Physics 205
Dr. Dimitrios Psaltis
Submitted: 5 December 2006

By
Christopher Limbach, Alexander Luce, and Amanda Stiteler
I. INTRODUCTION

The standard Big Bang cosmological model traces the universe's origins back to a hot, dense, radiation-dominated state. Depending on the density of matter, radiation, and energy, the universe may either continue to expand indefinitely or collapse. On a large-scale average, we observe a universe that is nearly homogeneous and isotropic, with negligible small-scale mass distribution fluctuations. Recent supernova and CMB measurements also indicate an accelerating expansion.

Einstein formulated a cosmological model prior to the discovery of cosmic expansion. This model was compatible with Mach's principle, in which all the matter in the universe provided a "background" frame of reference against which motion could be measured. The gravitational field in such a universe is governed by the equation,

\[ R_{ij} - \frac{1}{2} g_{ij} R - \Lambda g_{ij} = 8\pi G T_{ij}, \]

where the matter distribution, \( R_{ij} \) (the Ricci tensor) and \( R \) (the curvature scalar), are functions of the space-time metric tensor \( g_{ij} \). \( G \) is Newton's gravitation constant, \( T_{ij} \) is the stress-energy tensor measuring the relevant properties of matter in the universe, and \( \Lambda \) is the cosmological constant, a term included to allow for conditions of static equilibrium. Without \( \Lambda \), matter density and spatial curvature would be negative. This constant was added to agree with observational evidence available at the time. However, Einstein later discarded \( \Lambda \) after new observations indicated an expanding universe.

The Friedmann-Lemaître model modifies Einstein's formulation by addressing the universe's expansion. This model details the expansion history of the universe by adopting a set of dimensionless parameters which sum to unity,

\[ \Omega_{\text{M0}} + \Omega_{\text{R0}} + \Omega_{\Lambda0} + \Omega_{\text{K0}} = 1 \]

where \( \Omega_{\text{M0}} \) measures the present average mass density of non-relativistic matter (baryons and non-baryonic dark matter), \( \Omega_{\text{R0}} \) measures the present mass in the relativistic thermal cosmic microwave background radiation and the accompanying neutrinos, \( \Omega_{\Lambda0} \) measures the present dark-energy density, and \( \Omega_{\text{K0}} \) is a measure of space's curvature. Current observations suggest that \( \Omega_{\text{K0}} = 0 \), \( \Omega_{\Lambda0} \approx 0.73 \), \( \Omega_{\text{M0}} \approx 0.27 \), and \( \Omega_{\text{R0}} \approx 0.00003 \).

The density parameter of a particular component, \( \Omega_i \), is proportional to that component's energy density, \( \rho_i \), divided by a critical energy density \( \rho_c \), being the critical density of a spatially flat universe defined by

\[ \rho_c = \frac{3H^2}{8\pi G} \]

These results imply that matter accounts for only 27% of the universe's energy; the remaining 73% is composed of some unknown, "dark energy".

Observations of high redshift supernovae, the cosmic microwave background (CMB), and galaxy clustering have led to the conclusion that the universe is accelerating. This can be observed in a dynamic Hubble parameter, which is defined as,

\[ H = \frac{1}{a} \frac{da}{dt} \]
Where $a(t)$ is a dimensionless scale factor. The present value of the Hubble parameter, $H_0$, has been measured to be $68 \pm 7$ km/s/Mpc$^1$.2.

The acceleration of the universe is related to Newton’s gravitational constant, the mean mass density of all matter, $\rho(t)$, and the pressure of all matter, $p(t)$, by

$$\frac{1}{a} \frac{da}{dt} = -\frac{4}{3} \pi G (\rho + p)$$

Integration of this equation and the local energy conservation laws give

$$\frac{d\rho}{dt} = -\frac{3}{\alpha} \frac{d\alpha}{dt} (\rho + p)$$

which results in the Friedmann equation:

$$\left( \frac{1}{a} \frac{da}{dt} \right)^2 = \frac{8}{3} \pi G \rho + \text{const.} = \frac{8}{3} \pi G (\rho_r + \rho_m + \rho_\Lambda) - \frac{k c^2}{l_0^2 a^2}$$

with $k$ being a parameter that measures the universe’s curvature and $l_0$ being the present distance between two galaxies2,6.

Coincidently, it appears that we exist in an era where matter, $\rho_m$, and the inherent vacuum energy of space $\rho_{\text{vac}}$, the lowest attainable energy state, are similar4. One of the simplest explanations for this invokes “quintessence”: a scalar field slowly rolling in a potential $V(\phi)^7$. The energy density of this scalar field equals the sum of its kinetic and potential energies,

$$\rho_\phi = \frac{m_p^2}{32 \pi} \left[ (\dot{\phi})^2 + \kappa m_p^2 \phi^{-\alpha} \right]$$

where $\kappa$ and $\alpha$ are constants which characterize the potential $V(\phi)$. Motion of the scalar in a homogeneous, isotropic, expanding universe is modeled by the differential equation

$$\ddot{\phi} + 3 \frac{a}{a} \dot{\phi} - \frac{\kappa c}{2} m_p^2 \phi^{-(\alpha + 1)} = 0.$$  

In order to mimic a nearly constant $\Lambda$ at late times $V(\phi)$ must be large and fairly flat7. This model follows from the inflation scenario, which explains the large-scale homogeneity of the current universe. The presence of a relatively flat scalar field potential, $V(\phi)$, acting like a cosmological constant, may be the cause of inflation. In this scenario, $\phi$ rolls down its potential until $V(\phi)$ steepens enough to end inflation. At this point, the Friedmann-Lemaître model takes effect, resulting in the decay of the scalar field’s energy into matter and radiation. As the universe continues to expand over time, this “dark energy” density continues to decrease5.

In this project, we study the universal effects of a dynamical dark energy density by comparing the evolution of a universe with constant dark energy density to one with variable dark energy density modeled by, 6

$$V(\phi) = \kappa \phi^{-\alpha}$$

We determine restrictions for the values of the constants $\kappa$ and $\alpha$ using the observationally allowed age range of the universe and examine the dependence of the universe’s age on each parameter. This is accomplished using a program that simulates the universe backwards in time from present until it reaches a size of less than 5% its current size. We use these results to compare variable and constant dark energy density universes. This comparison
involves the scale factor, energy densities, and the behavior of the scalar $\phi$, both in the past and future of the universe.

II. ASSUMPTIONS

According to Robert Kirshner, “Part of the fun of Cosmology is that it takes us into realms where laboratory physics doesn’t yet reach”\(^8\). Because of our limited understanding of the universe, we are forced to assume certain conditions that, although they agree with observational data, are still experimentally unproven. These assumptions play a crucial role in the outcome of our numerical simulations and have been chosen based on current observations.

The Hubble Constant, $H_0$, is a measure of the current expansion rate of the universe. The Hubble Key Project is an attempt to measure the Hubble Constant using the Hubble Space Telescope whose work has lead to a value of 72 ± 7 km/sec/megaparsec\(^9\). This value agrees with the value (72.5 ± 0.05) calculated using Wilkinson Microwave Anisotropy Probe (WMAP) observations\(^2\). The fact that these two values agree is important given that they rely on different methods of observation. In our simulations we assume that the current value of the Hubble Parameter is 68 km/sec/Mpc\(^2\), which is the ‘most probable’ value given by the WMAP results\(^3\).

The spatial curvature parameter $k$ is assumed to be zero since the net energy density of the universe must be very, very nearly the critical density for the universe to be what it is today. The general consensus holds that the current energy density equals the critical density, because it would be quite a puzzle for the density to be very nearly but not exactly the critical density. In this case our model reduces to the Friedmann equation with,

$$\Omega_{M0} + \Omega_{\Lambda 0} = 1$$

(12)

The universe is modeled with $\Omega_{M0} = \Omega_{\Lambda 0} \approx 0.73$ and $\Omega_{M0} \approx 0.27$, where the 0 subscript denotes observationally derived values for the current epoch\(^5\). Most importantly, we assume that the universe is homogeneous and isotropic, is currently accelerating such that $d^2a/dt^2 > 0$, and that this acceleration is caused by a scalar field $\phi$ following the treatment in Peebles and Ratra\(^6\).

