are also given in numerical form in Tables 1 and 2.

4.6 Figure 8

Note that one dotted curve (the last one in the box) is defined by,

$$\Omega^{B+D} = 0.2, \quad \alpha = 0.68, \quad \Omega^{B} = 0.049,$$

and another dotted curve by,

$$\Omega^{B+D} = 0.27, \quad \alpha = 0.3, \quad \Omega^{B} = 0.049.$$

In both, Ω^{Λ} is given by the flatness equation.

In the two remaining graphs, the parameters are as in 3.5, where we fix $\alpha\Omega^{\Lambda\mathrm{R}}_{T} = 0.06$, or 0.16.

4.7 Figures 9 and 10

There are three curves with $\alpha\Omega^{\Lambda\mathrm{R}}_{T} \in \{0.01, 0.02, 0.03\}$. These models are computed as in 3.5. The optimal ΛR-model is as in Figure 9.

4.8 Figures 11 and 12

Define $H_t = \dot{a}(t)/a(t)$, $t \geq 0$. Here, we have different values for H_T, all of which need to be converted to Gyrs, see 3.1. For the pure F-models, we have $\alpha = 0$, and all other required parameters are shown.

We choose the ΛR-model's parameter α, s.t., its age of the universe matches with the F-model 1, while $\Omega^{B+D} = 0.307$, $\Omega^{B} = 0.049$, and Ω^{Λ} is solved from the flatness equation.

