BRANECODE: A program for simulations of braneworld dynamics ☆

Johannes U. Martin [a,*], Gary N. Felder [b], Andrei V. Frolov [c],
Lev Kofman [d], Marco Peloso [e]

[a] *Physikalisches Institut, Nussallee 12, 53115 Bonn, Germany*
[b] *Department of Physics, Clark Science Center, Smith College Northampton, MA 01063, USA*
[c] *KIPAC/ITP, Stanford University, Stanford, CA 94305-4060, USA*
[d] *CITA, University of Toronto, 60 St George St., Toronto, ON M5S 3H8, Canada*
[e] *School of Physics and Astronomy, University of Minnesota, Minneapolis, MN 55455, USA*

Abstract

We describe an algorithm and a C++ implementation that we have written and made available for calculating the fully nonlinear evolution of 5D braneworld models with scalar fields. Bulk fields allow for the stabilization of the extra dimension. However, they complicate the dynamics of the system, so that analytic calculations (performed within an effective 4D theory) are usually only reliable for static bulk configurations or when the evolution of the extra dimension is negligible. In the general case, the nonlinear 5D dynamics can be studied numerically, and the algorithm and code we describe are the first ones of that type designed for this task. The program and its full documentation are available on the Web at http://www.cita.utoronto.ca/~jmartin/BRANECODE/.[1] In this paper we provide a brief overview of what the program does and how to use it.

Program summary

Title of program: BRANECODE
Catalogue identifier: ADVX
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/ADVX
Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland
Licensing provisions: none

---

1. Introduction

Many extensions of the Standard Model have in common the presence of extra dimensions. This has to be contrasted with the fact that our world looks four-dimensional, so one has to explain why the presence of the extra space has not yet been detected. The traditional answer has been that the extra space is compact and very small, so that the fields associated with its excitations are too heavy to be observable in accelerators or cosmology. More recently, it has been realized that ordinary matter and gauge interactions may be confined on lower dimensional submanifolds, known as branes. In this case, they could be four-dimensional objects, even if the geometry of the theory is higher dimensional. The situation is different for gravity, which propagates in the whole bulk space. Several questions naturally arise, such as why a compact space would remain small while the three non-compact dimensions are undergoing cosmological expansion, or why the expansion of the universe we see is described by $(3 + 1)$-dimensional general relativity so well. The presence of extra dimensions may cause deviations from the standard FRW cosmology that is supported by observations.

In most cases, these two questions turn out to be intimately related. Only if the extra space is static can the evolution of the non-compact coordinates behave as in the standard four-dimensional case. Hence, the dynamics of the hidden dimensions becomes a crucial ingredient in understanding the evolution of the ones we observe. In some particular cases, static bulk configurations can be achieved under the combined action of the bulk/brane gravity. In most realistic examples that could account for our observed four-dimensional cosmology the stability is due to the presence of additional fields that acquire nontrivial configurations in the bulk. While the stabilization has to be effective at relatively "late" times, the first stages of our universe (before primordial nucleosynthesis, for instance) are much less constrained. The evolution of the bulk may have been significant at this phase, and this offers many new possibilities for phenomenology. This is particularly true with the addition of the fields responsible for the "late" time stabilization, since they constitute new dynamical degrees of freedom for the system.

While the above considerations are valid for all models with extra dimensions, significant computations have been performed in the framework of brane models. These models can be thought of as simplified, phenomenological (bottom-up) versions of branes in string theory. The string dynamics is ignored and the primary focus is on the classical dynamics from the

point of view of General Relativity. Branes act as a source for the Einstein equations of the system, with their tension and possibly with the energy density of fields confined on them. Additional sources are a bulk cosmological constant, or the energy density of possible bulk fields. This set-up is sufficiently rich to describe very interesting situations. For example, inflation in braneworlds can acquire a nice geometrical interpretation, with the inflation associated with the distance between different branes, the Hubble parameter scale associated with the induced curvature on the branes, and with reheating through radion oscillations.