III. EQUATIONS SOLVED

There are three main equations used in this project, Equations 8, 9, and 10. Because a flat universe is assumed Equation 8 reduces to,

$$\left(\frac{1}{a} \frac{da}{dt}\right)^2 = \frac{8}{3} \pi G (\rho_r + \rho_M + \rho_\Lambda),$$

(13)

To convert equation (13) into dimensionless units, let the dimensionless energy density, $\rho_{id}$, be defined as,

$$\rho_{id} = \frac{\rho_i}{\rho_0},$$

(14)

We set $\rho_0$ equal to the critical density, $\rho_{id}$.

$$\rho_0 = \rho_{id} = \frac{3H_0^2}{8\pi G},$$

(15)

We furthermore define the dimensionless time $t_d$ as follows,
Analysis of equation (13) with these dimensionless parameters results in,

$$t_0 = \sqrt{\frac{1}{\rho_0 G}} = 41.6195 \text{ Gyr} \quad (17)$$

To determine initial conditions we rearrange Equation 10,

$$3\frac{\dot{a}}{a} = \dot{\phi} - \frac{\kappa \alpha}{2} \phi^{-(\alpha+1)} \quad (18)$$

Differentiating Equation 9 with respect to time gives,

$$0 = \phi \left[ \ddot{\phi} - \frac{\kappa \alpha}{2} \phi^{-(\alpha+1)} \right] \quad (19)$$

Substituting Equation 18 into 19 results in,

$$-3\phi^2 \frac{\dot{a}}{a} = 0 \quad (20)$$

Because $H_0$ is non-zero and observing that at $t = 0$, $\rho_d = \rho_r \approx 0.73$, we find our initial conditions for $\phi$ to be as follows.

$$\dot{\phi} = 0 \quad (21)$$

$$\phi_i = \left( \frac{\kappa}{(0.73)32\pi} \right)^{\frac{1}{\alpha}} \quad (22)$$

$$\dot{\phi}_i = \frac{\kappa \alpha}{2} \left( \frac{\kappa}{(0.73)32\pi} \right)^{\frac{\alpha+1}{\alpha}} \quad (23)$$

IV. Numerical Methods

We use the fourth-order Runge-Kutta method to simultaneously solve three coupled equations,

$$\frac{da}{dt_d} = a \sqrt{\frac{8}{3}} \pi (\rho + \rho_\phi) \quad (24)$$

$$\frac{d\dot{\phi}}{dt_d} = \ddot{\phi} \quad (25)$$

Taking into consideration redshift and geometry we find that the energy density of matter and radiation vary with time like$^5$, 

$$\frac{d\phi}{dt_d} = \dot{\phi}$$

$$\frac{d\dot{\phi}}{dt_d} = \ddot{\phi} \quad (26)$$
\[ \rho_{\text{md}}(a) = \frac{\rho_{\text{md}0}}{a^3} \]  
\[ \rho_{\text{ed}}(a) = \frac{\rho_{\text{ed}0}}{a^4} \]
while \( \rho_{\text{ed}}(a) \) is governed by Equation 9 and \( \dot{\phi} \) is defined by Equation 10.

**V. VERIFICATION OF THE NUMERICAL METHODS**

The fourth-order Runge-Kutta code is the heart of every simulation run for this project. Two flavors of this code were written: one with a constant dark energy density (Runge-Kutta only for the scale factor \( a \)) and one with a variable dark energy (Runge-Kutta for \( a, \phi \) and \( \dot{\phi} \)). The verification of each flavor will be treated in turn.

1. **Verification of Runge-Kutta 4 for Constant Dark Energy**

   With a constant dark energy density, only Equation 24 needs numerically because \( \rho_{\text{ed}} = 0.73 \). Examining the fictional case where \( \rho_{\text{md}} = \rho_{\text{ed}} = 0 \) and \( \dot{\rho}_{\text{md}} = 1 \) allows for an analytic solution, which is

   \[ a = \left( \frac{8\pi G \rho_{\text{md}}}{2\rho_0} t + 1 \right)^{\frac{2}{3}}. \]

   The program minerr uses the 4th order Runge-Kutta to numerically compute \( a \) when the universe is 20 billion years older than at present and when \( a = 0.05 \) (analytically). This is done for a range of time-steps, and the difference between the analytic solution and the numeric solution is found. This data is plotted in Fig. 1.

   As can be seen, the ideal time-step in each case is different: \( h = 10^{-4} \) for the \( t = +20 \) Gyr limit and \( h = 10^{-3} \) for the \( a = 0.05 \) limit. However, for both ideal time-steps the maximum error is very small, on the order of \( 10^{-13} \). Fig. 1 shows not only that the numerical solution agrees extremely well with the analytic solution, but a time-step of \( h = 10^{-4} \) will minimize the error in the numeric solution. For this investigation the numeric solution’s accuracy will be most critical when the universe is very small and evolving rapidly than when it is large and evolving more slowly. The scale-factor’s accuracy at early times directly impacts the accuracy in the calculated age of the universe.
2. Verification of Runge-Kutta 4 for a Universe where $\rho_d = \rho_\phi$.

When $\rho_d = \rho_\phi$ the coding can no longer be verified using an analytic solution for a simplification of the equations. In this case the fourth-order Runge-Kutta is implemented for Equations 24, 25 and 26 simultaneously using the initial conditions derived in Section III. The numerical solution is verified by examining its convergence to the real, unknown solution as the time-step decreases.

The program minerr2 examines this convergence. First, minerr2 computes $a$ and $\phi$ when $t = 20$ billion years and $a < 0.05$ for an initial time-step. It then changes the time-step by some small increment and computes $a$ and $\phi$ at those same conditions but for the new time-step. The difference between the two models is computed and the process continues another small time-step increment forward. Plotting this difference against time-step size produces Fig. 2. The linearity at small time-steps shows that the solution for $a$ and $\phi$ converges with a power-law dependence. At some very small time-step the error will begin to increase due to round-off errors, as seen in Fig. 1. However, Fig. 2 shows that this must occur when $h < 10^{-6}$.

3. Conclusions and Other Considerations

Sub-sections V.1. and V.2. show how both flavors of the Runge-Kutta 4 code are verified through convergence tests. In the first case, the numeric solution converges to the analytic solution. In the second cases the numeric solutions converge to "real" solutions which are unknown and cannot be derived analytically.

In addition to these direct verification techniques there are other checks on other aspects of the code. For instance, the accepted age of the universe derived using similar simulations can serve as a check on the constant dark energy density universe with $\rho_{rd} = 0.00003$, $\rho_{md} = 0.27$, $\rho_{dd} = 0.73$ \cite{6}. Our numerical solution predicts the universe began 13.6 years ago, which agrees with WMAP simulation results of $13.7 \pm 0.2$ billion years\cite{2}.

Continuity also serves as a check on the initial conditions for $\phi$ and $\dot{\phi}$. As shown in Section VI, the program universe produces graphs of $a(t), \phi(t)$ and $\dot{\phi}(t)$. The continuity and smoothness of these graphs around $t = 0$ and their approximation of a constant dark energy density universe indicate that the initial conditions are correctly derived.
VI. Results

At the outset of this project two guiding questions were posed: how does a universe with \( \rho_d = \textit{constant} \) compare to one where \( \rho_d = \rho_\phi(t) \) and what restrictions on \( \kappa \) and \( \alpha \) can be imposed using the constraint of the universe’s age. In the interest of clarity we examine the latter question first.

Current empirical data constrains the age of the universe to be greater than 12.7 Gyr\(^2\). Simulations of the universe with a variable dark energy density yield different values for the age of the universe depending on the values of \( \kappa \) and \( \alpha \) chosen to describe the rolling potential. The program \textit{restrictor} simulates the evolution of universe backwards in time and calculates the time of the big bang, where \( t = 0 \) corresponds to the present. It does this for a large range of \( \kappa \) and \( \alpha \) and outputs the resulting times as a function of \( \kappa \) and \( \alpha \). A color plot of this data is shown in Fig. 3 below.

![Effect of Kappa and Alpha on Age of Universe](image)

Fig. 3.– A plot of the Big Bang time against \( \kappa \) and \( \alpha \). Acceptable values of these two parameters lies in the green and blue spaces.