5 Appendix

5.1 Figures

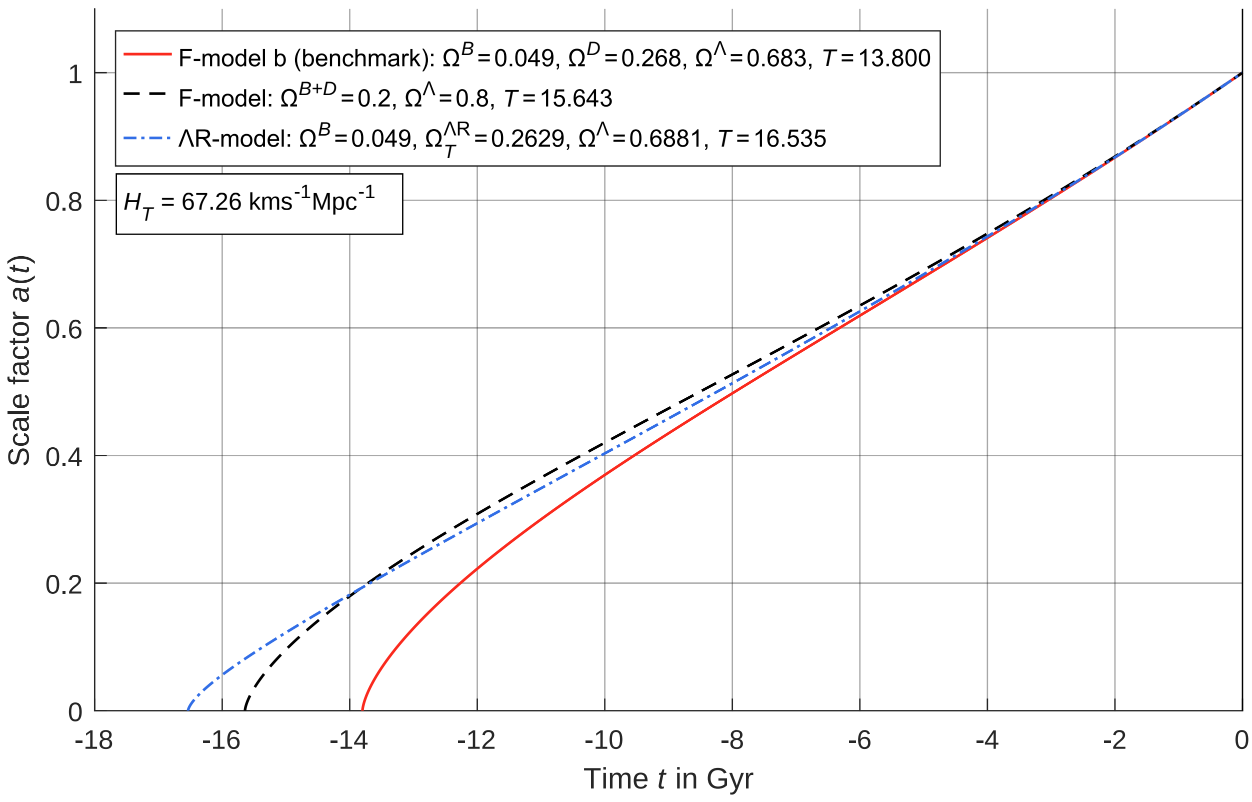

Figure 1

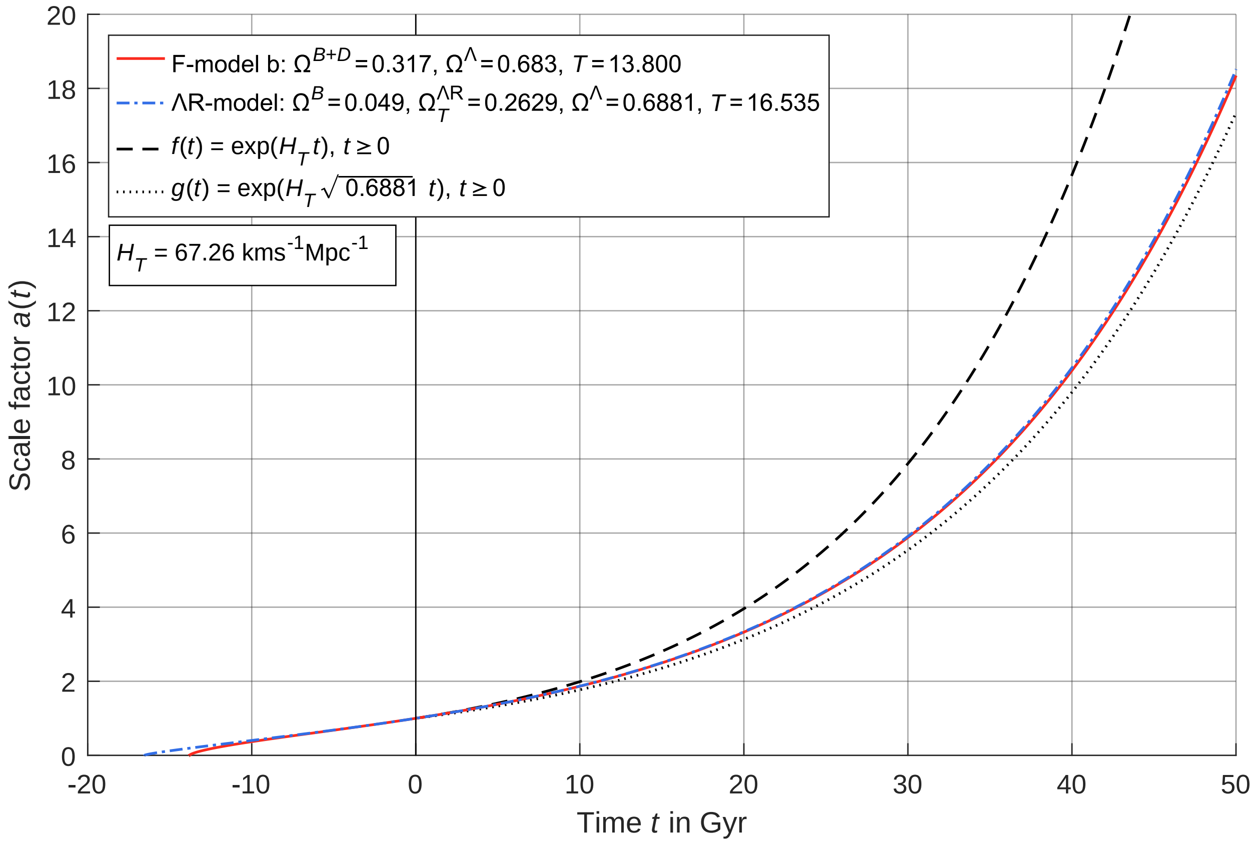

Figure 2

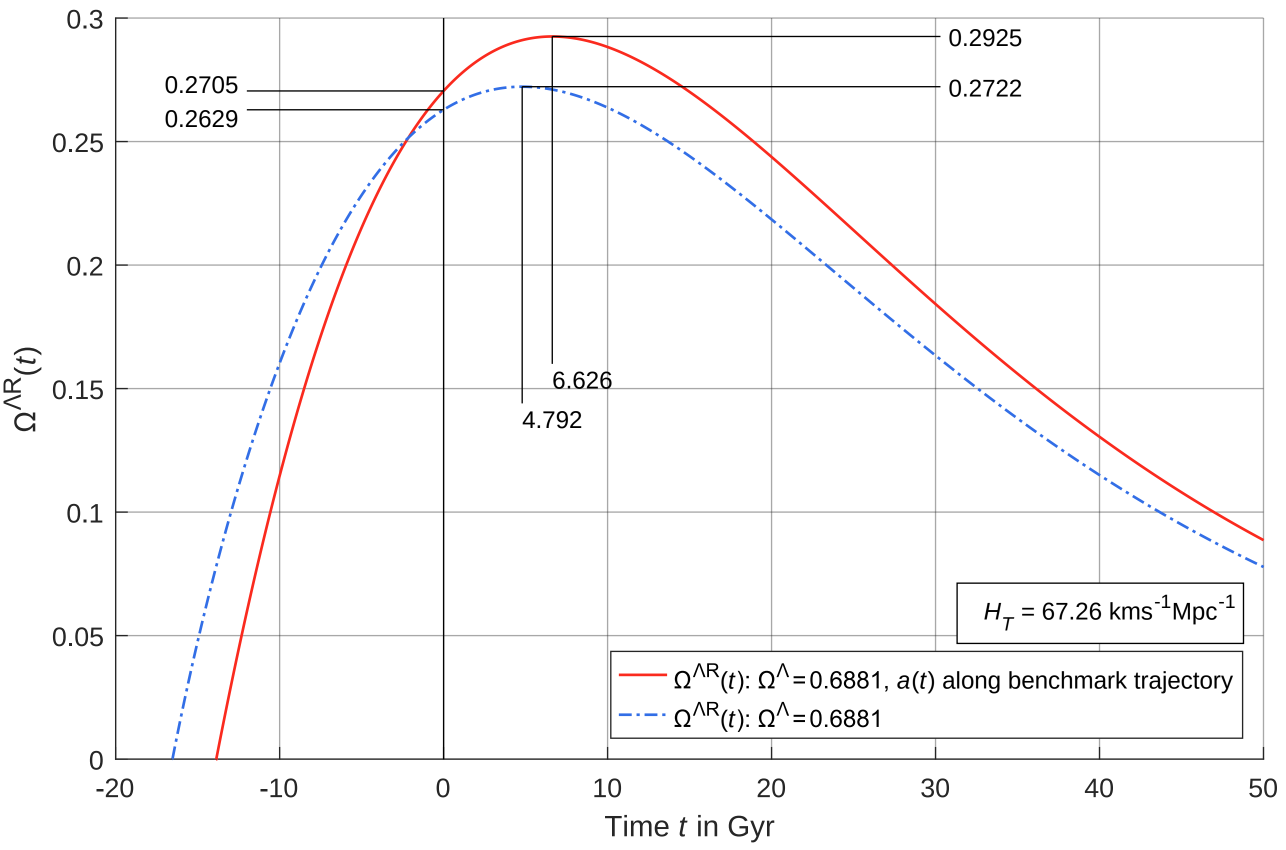

Figure 3

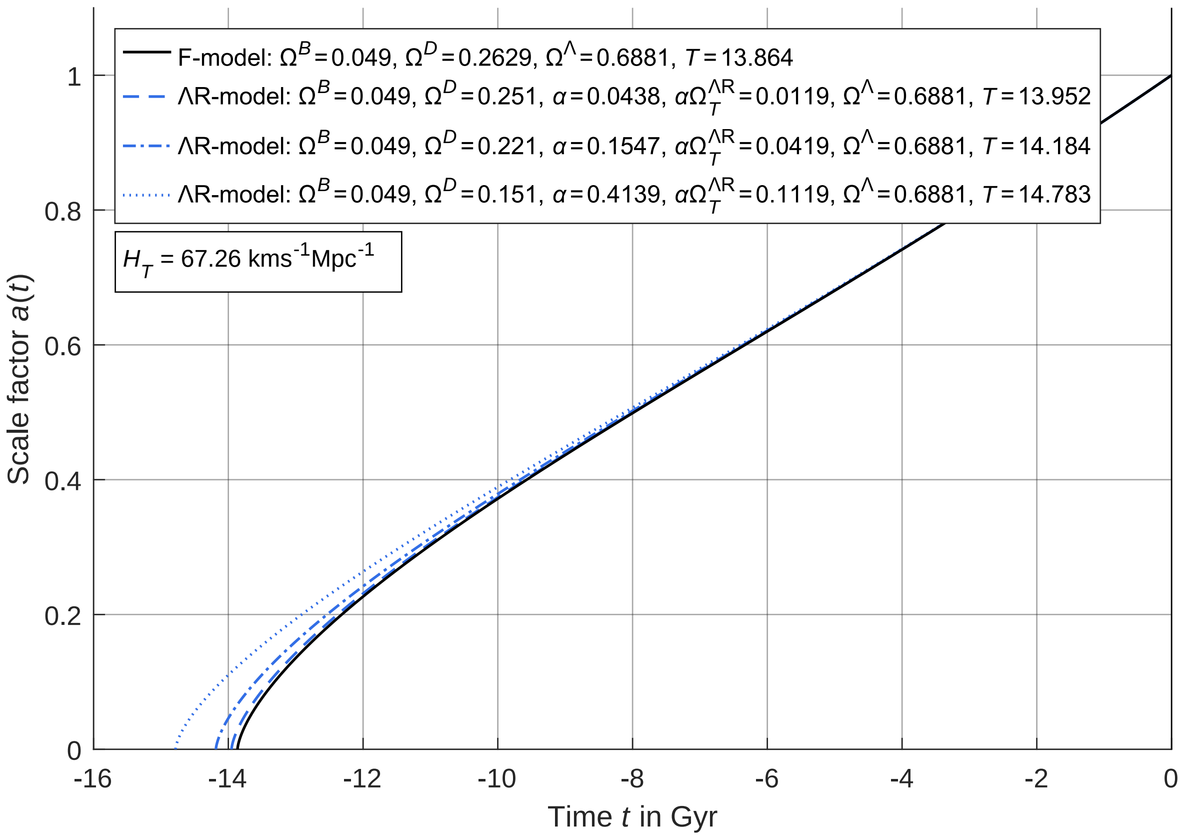

Figure 4

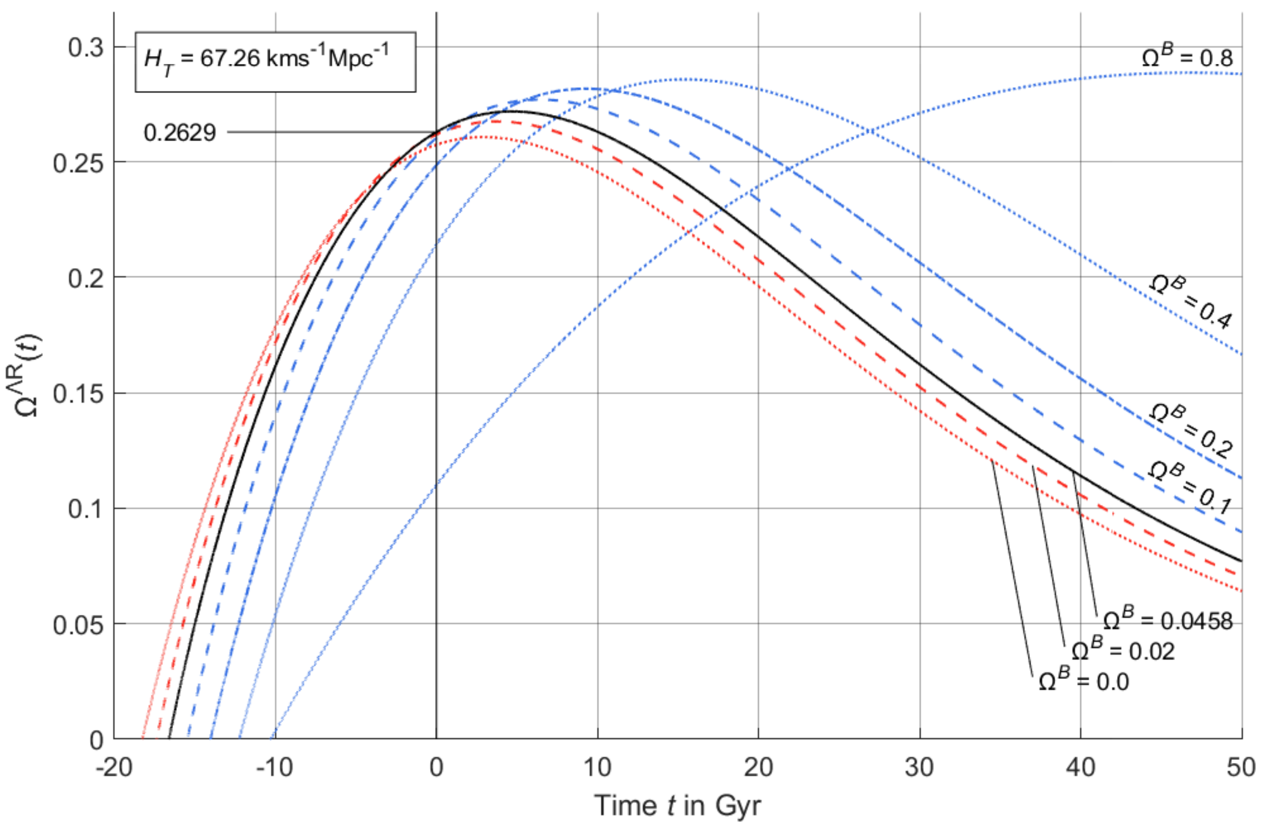

Figure 5

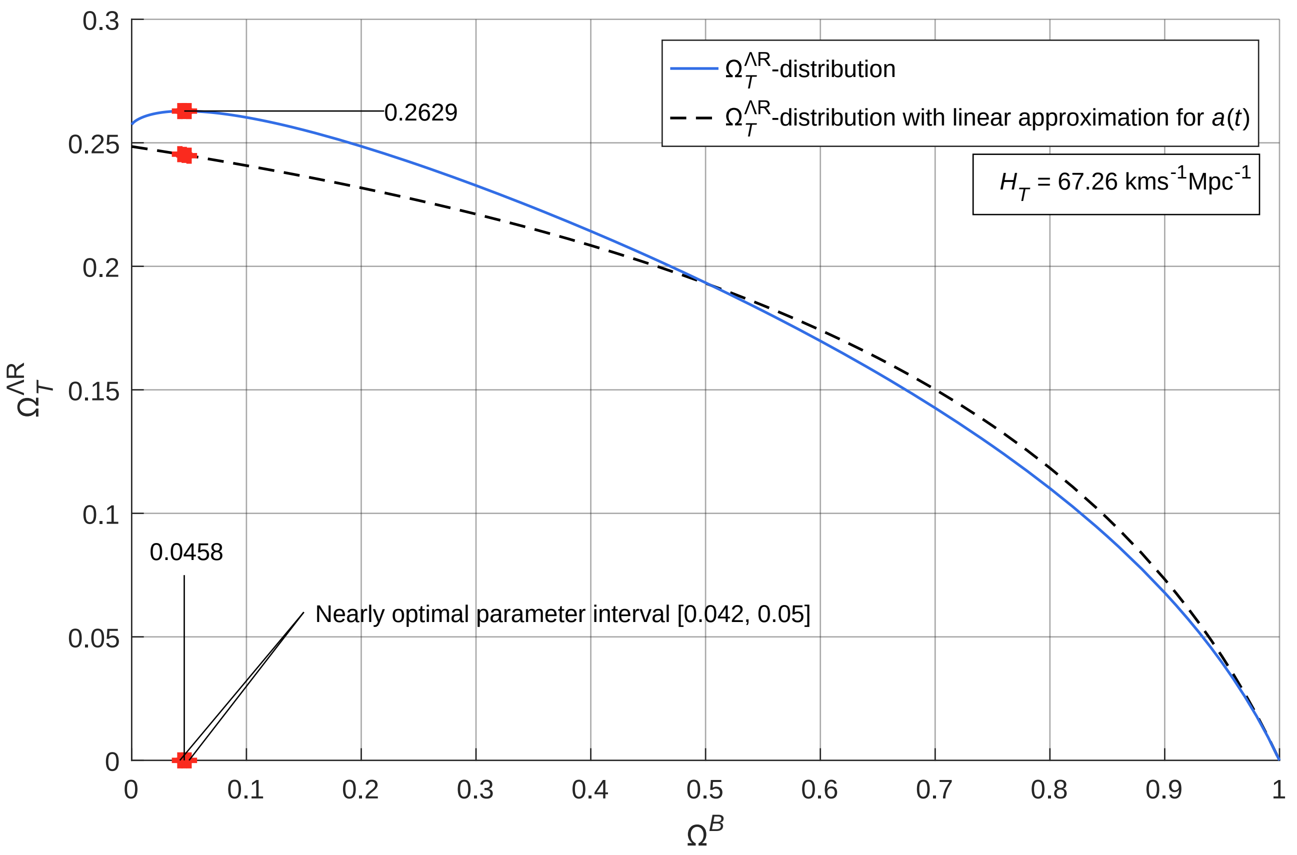

Figure 6

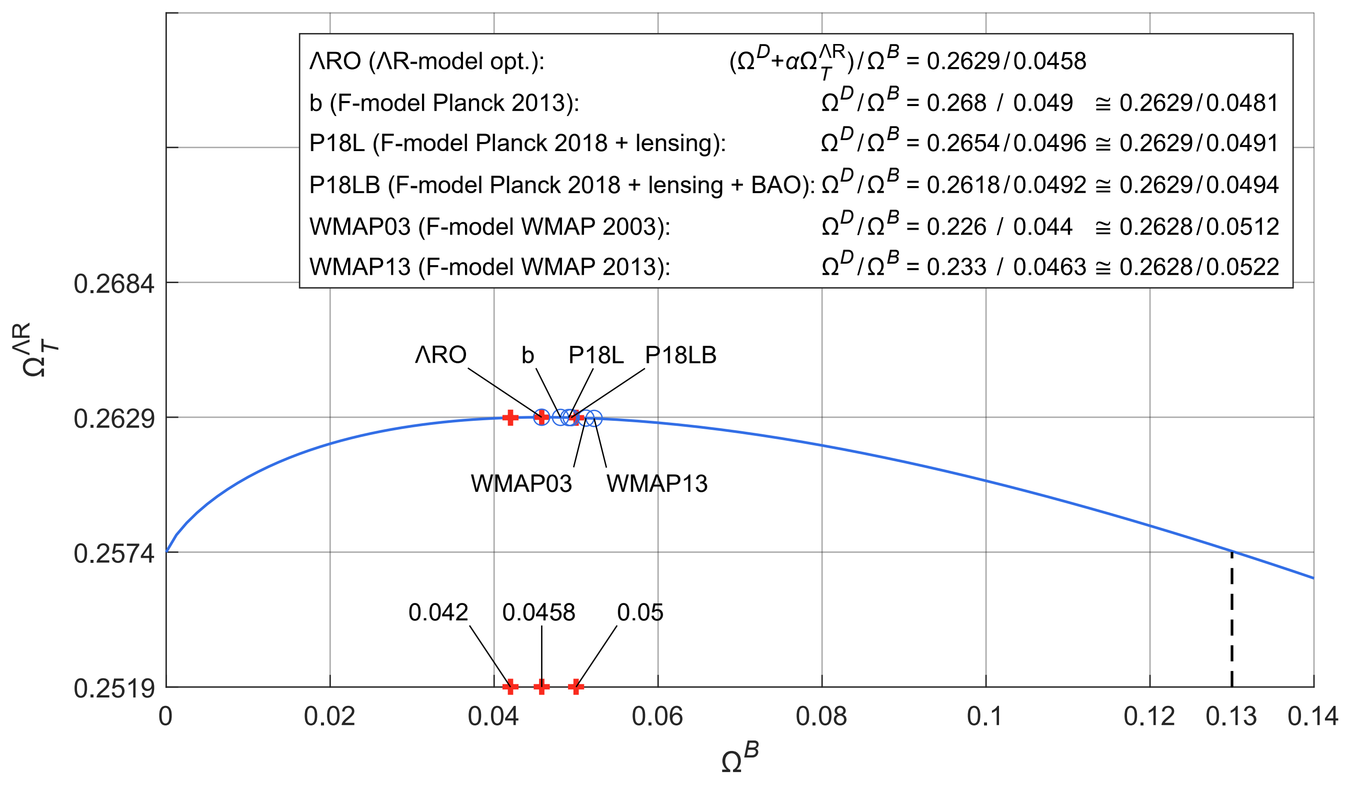

Figure 7

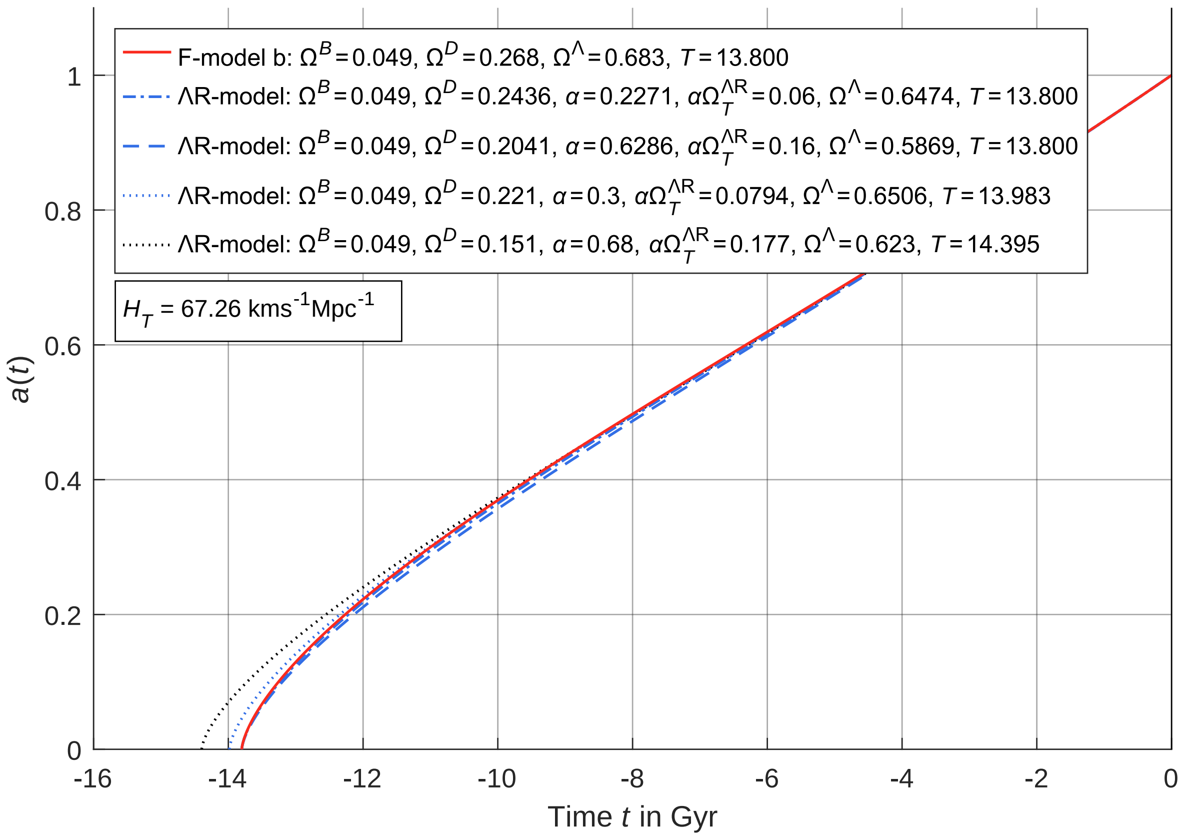

Figure 8

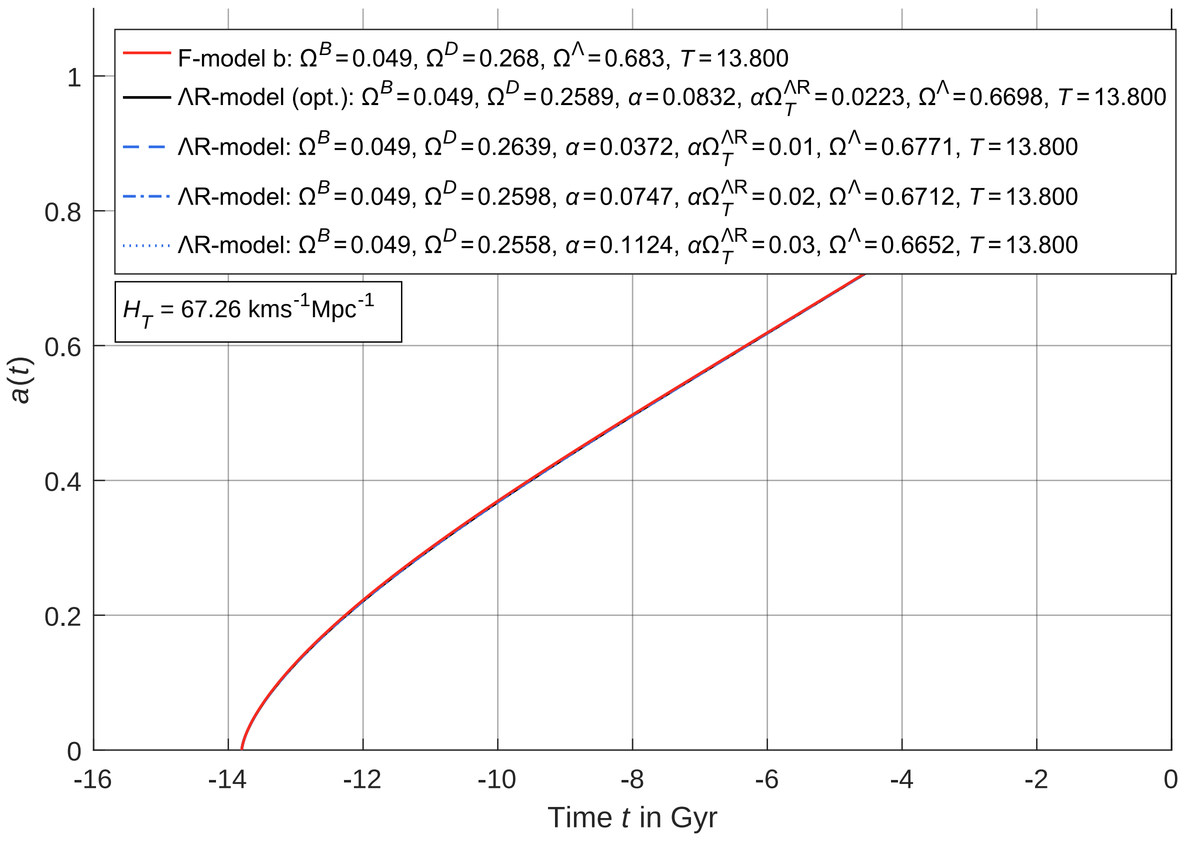

Figure 9

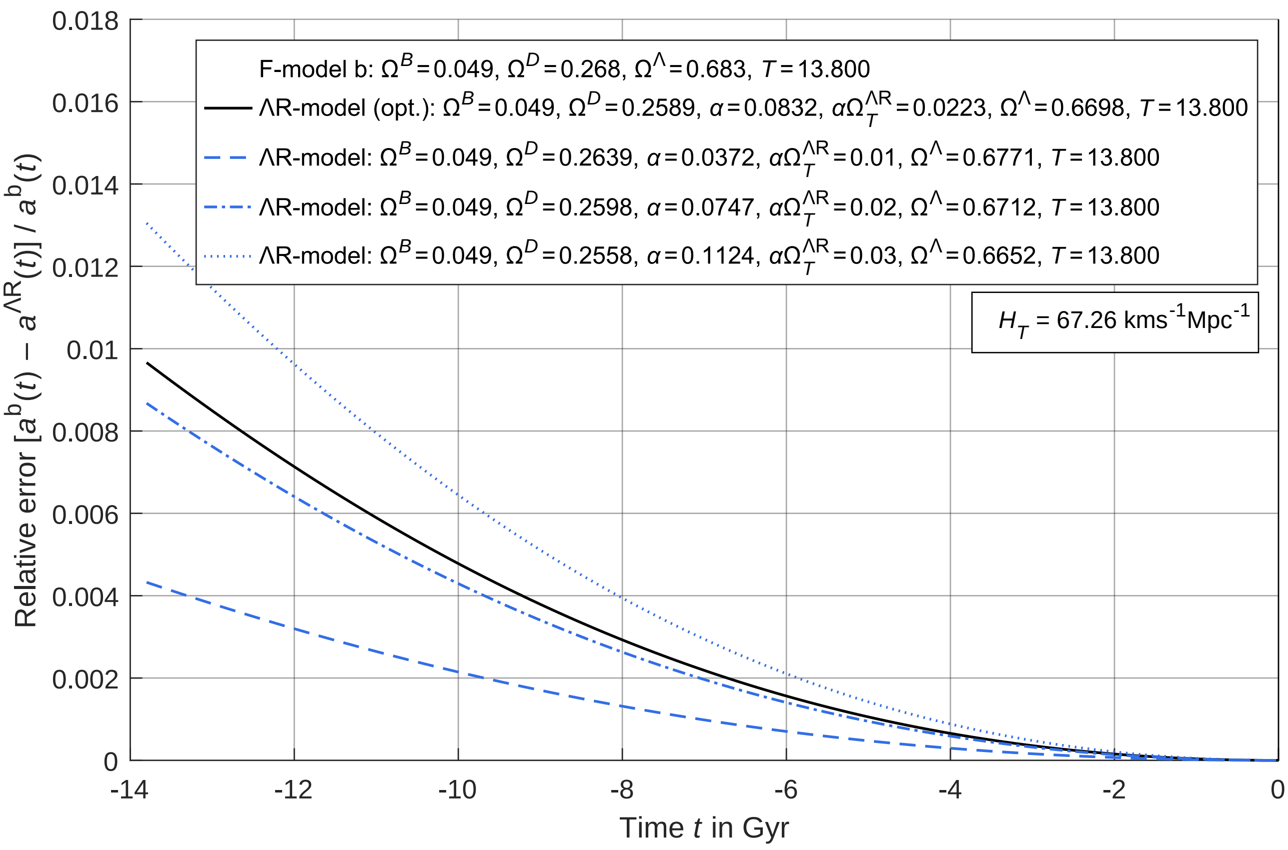

Figure 10

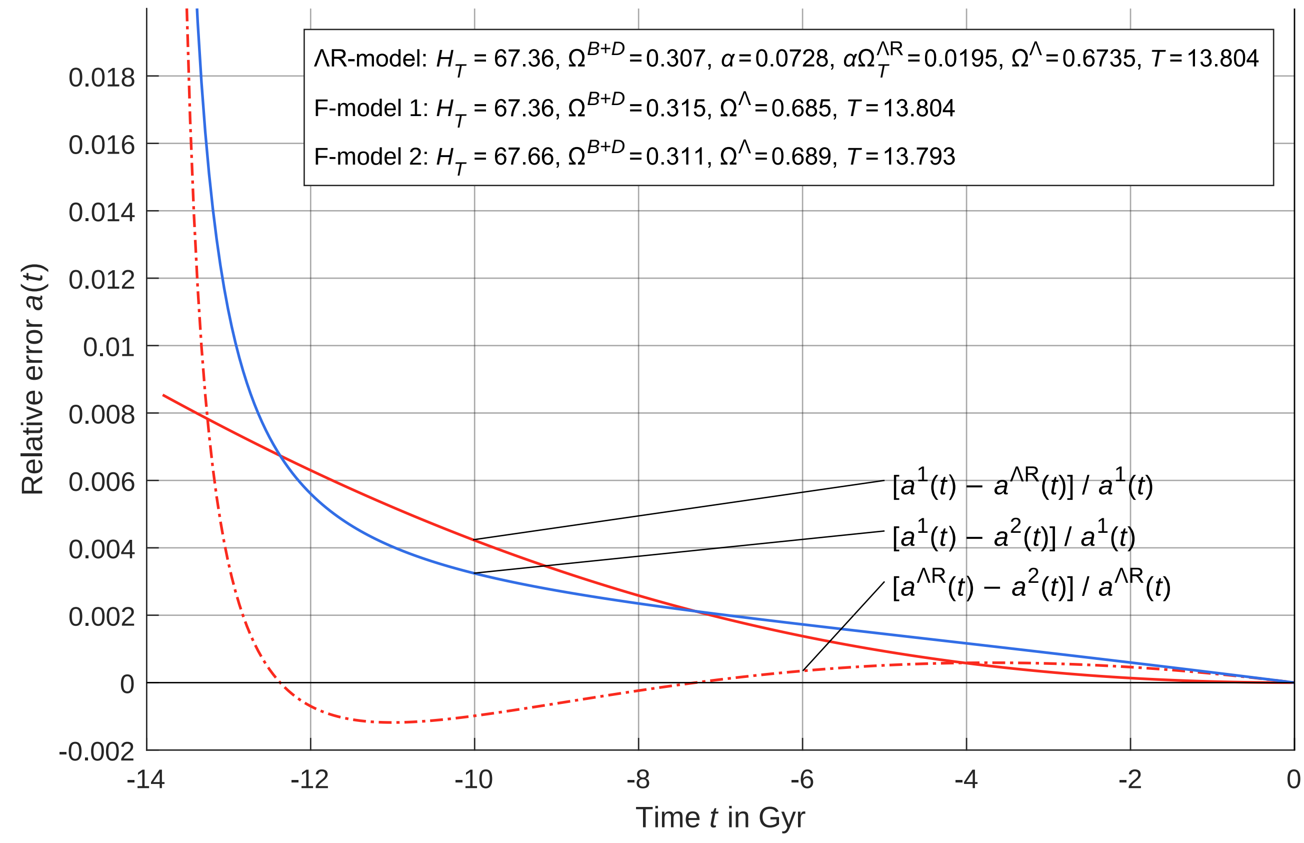

Figure 11

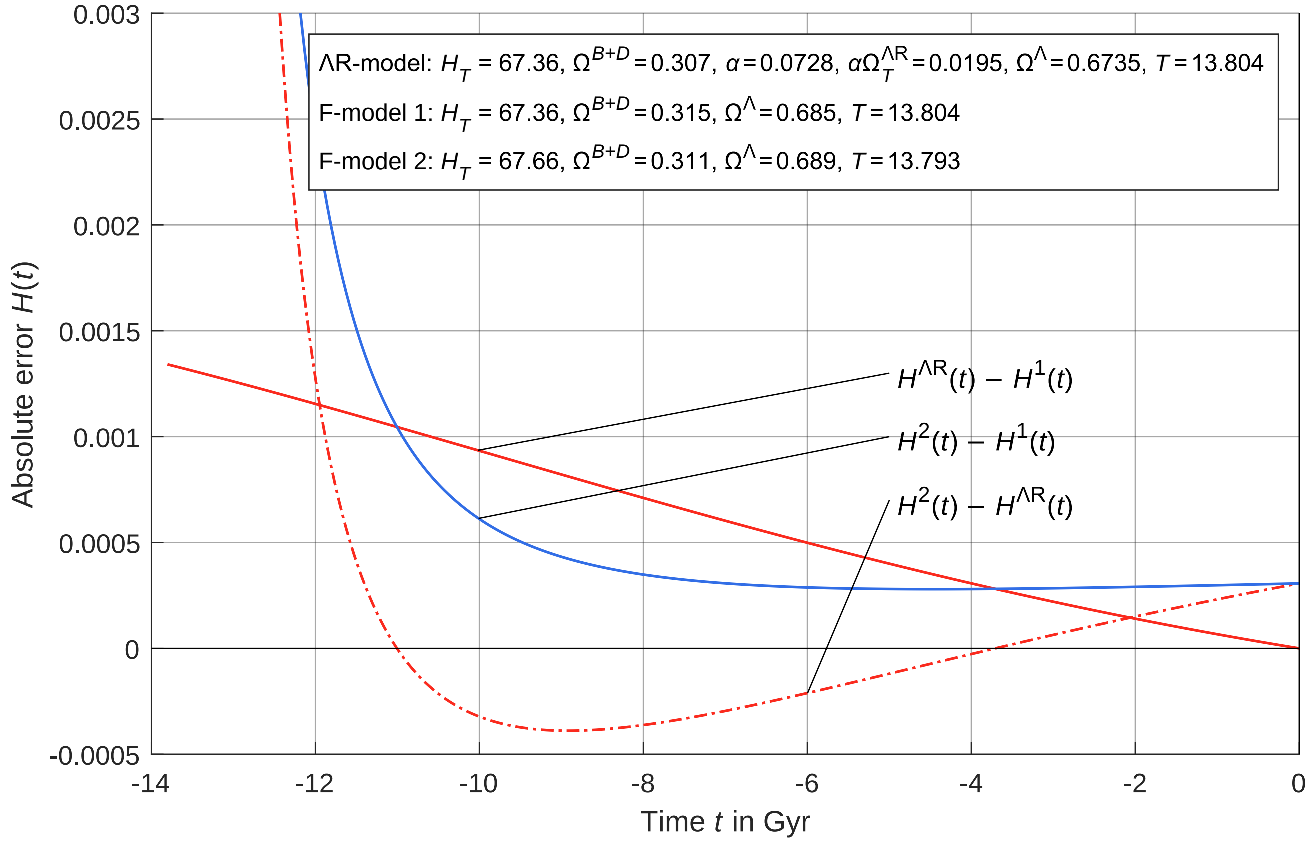