This plot shows that in order to agree with observational data \( \kappa \) and \( \alpha \) must have values corresponding to the green and blue regions. Values in the yellow and red areas give rise to universes that evolve much more rapidly than the one in which we live.
In addition to examining the restrictions on $\kappa$ with $\alpha$, we can also look at the dependence of the age of the universe on both parameters. The program dependence does just this and creates plots of the time since Big Bang vs. $\kappa$ with $\alpha$ constant and time since Big Bang vs. $\alpha$ with $\kappa$ held constant. Fig. 4 and Fig. 5 show these results, respectively. Interesting results are found in both Figures. From Fig. 4 we see that at large $\kappa$ the age of the universe tends towards constancy. It also appears that for $\kappa < 75$ larger values of $\alpha$ correspond to older universes while when $\kappa > 75$ larger values of $\alpha$ correspond to younger universes. Also, in Fig. 5, as $\alpha$ increases the age of the universe also tends towards constancy. These plots indicate the conditions for which variations in $\kappa$ with $\alpha$ translate into small differences in the universe’s age and when they translate into large differences. The critical points appear to be at $\kappa = 75$ and $\alpha = 0.5$.

**Fig. 4.** Shows the dependence of the age of the universe on $\alpha$ for different values of $\kappa$.

**Fig. 5.** Shows the dependence of the age of the universe on $\kappa$ for different values of $\alpha$.

**Energy Densities for a Dynamic Dark Energy Universe**

**Energy Densities for a Constant Dark Energy Universe**

**Fig. 6.** The matter, radiation and dark energy densities as a fraction of the critical density, plotted as a function of log (time since Big Bang) for a variable dark energy density.

**Fig. 7.** The matter, radiation and dark energy densities as a fraction of the critical density, plotted as a function of log (time since Big Bang) for a constant dark energy density.
With the acceptable ranges of $\kappa$ and $\alpha$ determined using restrictor, different parameters of the universe can be examined as a function of time for representative values of $\kappa$ and $\alpha$. This is done using the program universe. The results for the density contributions of various constituents are shown in Fig. 6. Fig. 7 shows the same quantities plotted for a constant dark energy density universe using the program constantrp. These figures show that at late times the densities of matter, energy and radiation are similar for both models of dark energy. However, at early times the models diverge. In the constant dark energy density universe matter comes to dominate at intermediate times and radiation at very early times. By contrast, the dynamic dark energy model shows that at very early times dark energy again dominates instead of radiation. Fig. 8 and 9 show this difference more clearly due to their y-axes being a log plot of the density parameters.

The domination of dark energy again at early times may be tied to the inflationary scenario, which correctly explains the homogeneity and density properties of the universe. At a more basic level this result can be understood in terms of the equation of motion for $\phi$ (Eq. 10). This is basically the equation for a damped harmonic oscillator. Interestingly, the damping coefficient is $3H$, meaning that resistance to the motion of $\phi$ is proportional to the expansion rate. Looking at Fig. 10 we see that this system must be over-damped. As time passes $\phi$ seems to slowly roll towards a natural value such that as $t \to \infty$, $\rho_\phi \to 0$. This agrees with the results derived by Peebles and
chosen such that the age of the universe for a variable dark energy density approximated that for a constant dark energy density.

Going back to Fig. 3, we see that a non-trivial set of values for \( \kappa \) and \( \alpha \) yield the same age of the universe. In Fig. 11 it is apparent that for \( \kappa = 300 \) and \( \alpha = 0.5 \) the two models of the universe are identical at late times, but this may not be the case for other values of \( \kappa \) and \( \alpha \), even if those values yield the same age of the universe. In order to resolve this question, plots of \( a(t) \) were created for values of \( \kappa \) and \( \alpha \) that created a universe where the current age is 13.7 Gyr. These plots are shown overlaid in Fig. 12.

Counter-intuitively, different values of \( \kappa \) and \( \alpha \) do not create universes with different dynamics, with the constraint that the age of the universe is held constant. If dark energy is variable in the manner assumed this result indicates it may be difficult to determine exactly what form \( V(\phi) \) takes. This is a point that demands further investigation from the stand-point of observational verification.

VII. CONCLUSIONS

In this paper we examine the differences between a universe with a constant dark energy density and one with a variable dark energy density whose variability is governed by Equations 8, 9 and 10. At late times these two models agree, as \( \rho_{\phi} \) tends toward a constant value. However, at early times these two models predict widely varying behavior in the relative density contributions of matter, radiation and dark energy. Based on the observational constraints on the age of the universe, restrictions were placed on \( \kappa \) and \( \alpha \). The acceptable values for these parameters are located in the blue and green areas on Fig. 3. We found that when the age of the

---

**Ratra**. We see from analyzing the energy density plots and \( \phi \) itself that at late times the energy density tends toward a constant value, and the universe with a dynamic dark energy density is well-approximated by one with a constant dark energy density. Our simulations indicate that this is also true for the universe’s scale factor. Fig. 11 shows a plot of \( a \) for a constant and variable dark energy density; the two lines are identical at late times. For this figure \( \kappa \) and \( \alpha \) were

---
universe is held constant, any values of $\kappa$ and $\alpha$ chosen will result in a universe with the same dynamical behavior.

From this investigation we have determined that the universe could in fact have a variable dark energy density, and that, based on observational constraints, only certain values of $\kappa$ and $\alpha$ are allowed. However, many questions remain open, including explanation of the universe's behavior at very early times with a dynamic dark energy. Future investigations that accurately model this behavior can be compared with CMB data and may place additional restrictions $\kappa$ and $\alpha$ or eliminate this model entirely. One surety is that the power law model of a dynamic dark energy warrants further in-depth investigation and work towards observational verification.
VIII. REFERENCES


IX. APPENDIX A

**TABLE OF PROGRAMS**

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test</td>
<td>Compares the analytic solution to the numeric solution graphically.</td>
</tr>
<tr>
<td>2</td>
<td>Minerr</td>
<td>Shows the 4th order convergence of the 1-equation Runge-Kutta numeric solution to the analytic solution for decreasing time steps.</td>
</tr>
<tr>
<td>3</td>
<td>Minerr2</td>
<td>Shows the convergence of the 3-equation 4th Order Runge-Kutta to the real, unknown solution for decreasing time steps.</td>
</tr>
<tr>
<td>4</td>
<td>Constantrp</td>
<td>Simulates the universe with a constant dark energy density using the ideal time-step found using minerr.</td>
</tr>
<tr>
<td>5</td>
<td>Universe</td>
<td>Simulates the universe with a variable dark energy density given values of κ and α. May be altered to use the variable time step h = ho/H.</td>
</tr>
<tr>
<td>6</td>
<td>Restrictor</td>
<td>Finds the age of the universe for various values of κ and α and prints the results in [x y z] format.</td>
</tr>
<tr>
<td>7</td>
<td>Dependence</td>
<td>Creates plots of the universe’s age as a function of either κ or α.</td>
</tr>
<tr>
<td>8</td>
<td>k_and_a_value</td>
<td>Prints κ and α. values that allow for a universe that is 13.7 ± 0.03 billion years old.</td>
</tr>
</tbody>
</table>

*Note on the Cover Image:* The figure found on the cover is a surface mesh representation of the effect that κ and α have on the age of the universe. This image was plotted with the data from the restrictor.c code and contains the same data found in Figure (3).
/test.c
#include<stdio.h>
#include<math.h>

/* This program tests the Runge-Kutta 4 code for rho=rhomat ONLY and compares that to */
/* The analytic solution */

/* This function computes the second time derivative of phi */
float dadt(float a)
{
    float pi=3.1415926;
    float rhomat=0.27;
    float deriv;
    deriv=a*pow(8.0*pi/3.0*rhomat*pow(a,-3),0.5);
    return deriv;
}

float analytic(float time)
{
    float pi=3.1415926;
    float rhomat=0.27;
    float a;
    a=pow(1.5*pow(8*pi*rhomat/3,0.5)*time+1.0,2.0/3.0);
    return a;
}

int main(void)
{
    /* Initialize the variables. For ki 0<=i<=3, i is the derivative of psi */
    float a1, a2;
    float i, ka[4];
    float chartime=41.6195; /* billions of years */
    float h=0.01;
    ka[0]=0;
    ka[1]=0;
    ka[2]=0;
    ka[3]=0;

    a1=1;
    FILE *rout;
    rout=fopen("test.dat","w");
    i=0;
    fprintf(rout,"%f\t %f\t %f\n",i,a1,analytic(i));
    while(i<=1)
    {
        /* Runge-Kutta Order 4 for a */
        ka[0]=h*dadt(a1);
        ka[1]=h*dadt(a1+0.5*ka[0]);
        ka[2]=h*dadt(a1+0.5*ka[1]);
        ka[3]=h*dadt(a1+ka[2]);
/* Calculate new values, both numeric and analytic*/

i=i+h;
a1=a1+ka[0]/6+ka[1]/3+ka[2]/3+ka[3]/6;
a2=analytic(i);