Figure 12

Table 1: T calculated for $H_T = 67.26\ \mathrm{kms}^{-1}\mathrm{Mpc}^{-1}$

Ω^B	$\Omega_T^{\Lambda\mathrm{R}}$	Ω^Λ	T	Ω^B	$\Omega_T^{\Lambda\mathrm{R}}$	Ω^Λ	T
0	0.2574	0.7426	18.270	0.48	0.1977	0.3223	11.746
0.01	0.2604	0.7296	17.749	0.50	0.1934	0.3066	11.631
0.02	0.2618	0.7182	17.374	0.52	0.1889	0.2911	11.520
0.03	0.2625	0.7075	17.054	0.54	0.1843	0.2757	11.413
0.04	0.2628	0.6972	16.769	0.56	0.1796	0.2604	11.310
0.042	0.2629	0.6951	16.715	0.58	0.1748	0.2452	11.210
0.0458	0.2629	0.6913	16.616	0.60	0.1698	0.2302	11.114
0.049	0.2629	0.6881	16.535	0.62	0.1647	0.2153	11.020
0.05	0.2629	0.6871	16.511	0.64	0.1594	0.2006	10.930
0.06	0.2626	0.6774	16.273	0.66	0.1540	0.1860	10.842
0.07	0.2623	0.6677	16.052	0.68	0.1484	0.1716	10.757
0.08	0.2617	0.6583	15.846	0.70	0.1427	0.1573	10.674
0.09	0.2611	0.6489	15.651	0.72	0.1367	0.1433	10.594
0.10	0.2603	0.6397	15.468	0.74	0.1304	0.1296	10.516
0.12	0.2585	0.6215	15.129	0.76	0.1240	0.1160	10.441
0.14	0.2563	0.6037	14.820	0.78	0.1172	0.1028	10.367
0.16	0.2540	0.5860	14.538	0.80	0.1101	0.0899	10.296
0.18	0.2514	0.5686	14.276	0.82	0.1027	0.0773	10.227
0.20	0.2486	0.5514	14.033	0.84	0.0948	0.0652	10.159
0.22	0.2457	0.5343	13.806	0.86	0.0864	0.0536	10.094
0.24	0.2426	0.5174	13.594	0.88	0.0775	0.0425	10.030