/* Print Results! */

fprintf(rout,"%f	%f	%f\n",i*chartime,a1,a2);
}

/* redo everything backwards in time */

h=-h;
i=0;
a1=1;

/* Do Runge-Kutta 4 again */

while(a1>0.05 && a2>0.05)
{
    ka[0]=h*dadt(a1);
    ka[1]=h*dadt(a1+0.5*ka[0]);
    ka[2]=h*dadt(a1+0.5*ka[1]);
    ka[3]=h*dadt(a1+ka[2]);

    /* Calculate new values */

    i=i+h;
a1=a1+ka[0]/6+ka[1]/3+ka[2]/3+ka[3]/6;
a2=analytic(i);

    /* Print Results! */

    fprintf(rout,"%f	%f	%f\n",i*chartime,a1,a2);
}

close(rout);
return;
}

/* The end */
/minerr.c

#include<stdio.h>
#include<math.h>

/* This program tests the Runge-Kutta 4 code for rho=rhomat ONLY and compares that to */
/* The analytic solution */

/* This function computes the second time derivative of phi */
float dadt(float a)
{
    float pi=3.1415926;
    float rhomat=0.27;
    float deriv;
    deriv=a*pow(8.0*pi/3.0*rhomat*pow(a,-3),0.5);
    return deriv;
}

float analytic(float time)
{
    float pi=3.1415926;
    float rhomat=0.27;
    float a;
    a=pow(1.5*pow(8*pi*rhomat/3,0.5)*time+1.0,2.0/3.0);
    return a;
}

int main(void)
{
    /* Initialize the variables. For ki 0<=i<=3, i is the derivative of psi */
    float a1, a2;
    float i, ka[4];
    float chartime=41.6195; /* billions of years */
    float h, err1, err2;
    h=2*pow(10,-5);
    FILE *rout;
    rout=fopen("testh.dat","w");

    while(h<=0.015)
    {
        printf("working...%f complete\n",h*1000);

        a1=1;
        i=0;
        ka[0]=0;
        ka[1]=0;
        ka[2]=0;
        ka[3]=0;

        while(i*chartime<=40)
        {
            /* Runge-Kutta Order 4 for a */
        }
    }
}
ka[0]=h*dadt(a1);
ka[1]=h*dadt(a1+0.5*ka[0]);
ka[2]=h*dadt(a1+0.5*ka[1]);
ka[3]=h*dadt(a1+ka[2]);

/* Calculate new values, both numeric and analytic*/

i=i+h;
a1=a1+ka[0]/6+ka[1]/3+ka[2]/3+ka[3]/6;
a2=analytic(i);
}

erl1=a1-a2;

/* redo everything backwards in time */

h=-h;
i=0;
a1=1;
ka[0]=0;
ka[1]=0;
ka[2]=0;
ka[3]=0;

/* Do Runge-Kutta 4 again */

while(a1>=0.1)
{
    ka[0]=h*dadt(a1);
    ka[1]=h*dadt(a1+0.5*ka[0]);
    ka[2]=h*dadt(a1+0.5*ka[1]);
    ka[3]=h*dadt(a1+ka[2]);

    /* Calculate new values */

    i=i+h;
a1=a1+ka[0]/6+ka[1]/3+ka[2]/3+ka[3]/6;
a2=analytic(i);
}

erl2=a1-a2;
fprintf(rout,"%f	%f	%f\n",log10(-h),pow(err1*pow(10,5),2.0),pow(err2*pow(10,5),2.0));
h=-h;
h=h+2*pow(10,-5);
}
close(rout);
return;

/* The end */
/* This program computes the future evolution of the universe given the conditions that */
/* space-time is normally flat and the dark energy density is created by a rolling */
/* potential function of psi (see Peebles and Ratra). Output is dumped into three */
/* files, one for a and H (Hubble's "constant"), one for values of psi and its derivs, */
/* and one for the energy density of various forms of energy. Enjoy! */

static double pi=3.1415926;
static double rhorad=0.00003;
static double rhomat=0.27;
static double rhodark=0.73-0.00003;
double kappa;
double alpha;
static double G=1;
static double mp=1; /* equals G to the 0.5 power */
static double chartime=41.6195; /* billions of years */
double h=0.000001;

/* This function computes the second time derivative of phi */

double func(double psi0, double psi1, double a)
{
    double adot, rphi, result;
    rphi=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
adot=a*pow(8.0*pi/3.0*(rhorad*pow(a,-4)+rhomat*pow(a,-3)+rphi),0.5);
result=kappa*alpha*0.5*pow(mp,2)*pow(psi0,-alpha-1)-3*adot/a*psi1;
    return result;
}

/* This function is the friedmann equation. The time rate of change of a is computed */

double dadt(double a, double psi0, double psi1)
{
    double deriv, rphi;
    rphi=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    deriv=a*pow(8.0*pi/3.0*(rhorad*pow(a,-4)+rhomat*pow(a,-3)+rphi),0.5);
    return deriv;
}

/* This function computes the energy density of the rolling potential */

double rhophi(double psi0, double psi1)
{
    double result;
    result=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    return result;
}
int main(void)
{
    /* Initialize the variables. For ki 0<=i<=3, i is the derivative of psi */
    double a, rhoinit, psi[3], psi2p;
    double i, k0[4], k1[4], k2[4], ka[4];
    double pcrit, initpcrit, rad, mat, H;
    double psilast, trials, alast;
    
    /* Set initial conditions */
    kappa=300;
    alpha=0.5;
    trials=0;
    rhoinit=rhodark;
    a=1;
    psi[0]=pow(pow(mp,4)*kappa/(32*pi*rhoinit),1/alpha);
    psi[1]=0;
    psi[2]=kappa*alpha*pow(mp,2)*0.5*pow(psi[0],-alpha-1);
    H=dadt(a,psi[0],psi[1])/a;
    initpcrit=3*pow(H,2)/(8*pi*G);
    pcrit=initpcrit;
    rad=rhorad*pow(a,-4);
    mat=rhomat*pow(a,-3);
    
    /* Initialize the output files which are grouped for convenience */
    FILE *rout;
    rout=fopen("testhpsi.dat","w");
    
    /* Print initial values of everything */
    i=0;
    while(h<=pow(10,-5))
    {
        trials=trials+1;
        printf("Working...
");

        while(i*chartime<=20)
        {
            /* Runge-Kutta Order 4 for psi, psi-dot and a */
            k0[0]=h*psi[1];
            k1[0]=h*psi[2];
            ka[0]=h*dadt(a,psi[0],psi[1]);
            k0[1]=h*(psi[1]+0.5*k1[0]);
            k1[1]=h*(psi[2]+0.5*k2[0]);
            ka[1]=h*dadt(a+0.5*ka[0],psi[0]+0.5*k0[0],psi[1]+0.5*k1[0]);
            k0[2]=h*(psi[1]+0.5*k1[1]);
            k1[2]=h*(psi[2]+0.5*k2[1]);
            ka[2]=h*dadt(a+0.5*ka[1],psi[0]+0.5*k0[1],psi[1]+0.5*k1[1]);
        }
    }
}
k0[3]=h*(psi[1]+k1[2]);  
ka[3]=h*dadt(a+ka[2],psi[0]+k0[2],psi[1]+k1[2]);

/* Calculate values for all the interesting and cool stuff */

alast=a;  
psilast=psi[0];  
a=a+ka[0]/6+ka[1]/3+ka[2]/3+ka[3]/6;  
psi[0]=psi[0]+k0[0]/6+k0[1]/3+k0[2]/3+k0[3]/6;  
psi[1]=psi[1]+k1[0]/6+k1[1]/3+k1[2]/3+k1[3]/6;  
psi[2]=func(psi[0],psi[1],a);

/* Print Results! */
i=i+h;
}
fprintf(rout,"%f	%f	%f",
log10(h),
log10(fabs(psi[0]-psilast)),
log10(fabs(a-alast)));

/* redo everything backwards in time */
h=-h;
i=0;

/* reset initial values */
rhoinit=rhodark;
a=1;
psi[0]=pow(pow(mp,4)*kappa/(32*pi*rhoinit),1/alpha);
psi[1]=0;
psi[2]=kappa*alpha*pow(mp,2)*0.5*pow(psi[0],-alpha-1);
H=dadt(a,psi[0],psi[1])/a;
prcr=initpcr;
rad=rhorad*pow(a,-4);
mat=rhomat*pow(a,-3);

/* Do Runge-Kutta 4 again */
while(a>0.1 && a<=2.0)
{
   /* Runge-Kutta Order 4 for psi and a */
   k0[0]=h*psi[1];  
k1[0]=h*psi[2];  
ka[0]=h*dadt(a,psi[0],psi[1]);

   k0[1]=h*(psi[1]+0.5*k1[0]);  
k1[1]=h*(psi[2]+0.5*k2[0]);  
ka[1]=h*dadt(a+0.5*ka[0],psi[0]+0.5*k0[0],psi[1]+0.5*k1[0]);

   k0[2]=h*(psi[1]+0.5*k1[1]);  
k1[2]=h*(psi[2]+0.5*k2[1]);  
ka[2]=h*dadt(a+0.5*ka[1],psi[0]+0.5*k0[0],psi[1]+0.5*k1[1]);
\[ k_0[3] = h \cdot (\psi[1] + k_1[2]) \]
\[ k_1[3] = h \cdot (\psi[2] + k_2[2]) \]
\[ k_a[3] = h \cdot d\!a/dt(a + k_a[2], \psi_0[0] + k_0[2], \psi_1[1] + k_1[2]) \]

/* Calculate values for all the interesting and cool stuff */
alast = a;
psilast = \psi[0];
a = a + k_a[0]/6 + k_a[1]/3 + k_a[2]/3 + k_a[3]/6;
\psi[0] = \psi[0] + k_0[0]/6 + k_0[1]/3 + k_0[2]/3 + k_0[3]/6;
\psi[1] = \psi[1] + k_1[0]/6 + k_1[1]/3 + k_1[2]/3 + k_1[3]/6;
\psi[2] = func(\psi_0[0], \psi_1[1], a);

/* Print Results! */
i = i + h;

fprintf(rout, "%f \t %f\n", log10(fabs(\psi[0] - psilast)), log10(fabs(a - alast)));
h = -h;
h = h + log10(trials) * pow(10, -6);
}
close(rout);
printf("Complete\n");
return;
}

/* The end */
//constantrp.c
#include<stdio.h>
#include<math.h>

/* for parameters rad, mat and dark this program runs the scale of */
/* the universe forward and backward in time for a constant dark energy density */
/* This function computes the time derivative of the scale factor */

float fried(float a)
{
    float rad, mat, dark;
    float pi, deriv;
    pi=3.14159;
    rad=3*10.0,-5.0;
    mat=0.27;
    dark=0.73-rad;

    deriv=a*8.0*pi/3.0*(rad*pow(a,-4.0)+mat*pow(a,-3.0)+dark),0.5);
    return deriv;
}

int main(void)
{
    /* Initialize variables and output files */
    float a, h, i, k[4], chartime, H, mat, rad, dark, pcrit, pi;

    FILE *rout, *dout;
    rout=fopen("acon.dat","w");
    dout=fopen("denscon.dat","w");

    pi=3.14159;
    chartime=41.6195; /* billions of years */
    h=0.003;

    /* Set and print initial values */

    i=0;
    a=1;
    rad=3*10.0,-5.0;
    mat=0.27;
    dark=0.73-rad;
    H=fried(a)/a;
    pcrit=3*pow(H,2)/(8*pi);
    k[0]=0;
    k[1]=0;
    k[2]=0;
    k[3]=0;

    fprintf(rout,"%f\t%f\t%f\n",i*chartime,a,H);
    fprintf(dout,"%f\t%f\t%f\t%f\n",i*chartime,rad*pow(a,-4)/pcrit,mat*pow(a,-3)/pcrit,pcrit);

    /* Do runge-kutta forward in time and print results*/

while(i*chartime<=40)
{
    k[0]=h*fried(a);
    k[1]=h*fried(a+0.5*k[0]);
    k[2]=h*fried(a+0.5*k[1]);
    k[3]=h*fried(a+k[2]);
    a=a+k[0]/6+k[1]/3+k[2]/3+k[3]/6;

    i=i+h;
    H=fried(a)/a;
    pcrit=3*pow(H,2)/(8*pi);
    fprintf(rout,"%f\t%f\t%f\n",i*chartime,a,H);
    fprintf(dout,"%f\t%f\t%f\t%f\n",i*chartime,rad*pow(a,-4)/pcrit,mat*pow(a,-3)/pcrit,dark/pcrit);
}

/* Reset and go backwards in time */

h=-h;
 i=0;
 a=1;
 rad=3*pow(10.0,-5.0);
 mat=0.27;
 dark=0.73-rad;
 H=fried(a)/a;
 k[0]=0;
 k[1]=0;
 k[2]=0;
 k[3]=0;

while(a>=0.05)
{
    k[0]=h*fried(a);
    k[1]=h*fried(a+0.5*k[0]);
    k[2]=h*fried(a+0.5*k[1]);
    k[3]=h*fried(a+k[2]);
    a=a+k[0]/6+k[1]/3+k[2]/3+k[3]/6;

    i=i+h;
    H=fried(a)/a;
    pcrit=3*pow(H,2)/(8*pi);
    fprintf(rout,"%f\t%f\t%f\n",i*chartime,a,H);
    fprintf(dout,"%f\t%f\t%f\t%f\n",i*chartime,rad*pow(a,-4)/pcrit,mat*pow(a,-3)/pcrit,dark/pcrit);
}

fclose(rout);
fclose(dout);
printf("Complete\n");
return;
/universe.c
#include<stdio.h>
#include<math.h>

/* This program computes the future evolution of the universe given the conditions that */
/* space-time is normally flat and the dark energy density is created by a rolling */
/* potential function of psi (see Peebles and Ratra). Output is dumped into three */
/* files, one for a and H (Hubble's "constant"), one for values of psi and its derivs, */
/* and one for the energy density of various forms of energy. Enjoy! */

static float pi=3.1415926;
static float rhorad=0.00003;
static float rhomat=0.27;
static float rhodark=0.73-0.00003;
float kappa;
float alpha;
static float G=1;
static float mp=1;  /* equals G to the 0.5 power */
static float chartime=41.6195;  /* billions of years */
static float h=0.003;

/* This function computes the second time derivative of phi */

float func(float psi0, float psi1, float a)
{
    float adot, rphi, result;
    rphi=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    adot=a*pow(8.0*pi/3.0*(rhorad*pow(a,-4)+rhomat*pow(a,-3)+rphi),0.5);
    result=kappa*alpha*0.5*pow(mp,2)*pow(psi0,-alpha-1)-3*adot/a*psi1;
    return result;
}

/* This function is the friedmann equation. The time rate of change of a is computed */

float dadt(float a, float psi0, float psi1)
{
    float deriv, rphi;
    rphi=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    deriv=a*pow(8.0*pi/3.0*(rhorad*pow(a,-4)+rhomat*pow(a,-3)+rphi),0.5);
    return deriv;
}