0.26	0.2394	0.5006	13.394	0.90	0.0679	0.0321	9.968
0.28	0.2361	0.4839	13.205	0.91	0.0627	0.0273	9.938
0.30	0.2327	0.4673	13.026	0.92	0.0574	0.0226	9.909
0.32	0.2292	0.4508	12.856	0.93	0.0518	0.0182	9.879
0.34	0.2256	0.4344	12.695	0.94	0.0458	0.0142	9.851
0.36	0.2219	0.4181	12.541	0.95	0.0396	0.0104	9.823
0.38	0.2181	0.4019	12.394	0.96	0.0329	0.0071	9.795
0.40	0.2142	0.3858	12.253	0.97	0.0257	0.0043	9.768
0.42	0.2102	0.3698	12.119	0.98	0.0179	0.0021	9.742
0.44	0.2062	0.3538	11.989	0.99	0.0094	0.0006	9.716
0.46	0.2020	0.3380	11.865	1	0	0	9.692

Table 2: a linear approximation for $a(t)$ is used

Ω^B	$\Omega_T^{\Lambda\mathrm{R}}$	Ω^Λ	Ω^B	$\Omega_T^{\Lambda\mathrm{R}}$	Ω^Λ
0	0.2485	0.7515	0.48	0.1964	0.3236
0.01	0.2478	0.7422	0.50	0.1931	0.3069
0.02	0.2471	0.7329	0.52	0.1897	0.2903
0.03	0.2463	0.7237	0.54	0.1861	0.2739
0.04	0.2456	0.7144	0.56	0.1823	0.2577
0.042	0.2454	0.7126	0.58	0.1783	0.2417
0.0458	0.2451	0.7091	0.60	0.1742	0.2258
0.049	0.2449	0.7061	0.62	0.1699	0.2101
0.05	0.2448	0.7052	0.64	0.1653	0.1947
0.06	0.2440	0.6960	0.66	0.1605	0.1795
0.07	0.2432	0.6868	0.68	0.1555	0.1645
0.08	0.2424	0.6776	0.70	0.1502	0.1498
0.09	0.2416	0.6684	0.72	0.1445	0.1355
0.10	0.2408	0.6592	0.74	0.1386	0.1214
0.12	0.2391	0.6409	0.76	0.1323	0.1077
0.14	0.2374	0.6226	0.78	0.1255	0.0945
0.16	0.2355	0.6045	0.80	0.1184	0.0816
0.18	0.2337	0.5863	0.82	0.1107	0.0693
0.20	0.2318	0.5682	0.84	0.1024	0.0576
0.22	0.2298	0.5502	0.86	0.0935	0.0465
0.24	0.2277	0.5323	0.88	0.0838	0.0362
0.26	0.2256	0.5144	0.90	0.0733	0.0267
0.28	0.2234	0.4966	0.91	0.0677	0.0223
0.30	0.2211	0.4789	0.92	0.0618	0.0182
0.32	0.2188	0.4612	0.93	0.0555	0.0145
0.34	0.2163	0.4437	0.94	0.0490	0.0110
0.36	0.2138	0.4262	0.95	0.0420	0.008
0.38	0.2112	0.4088	0.96	0.0347	0.0053
0.40	0.2085	0.3915	0.97	0.0269	0.0031
0.42	0.2056	0.3744	0.98	0.0186	0.0014
0.44	0.2027	0.3573	0.99	0.0096	0.0004
0.46	0.1996	0.3404	1	0	0