/* This function computes the energy density of the rolling potential */

float rhophi(float psi0, float psi1)
{
    float result;
    result=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    return result;
}
int main(void)
{
    /* Initialize the variables. For ki 0<=i<=3, i is the derivative of psi */
    float a, rhoinit, psi[3], psi2p;
    float i, k0[4], k1[4], k2[4], ka[4];
    float pcrit, initpcrit, rad, mat, H;
    /* Set initial conditions */
    kappa=100;
    alpha=0.5;
    rhoinit=rhodark;
    a=1;
    psi[0]=pow(pow(mp,4)*kappa/(32*pi*rhoinit),1/alpha);
    psi[1]=0;
    psi[2]=kappa*alpha*pow(mp,2)*0.5*pow(psi[0],-alpha-1);
    H=dadt(a,psi[0],psi[1])/a;
    initpcrit=3*pow(H,2)/(8*pi*G);
    pcrit=initpcrit;
    rad=rhorad*pow(a,-4);
    mat=rhomat*pow(a,-3);
    /* Initialize the output files which are grouped for convenience */
    FILE *rout, *pout, *dout;
    rout=fopen("a_ka.dat","w");
    pout=fopen("psi_ka.dat","w");
    dout=fopen("dens_ka.dat","w");
    /* Print initial values of everything */
    i=0;
    fprintf(rout,"%f	%f	%f\n",i,a,H);
    fprintf(pout,"%f	%f	%f	%f\n",i,psi[0],psi[1],psi[2]);
    fprintf(dout,"%f\t%f\t%f\t%f\t%f\n",i,mat,rad,rhophi(psi[0],psi[1]),mat+rad+rhophi(psi[0],psi[1]));

    while(i*chartime<=40)
    {
        /* Runge-Kutta Order 4 for psi , psi-dot and a */
        k0[0]=h*psi[1];
        k1[0]=h*psi[2];
        k2[0]=h*func(psi[0],psi[1],a);
        ka[0]=h*dadt(a,psi[0],psi[1]);
        k0[1]=h*(psi[1]+0.5*k1[0]);
        k1[1]=h*(psi[2]+0.5*k2[0]);
        k2[1]=h*func(psi[0]+0.5*k0[0],psi[1]+0.5*k1[0],a+0.5*ka[0]);
        ka[1]=h*dadt(a+0.5*ka[0],psi[0]+0.5*k0[0],psi[1]+0.5*k1[0]);
\[ \begin{align*} 
k_0[2] &= h \cdot (\psi[1] + 0.5 \cdot k_1[1]) ; \\k_1[2] &= h \cdot (\psi[2] + 0.5 \cdot k_2[1]) ; \\k_2[2] &= h \cdot \text{func}(\psi[0] + 0.5 \cdot k_0[1], \psi[1] + 0.5 \cdot k_1[1], a + 0.5 \cdot k_2[1]) ; \\k_2[3] &= h \cdot \text{dadt}(a + 0.5 \cdot k_2[1], \psi[0] + 0.5 \cdot k_0[1], \psi[1] + 0.5 \cdot k_1[1]) ; \\
k_0[3] &= h \cdot (\psi[1] + k_1[3]) ; \\k_1[3] &= h \cdot (\psi[2] + k_2[3]) ; \\k_2[3] &= h \cdot \text{func}(\psi[0] + k_0[2], \psi[1] + k_1[2], a + k_2[2]) ; \\k_2[3] &= h \cdot \text{dadt}(a + k_2[2], \psi[0] + k_0[2], \psi[1] + k_1[2]) ; \\
\end{align*} \]

/* Calculate values for all the interesting and cool stuff */

\[ \begin{align*} 
a &= a + k_0[1]/6 + k_1[1]/3 + k_2[1]/3 + k_3[1]/6 ; \\
\psi[0] &= \psi[0] + k_0[0]/6 + k_0[1]/3 + k_0[2]/3 + k_0[3]/6 ; \\psi[1] &= \psi[1] + k_1[0]/6 + k_1[1]/3 + k_1[2]/3 + k_1[3]/6 ; \\
\psi[2] &= \text{func}(\psi[0], \psi[1], a) ; \\H &= \text{dadt}(a, \psi[0], \psi[1])/a ; \\pcrit &= 3 \cdot \text{pow}(H, 2)/(8 \cdot \text{pi} \cdot G) ; \\
\text{rad} &= \text{rhorad} \cdot \text{pow}(a, -4) \cdot \text{initpcrit}/\text{pcrit} ; \\text{mat} &= \text{rhomat} \cdot \text{pow}(a, -3) \cdot \text{initpcrit}/\text{pcrit} ; \\
\end{align*} \]

/* Print Results! */

\[ \begin{align*} 
i &= i + h ; \\
\text{fprintf}(&rout, "\%f\t \%f\t \%f\n", i \cdot \text{chartime}, a, H) ; \\
\text{fprintf}(&pout, "\%f\t \%f\t \%f\t \%f\n", i \cdot \text{chartime}, \psi[0], \psi[1], \psi[2]) ; \\
\text{fprintf}(&dout, "\%f\t \%f\t \%f\t \%f\t \%f\n", i \cdot \text{chartime}, \text{mat}, \text{rad}, \text{rhophi}(\psi[0], \psi[1]) \cdot \text{initpcrit}/\text{pcrit}, \text{mat} + \text{rad} + \text{rhophi}(\psi[0], \psi[1]) \cdot \text{initpcrit}/\text{pcrit}) ; \\
\end{align*} \]

/* redo everything backwards in time */

\[ \begin{align*} 
h &= -h ; \\
i &= 0 ; \\
\end{align*} \]

/* reset initial values */

\[ \begin{align*} 
\text{rhoinit} &= \text{rhodark} ; \\a &= 1 ; \\
\psi[0] &= \text{pow}((\text{pow}(m, 4) \cdot \kappa) / (32 \cdot \text{pi} \cdot \text{rhoinit}), 1/\alpha) ; \\
\psi[1] &= 0 ; \\
\psi[2] &= \kappa \cdot \alpha \cdot \text{pow}(m, 2) \cdot 0.5 \cdot \text{pow}(\psi[0], -\alpha - 1) ; \\
\H &= \text{dadt}(a, \psi[0], \psi[1])/0.5 \cdot \text{pow}(\psi[0], -\alpha - 1) ; \\pcrit &= \text{initpcrit} ; \\
\text{rad} &= \text{rhorad} \cdot \text{pow}(a, -4) ; \\
\text{mat} &= \text{rhomat} \cdot \text{pow}(a, -3) ; \\
\end{align*} \]

/* Do Runge-Kutta 4 again */

\[ \begin{align*} 
\text{while}(a > 0.05) 
\{ 
\text{/* Runge-Kutta Order 4 for psi and a */} 
\} 
\]
k0[0]=h*psi[1];
k1[0]=h*psi[2];
k2[0]=h*func(psi[0],psi[1],a);
ka[0]=h*dadt(a,psi[0],psi[1]);

k0[1]=h*(psi[1]+0.5*k0[0]);
k1[1]=h*(psi[2]+0.5*k2[0]);
k2[1]=h*func(psi[0]+0.5*k0[0],psi[1]+0.5*k1[0],a+0.5*ka[0]);
ka[1]=h*dadt(a+0.5*ka[0],psi[0]+0.5*k0[0],psi[1]+0.5*k1[0]);

k0[2]=h*(psi[1]+0.5*k1[1]);
k1[2]=h*(psi[2]+0.5*k2[1]);
k2[2]=h*func(psi[0]+0.5*k0[1],psi[1]+0.5*k1[1],a+0.5*ka[1]);
ka[2]=h*dadt(a+0.5*ka[1],psi[0]+0.5*k0[1],psi[1]+0.5*k1[1]);

k0[3]=h*(psi[1]+k0[1]);
k2[3]=h*func(psi[0]+k0[2],psi[1]+k1[2],a+ka[2]);
ka[3]=h*dadt(a+ka[2],psi[0]+k0[2],psi[1]+k1[2]);

/* Calculate values for all the interesting and cool stuff */

a=a+ka[0]/6+ka[1]/3+ka[2]/3+ka[3]/6;
psi[0]=psi[0]+k0[0]/6+k0[1]/3+k0[2]/3+k0[3]/6;
psi[1]=psi[1]+k1[0]/6+k1[1]/3+k1[2]/3+k1[3]/6;
psi[2]=func(psi[0],psi[1],a);
H=dadt(a,psi[0],psi[1])/a;
pcrit=3*pow(H,2)/(8*pi*G);
rad=rhorad*pow(a,-4)*initpcrit/pcrit;
mat=rhomat*pow(a,-3)*initpcrit/pcrit;

/* Print Results! */

i=i+h;
fprintf(rout,"%f\t%f\t%f\n",i*chartime,a,H);
fprintf(pout,"%f\t%f\t%f\n",i*chartime,psi[0],psi[1],psi[2]);
fprintf(dout,"%f\t%f\t%f\t%f\n",i*chartime,mat,rad,rhophi(psi[0],psi[1])*initpcrit/pcrit,mat+rad+rhophi(psi[0],psi[1])*initpcrit/pcrit);
}
close(rout);
close(pout);
close(dout);
printf("%f\n",i*chartime);
return;

}
//restrictor.c
#include<stdio.h>
#include<math.h>

/* This program runs the universe backwards in time for different values of */
/* kappa and alpha and prints their values if the universe began more than 12.7 */
/* Billion years ago. Must set initial kappa and alpha, while loop limits, and */
/* most importantly the step size of kappa and alpha (can be variable) */

static double pi=3.1415926;
static double rhorad=0.00003;
static double rhomat=0.27;
static double rhodark=0.73-0.00003;
static double kappa;
static double alpha;
static double G=1;
static double mp=1;    /* equals G to the 0.5 power */
static double chartime=41.6195; /* billions of years */
static double h=-0.00001;

/* This function computes the second time derivative of phi */
double func(double psi0, double psi1, double a)
{
    double adot, rphi, result;
    rphi=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    adot=a*pow(8.0*pi/3.0*(rhorad*pow(a,-4)+rhomat*pow(a,-3)+rphi),0.5);
    result=kappa*alpha*0.5*pow(mp,2)*pow(psi0,-alpha-1)-3*adot/a*psi1;
    return result;
}

/* This function is the friedmann equation. The time rate of change of a is */
/* computed */
double dadt(double a, double psi0, double psi1)
{
    double deriv, rphi;
    rphi=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    deriv=a*pow(8.0*pi/3.0*(rhodark*pow(a,-4)+rhomat*pow(a,-3)+rphi),0.5);
    return deriv;
}

/* This function computes the energy density of the rolling potential */

double rhophi(double psi0, double psi1)
{
    double result;
    result=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    return result;
}

int main(void)
{
/* Initialize the variables. For ki 0<=i<=2, i is the derivative of psi */

double a, rhoinit, psi[3], psi2p;
double i, k0[4], k1[4], k2[4], ka[4];
double pcrit, rad, mat, H;
double atrials, ktrials;
int q;

/* Set initial conditions and creates an output file*/
FILE *rout;
rout=fopen("restrict.dat","w"); /* Initial kappa an alpha values */
kappa=50.0;
alpha=0.002;
ktrials=0;
atrials=0;

/* cycle through the different values of kappa */
while(kappa<=500.0)
{
    printf("Complete %f kappas\n",ktrials); /* progress report */
    ktrials=ktrials+1;

    /* cycle through the different values for alpha, for each kappa */
    while(alpha<=5.0)
    {
        printf("Completed %f alphas\n",atrials);
        atrials=atrials+1;

        /* set initial conditions */
        i=0.0;
rhoinit=rhodark;
a=1.0;
psi[0]=pow(pow(mp,4.0)*kappa/(32*pi*rhoinit),1/alpha);
psi[1]=0;
psi[2]=kappa*alpha*pow(mp,2.0)*0.5*pow(psi[0],-alpha-1.0);
H=dadt(a,psi[0],psi[1])/a;
pcrit=3*pow(H,2.0)/(8*pi*G);
rad=rhorad*pow(a,-4.0);
mat=rhomat*pow(a,-3.0);

for(q=0;q<=3;q++)
{
    k0[q]=0;
    k1[q]=0;
    ka[q]=0;
}

    /* Do Runge-Kutta 4 for the given kappa and alpha */
    while(a>0.01 && a<1.5 && i*chartime>-40.0) /* bound the function */
    {
        k0[0]=h*psi[1];
k1[0]=h*psi[2];
        ka[0]=h*dadt(a,psi[0],psi[1]);
        k0[1]=h*psi[2];
k1[1]=h*psi[3];
        ka[1]=h*dadt(a,psi[1],psi[2]);
k0[1]=h*(psi[1]+0.5*k1[0]);
k1[1]=h*(psi[2]+0.5*k2[0]);
ka[1]=h*dadt(a+0.5*ka[0],psi[0]+0.5*k0[0],psi[1]+0.5*k1[0]);

k0[2]=h*(psi[1]+0.5*k1[1]);
k1[2]=h*(psi[2]+0.5*k2[1]);
ka[2]=h*dadt(a+0.5*ka[1],psi[0]+0.5*k0[1],psi[1]+0.5*k1[1]);

k0[3]=h*(psi[1]+k1[3]);
ka[3]=h*dadt(a+ka[2],psi[0]+k0[2],psi[1]+k1[2]);

/* Calculate values for all the interesting and cool stuff */
a=a+ka[0]/6+ka[1]/3+ka[2]/3+ka[3]/6;
psi[0]=psi[0]+k0[0]/6+k0[1]/3+k0[2]/3+k0[3]/6;
psi[1]=psi[1]+k1[0]/6+k1[1]/3+k1[2]/3+k1[3]/6;
psi[2]=func(psi[0],psi[1],a);
i=i+h;    /* go until one of the boundaries is reached */

/* if the boundary was for time or too small a scale factor, print kappa, alpha */
if(a<0.01 && pow(i*chartime,2)>=161.29)
{
    fprintf(rout,"%f	%f	%f\n",log10(kappa),log10(alpha),i*chartime,a);
}
alpha=alpha+pow(10,atrials*0.01)*0.002;    /* step alpha ahead */
atrials=0;
alpha=0.002;                               /* reset alpha and step kappa ahead */
kappa=kappa+pow(10,ktrials*0.01)*5.0;

printf("Complete\n");
return;
/* The end */
/dependence.c

#include<stdio.h>
#include<math.h>

/* This program computes the future evolution of the universe given the conditions that */
/* space-time is normally flat and the dark energy density is created by a rolling */
/* potential function of psi (see Peebles and Ratra). Output is dumped into three */
/* files, one for a and H (Hubbles "constant"), one for values of psi and its */
/* derivs, */
/* and one for the energy density of various forms of energy. Enjoy! */

static float pi=3.1415926;
static float rhorad=0.00003;
static float rhomat=0.27;
static float rhodark=0.73-0.00003;
static float kappa;
static float alpha;
static float G=1;
static float mp=1; /* equals G to the 0.5 power */
static float chartime=41.6195; /* billions of years */
static float h=-0.003;

/* This function computes the second time derivative of phi */

float func(float psi0, float psi1, float a)
{
    float adot, rphi, result;
    rphi=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    adot=a*pow(8.0*pi/3.0*(rhorad*pow(a,-4)+rhomat*pow(a,-3)+rphi),0.5);
    result=kappa*alpha*0.5*pow(mp,2)*pow(psi0,-alpha-1)-3*adot/a*psi1;
    return result;
}

/* This function is the friedmann equation. The time rate of change of a is */
/* computed */

float dadt(float a, float psi0, float psi1)
{
    float deriv, rphi;
    rphi=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    deriv=a*pow(8.0*pi/3.0*(rhorad*pow(a,-4)+rhomat*pow(a,-3)+rphi),0.5);
    return deriv;
}