5.2 Code

This Matlab code is available on Github [2].

```
%%% Manual and Code to Compute Solutions and to Generate Figures for Flat
    Friedmann Differential Equation

% Harri Ehtamo and Lauri Jokinen.
% The code is maintained by Lauri Jokinen, lauri.jokinen@iki.fi.
% https://github.com/lauri-jokinen/FDE-with-Lambda-R-model

% This file includes all core routines regarding the figures.
% To keep the code short, the complete code for the figures is not shown.

% Contents:
% * Definitions of global variables (Hubble constant, etc.)
% * Example figure
% * Evaluate numerical results (Optimal Omega^B etc.)
% * ODE integration functions
% * Equation solver functions (flatness equation solver etc.)


%%% DEFINITIONS OF GLOBAL VARIABLES

% We define a struct G which is then passed to functions.

% Hubble constant, and unit conversion factor
G.convertion_factor = 0.001 * 3.15576 / (1.495978707 * 6.48 / pi);
G.Hubble_constant = 67.26 * G.convertion_factor;

% ODE integration termination time
% (put a low value for speed; high value for robustness)
G.ode_t_termination = 30;

% This number sets required precision for all numerical routines
% Lower number increases precision.
G.precision = 1e-3;

% Options for numerical algorithms
% Decreasing the value G.precision will increase precision of the
    algorithms.
G.fsolveOptions  = optimoptions('fsolve', ...
            'OptimalityTolerance', G.precision, ...
            'FunctionTolerance', G.precision, ...
            'StepTolerance', G.precision, ...
            'MaxFunctionEvaluations', ceil(sqrt(1/G.precision)), ...
            'MaxIterations', ceil(sqrt(1/G.precision)), ...
            'Display', 'off');
G.fminuncOptions = optimoptions('fminunc', ...
            'OptimalityTolerance', G.precision, ...
            'StepTolerance', G.precision, ...
            'MaxFunctionEvaluations', ceil(sqrt(1/G.precision)), ...
            'MaxIterations', ceil(sqrt(1/G.precision)), ...
            'Display', 'off');


%%% EXAMPLE FIGURE

OmegaBD = 0.049;
```

```matlab
OmegaL = 0.6881;
alpha = 1;
[~, a, t, Omega_LR, b] = LR_model(G, OmegaBD, OmegaL, alpha);
plot(t, a)
title('Example \LambdaR-model')
xlabel('Time in Gyrs')
ylabel('a(t)')


%%% EVALUATE NUMERICAL RESULTS
% Evaluate numerical results described in the Code Manual
% We store the values to G

G.Omega_B = 0.049;

% Solve flatness equation
G.LR_Omega_L = flatness_solve_Omega_L(G, G.Omega_B, 1);

% Age of the universe, with benchmark model
[~, ~, t_benchmark] = LR_model(G, 0.049 + 0.268, 0.683, 0);
G.benchmark_age = -t_benchmark(1);

% Optimal Omega^B
disp('Evaluating optimal Omega^B... this may take a while')
[G.Omega_LR_opt, G.Omega_B_opt, G.Omega_L_opt] = optimal_Omega_B(G);
disp('Done!')

% Age-optimal Omega^D and alpha: case 2
[G.Omega_D_opt, G.alpha_opt] = optimal_Omega_D_and_alpha(G);

% Show the results
disp(G)


%%% ODE INTEGRATION

% Main function
function [Omega_LR_T, a, t, Omega_LR, b, T_index] = LR_model(G, Omega_BD,
   Omega_L, alpha)
    % G is the global struct
    % T_index has the property a(T_index) == 1 (or closest to it)

    % Initial values
    a0 = 1e-16; b0 = 0; t0 = 0;

    % Define the ODE system with function 'odes' which is defined below
    y_dot = @(t,y)odes(t, y, G.Hubble_constant, Omega_BD, Omega_L, alpha);

    % ODE options:
    %   define events with function 'ode_events' which is defined below
    %   ODE solver's precision is set with G.precision
    opts = odeset('Events', @(t,y)ode_events(t, y, y_dot), ...
        'RelTol', 1e-1*G.precision, ...
        'AbsTol', 1e-2*G.precision); % Create events function

    % Run ODE solver
    [t, y, t_events, y_events] = ode23s(y_dot, [t0, G.ode_t_termination], [
       a0;b0], opts);
    % Add the 'events' to the result vectors, one by one
    for p = 1:length(t_events)
        ind = sum(t < t_events(p));
        t = [t(1:ind);   t_events(p);   t((ind+1):end)];
        y = [y(1:ind,:); y_events(p,:); y((ind+1):end,:)];
    end

    a = y(:,1);
    b = y(:,2);

    % find the event, a(T) = 1, and shift time axis, s.t., a(t=0) = 1
    [~, T_index] = min(abs(a - 1));
    T_index = T_index(1);
    t = t - t(T_index);

    % evaluate Omega_Lambda_R(t)
    Omega_LR = G.Hubble_constant * sqrt(Omega_L) * b ./ a;
    Omega_LR_T = Omega_LR(T_index);
end

% The differential equations
function y_dot = odes(t, y, Hubble_constant, Omega_BD, Omega_L, alpha)
    % In pseudocode: y_dot = [a'(t); b'(t)];
    y_dot = [Hubble_constant * sqrt(Omega_BD/y(1) + alpha*Hubble_constant*
       sqrt(Omega_L)*y(2)/y(1)^2 + y(1)^2*Omega_L);
             y(1) * exp(-t * Hubble_constant * sqrt(Omega_L))];
end

% Declare ODE events
function [value, isterminal, direction] = ode_events(t, y, y_dot)
    y_dot_value = y_dot(t, y);
    value = [y(1) - 1;                                   % event a(t) = 1
             y(1)*y_dot_value(2) - y(2)*y_dot_value(1)];  % event max Omega
                 ^{Lambda R}_T
    % ignore further definitions
    isterminal = [0; 0];
    direction = [0; 0];
end


%%% EQUATION SOLVERS

% Solves Omega^Lambda from the flatness equation
function Omega_L = flatness_solve_Omega_L(G, Omega_B, alpha)
    % Function 'LR_model' returns value 'Omega_LR_T'
    Omega_L = fsolve(@(Omega_L) alpha*LR_model(G, Omega_B, abs(Omega_L),
       alpha) + Omega_B + abs(Omega_L) - 1, ...
                    (1-Omega_B)^1.6,... % initial guess
                    G.fsolveOptions);
    Omega_L = real(Omega_L); % due to numerical inaccuracies, the result
       may have a small imaginary component
end

% Solves optimal Omega^B
function [Omega_LR, Omega_B, Omega_L] = optimal_Omega_B(G)
    % Function 'LR_model' returns value 'Omega_LR_T'
    [Omega_B, Omega_LR] = fminunc(@(Omega_B)-LR_model(G, Omega_B,
        flatness_solve_Omega_L(G,Omega_B,1), 1), ...
                                        0.0458, ... % initial guess
                                        G.fminuncOptions);
    Omega_LR = -Omega_LR; % change sign because above we have minimization
        instead of maximization
    Omega_L = 1 - Omega_B - Omega_LR;
end

% Solves age-optimal Omega^D and alpha: case 2
function [Omega_D_opt, alpha_opt] = optimal_Omega_D_and_alpha(G)
    x = fsolve(@(x)objective_function_optimal_Omega_D_and_alpha(G, x(1), x
        (2)),[0.26,0.083], G.fsolveOptions);
    Omega_D_opt = x(1);
    alpha_opt = x(2);
end

% This is used in the above solver
function res = objective_function_optimal_Omega_D_and_alpha(G, Omega_D,
    alpha)
    Omega_L = 1 - G.Omega_LR_opt/G.Omega_B_opt * G.Omega_B - G.Omega_B;
    % Function 'LR_model' returns value 'Omega_LR_T'
    [~, ~, t, Omega_LR, ~, T_index] = LR_model(G, G.Omega_B + Omega_D,
        Omega_L, alpha);
    res = [(Omega_D + alpha*Omega_LR(T_index)) / G.Omega_B - G.Omega_LR_opt
        /G.Omega_B_opt;
           t(1) + G.benchmark_age];
end
```

References

[1] Harri Ehtamo. *A Coincidence Problem Related to the Λ-CDM Cosmological Model.* Private Manuscript, July 2019.

[2] Harri Ehtamo and Lauri Jokinen. *Manual and Code to Compute Solutions and to Generate Figures for Flat Friedmann Differential Equation.* 2023. The page is maintained by Lauri Jokinen. URL: https://github.com/lauri-jokinen/FDE-with-Lambda-R-model.

[3] Wikipedia. *Parsec.* visited in 2023. URL: https://en.wikipedia.org/wiki/Parsec.