/* This function computes the energy density of the rolling potential */

float rhophi(float psi0, float psi1)
{
    float result;
    result=pow(mp,2)/(32*pi)*(pow(psi1,2)+kappa*pow(mp,2)*pow(psi0,-alpha));
    return result;
}
```c
int main(void)
{
    /* Initialize the variables. For ki 0<=i<=2, i is the derivative of psi */
    float a, rhoinit, psi[3], psi2p;
    float i, k0[4], k1[4], k2[4], ka[4];
    float pcrit, rad, mat, H;
    float kappainit, alphainit;
    int q;

    /* Set initial conditions and creates an output file */
    FILE *rout, *pout;
    rout=fopen("adependence.dat","w");
    pout=fopen("kdependence.dat","w");
    kappainit=100.0;
    alphainit=0.1;
    kappa=kappainit;
    alpha=alphainit;

    /* cycle through the different values for alpha, for given kappa */
    while(alpha<=5.0)
    {
        /* set initial conditions */
        i=0.0;
        rhoinit=rhodark;
        a=1.0;
        psi[0]=pow(pow(mp,4.0)*kappa/(32*pi*rhoinit),1/alpha);
        psi[1]=0;
        psi[2]=kappa*alpha*pow(mp,2.0)*0.5*pow(psi[0],-alpha-1.0);
        H=dadt(a,psi[0],psi[1])/a;
        pcrit=3*pow(H,2.0)/(8*pi*G);
        rad=rhorad*pow(a,-4.0);
        mat=rhomat*pow(a,-3.0);

        for(q=0;q<=3;q++)
        {
            k0[q]=0;
            k1[q]=0;
            k2[q]=0;
            ka[q]=0;
        }

        /* Do Runge-Kutta 4 for the given kappa and alpha */
        while(a>0.05 && a<1.5 && i*chartime>-40.0) /* bound the function */
        {
```
\[ k_0[0] = h \cdot \psi[1]; \]
\[ k_1[0] = h \cdot \psi[2]; \]
\[ k_2[0] = h \cdot \text{func}(\psi[0], \psi[1], a); \]
\[ k_{a0}[0] = h \cdot \text{dadt}(a, \psi[0], \psi[1]); \]
\[ k_0[1] = h \cdot (\psi[1] + 0.5 \cdot k_1[0]); \]
\[ k_1[1] = h \cdot (\psi[2] + 0.5 \cdot k_2[0]); \]
\[ k_2[1] = h \cdot \text{func}(\psi[0] + 0.5 \cdot k_0[0], \psi[1] + 0.5 \cdot k_1[0], a + 0.5 \cdot k_{a0}[0]); \]
\[ k_{a1}[0] = h \cdot \text{dadt}(a + 0.5 \cdot k_{a0}[0], \psi[0] + 0.5 \cdot k_0[0], \psi[1] + 0.5 \cdot k_1[0]); \]
\[ k_0[2] = h \cdot (\psi[1] + 0.5 \cdot k_1[1]); \]
\[ k_1[2] = h \cdot (\psi[2] + 0.5 \cdot k_2[1]); \]
\[ k_2[2] = h \cdot \text{func}(\psi[0] + 0.5 \cdot k_0[1], \psi[1] + 0.5 \cdot k_1[1], a + 0.5 \cdot k_{a1}[0]); \]
\[ k_{a2}[0] = h \cdot \text{dadt}(a + 0.5 \cdot k_{a1}[0], \psi[0] + 0.5 \cdot k_0[1], \psi[1] + 0.5 \cdot k_1[1]); \]
\[ k_0[3] = h \cdot (\psi[1] + k_1[3]); \]
\[ k_1[3] = h \cdot (\psi[2] + k_2[3]); \]
\[ k_2[3] = h \cdot \text{func}(\psi[0] + k_0[2], \psi[1] + k_1[2], a + k_{a2}[2]); \]
\[ k_{a3}[0] = h \cdot \text{dadt}(a + k_{a2}[2], \psi[0] + k_0[2], \psi[1] + k_1[2]); \]

/* Calculate values for all the stuff */

\[
\begin{align*}
  a &= a + k_{0o}[0]/6 + k_{1o}[1]/3 + k_{2o}[2]/3 + k_{3o}[3]/6; \\
  \psi[0] &= \psi[0] + k_{0o}[0]/6 + k_{0o}[1]/3 + k_{0o}[2]/3 + k_{0o}[3]/6; \\
  \psi[1] &= \psi[1] + k_{1o}[0]/6 + k_{1o}[1]/3 + k_{1o}[2]/3 + k_{1o}[3]/6; \\
  \psi[2] &= \text{func}(\psi[0], \psi[1], a); \\
  i &= i + h; \quad /* go until one of the boundaries is reached */
\end{align*}
\]

/* once the boundary is reached, print alpha and age */

\[
\begin{align*}
  \text{fprintf}(\text{rout}, "\%f \t \%f \n", \alpha, i*\text{chartime}); \\
  \alpha &= \alpha + 0.01; \quad /* step alpha ahead */
\end{align*}
\]

/* Do the same thing for a constant alpha value */

\[
\begin{align*}
  \kappa &= 10; \\
  \alpha &= \alpha_{\text{init}}; \\
  \text{while}(\kappa \leq 500) \\
  \{ \\
  \quad /* set initial conditions */ \\
  \quad i = 0.0; \\
  \quad \text{rhoinit} = \text{rhodark}; \\
  \quad a = 1.0; \\
  \quad \psi[0] = \text{pow}((\text{pow}((m, 4.0) * \kappa, 32 * \pi * \text{rhoinit}), 1/\alpha)); \\
  \quad \psi[1] = 0; \\
  \quad \psi[2] = \kappa * \alpha * \text{pow}(m, 2.0) * 0.5 * \text{pow}(\psi[0], -\alpha - 1.0); \\
  \quad \text{H} = \text{dadt}(a, \psi[0], \psi[1])/a; \\
  \quad \text{pcrit} = 3 * \text{pow}(H, 2.0)/(8 * \pi * G); \\
  \quad \text{rad} = \text{rhorad} * \text{pow}(a, -4.0); \\
  \quad \text{mat} = \text{rhomat} * \text{pow}(a, -3.0);
\end{align*}
\]
for (q=0; q<=3; q++)
{
    k0[q]=0;
    k1[q]=0;
    k2[q]=0;
    ka[q]=0;
}

/* Do Runge-Kutta 4 for the given kappa and alpha */

while (a>0.05 && a<1.5 && i*chartime>-40.0) /* bound the function */
{
    k0[0]=h*psi[1];
    k1[0]=h*psi[2];
    k2[0]=h*func(psi[0],psi[1],a);
    ka[0]=h*dadt(a,psi[0],psi[1]);

    k0[1]=h*(psi[1]+0.5*k1[0]);
    k1[1]=h*(psi[2]+0.5*k2[0]);
    k2[1]=h*func(psi[0]+0.5*k0[0],psi[1]+0.5*k1[0],a+0.5*ka[0]);
    ka[1]=h*dadt(a+0.5*ka[0],psi[0]+0.5*k0[0],psi[1]+0.5*k1[0]);

    k0[2]=h*(psi[1]+0.5*k1[1]);
    k1[2]=h*(psi[2]+0.5*k2[1]);
    k2[2]=h*func(psi[0]+0.5*k0[1],psi[1]+0.5*k1[1],a+0.5*ka[1]);
    ka[2]=h*dadt(a+0.5*ka[1],psi[0]+0.5*k0[1],psi[1]+0.5*k1[1]);

    k0[3]=h*(psi[1]+k1[3]);
    k2[3]=h*func(psi[0]+k0[2],psi[1]+k1[2],a+ka[2]);
    ka[3]=h*dadt(a+ka[2],psi[0]+k0[2],psi[1]+k1[2]);

    /* Calculate values for all the interesting and cool stuff */
    a=a+ka[0]/6+ka[1]/3+ka[2]/3+ka[3]/6;
    psi[0]=psi[0]+k0[0]/6+k0[1]/3+k0[2]/3+k0[3]/6;
    psi[1]=psi[1]+k1[0]/6+k1[1]/3+k1[2]/3+k1[3]/6;
    psi[2]=func(psi[0],psi[1],a);

    i=i+h; /* go until one of the boundaries is reached */
}

/* once the boundary is reached, print kappa and age */
fprintf(pout,"%f\t%f
",kappa,i*chartime);

    kappa=kappa+10; /* step kappa ahead */
}

printf("Complete\n");
return;
}

/* The end */
/k_and_a_value.c
#include<stdio.h>
#include<math.h>

/*This program uses the output from restrictor.c to output values of kappa and alpha that result in an age of the universe between 12.67 to 12.73 billion years old*/

int main(void)
{
    //Variables
    float array[4026][3]; //This array will be used to store the contents of the data file
    int count=0; //counter for loops
    int i,j; //more counters
    float a,b,c,d; //place holders for array elements
    FILE *rout;
    rout=fopen("restrict(2).dat","r");
    //file that is generated by restrictor.c
    //contains, kappa, alpha, and age of universe in tab delimited columns
    if ( (rout = fopen("restrict(2).dat","r")==NULL))
    {
        printf("Error: Can't open file\n");
    }
    for(i=0;i<100;i++) //initially sets array elements to zero
        for(j=0;j<3;j++)
            array[i][j]=0.0;
    while( count<4026) //copies file contents into array
    {
        fscanf(rout,"%f %f %f %f\n",&a,&b,&c,&d);
        array[count][0]=a;
        array[count][1]=b;
        array[count][2]=c;
        count++;
    }
    close(rout);
    for(count=0;count <= 4025;count++) //prints values if age is between bounds
    {
            printf("%f\t%f\t%f\n",array[count][0],array[count][1],array[count][2]);
    }
    return 0;
}