Despite this great simplification, the whole dynamics is still very complicated, in particular when a bulk scalar field is present. In this case, analytical computations are typically performed within an effective 4-dimensional theory, obtained after the extra dimension is integrated out, or perturbatively, using linearized analysis around simple backgrounds. While these studies are very useful when the extra space is static or quasi-static, they are not sufficient to describe the system when the evolution of the bulk is important. For example, systems that are stable at low energy (low curvature of the branes) can become unstable when the energy/curvature is increased. The stability/instability can be studied analytically. However, one cannot determine analytically where the system will evolve towards when the initial configuration turns out to be unstable.

For example, we numerically examined the dynamics of brane collisions and found that, as the branes approach each other, the spacetime of the bulk asymptotically approaches the Kasner-type solution.

Motivated by these limitations with the analytical treatment, we undertook a numerical study of these models. We developed a numerical algorithm, implemented in a C++ code, specifically designed for codimension one braneworlds, with a scalar field included in order to provide stabilization of the bulk at low energies. With the assumption of homogeneity and isotropy along the brane spatial coordinates **x** (corresponding to the standard assumption of homogeneity and isotropy of the non-compact coordinates), the problem is reduced to an effectively 2-dimensional one. The independent variables are the bulk time t and the bulk dimension y. The program integrates numerically the full set of Einstein equations in the bulk, together with the Israel junction conditions at two orb-

ifold branes. We discretize the two-dimensional spacetime (time and the bulk coordinate) and solve the bulk differential equations by finite-differencing them using the so-called *leapfrog* scheme. This algorithm is sufficient for our problem, and it provides a good compromise between accuracy and computational time (see Section 2 for more details). The two branes act as (one-dimensional) boundaries of this space, and the junction conditions provide the boundary conditions for the system at each time-step. The solution of a boundary value problem is required to provide generic initial conditions that fulfill the constraint equations and the boundary conditions at the beginning of the evolution. In the static configurations we have mentioned above, this boundary value problem is significantly simpler than in the general case. The setup of initial conditions is explained in more detail in Section 4.

The first results obtained with the code have been presented in [1]. We are now making the code public on the World Wide Web under the name BRANE-CODE. Its website is at http://www.cita.utoronto.ca/~kofman/BRANECODE/. The website for the program has documentation, including derivations of all the equations used in the program. Here we present a short summary of what the program does and what it can be used for. For more details see the website. Section 2 of this paper gives an overview of what the program is and how it works. Sections 3 and 4 describe the evolution equations and the setting of initial conditions, respectively. Section 5 describes some of the output generated by the program. The references section is limited to papers from our group related to the BRANECODE design and its first results [1–5]. See these papers for a more complete set of references.

2. Overview and user adjustable files

In this section we give an overview of the program and how to adjust it for a particular simulation. More details can be found in the documentation.

To work with the BRANECODE the user must specify a model, consisting of bulk and brane potentials for the scalar field, plus initial conditions for the field and the geometry. This information is encoded in a *model file*, which is a header file read in by the program. The file should be called `model_numeric.h`

or `model_analytic.h` depending on whether the initial conditions are specified numerically or analytically. The model files contain the potentials and their first and second derivatives that are needed for the evolution of the bulk equations (3) and boundary conditions (5). For example, the BRANECODE distribution includes both a numeric and an analytic model file (with different initial conditions). The file `model_analytic.h` contains examples for branes in AdS and AdS-Schwarzschild geometries, whereas the file `model_numeric.h` we designed for the class of models with bulk scalar fields determined by the bulk and brane potentials

$$
\begin{aligned}
V(\phi) &= \tfrac{1}{2}m^2\phi^2 + \Lambda, \\
U_i(\phi) &= \tfrac{1}{2}M_i(\phi_i - \sigma_i)^2 + \lambda_i.
\end{aligned}
\tag{1}
$$

A bulk cosmological constant and brane tensions are included as constant terms in these potentials. Most importantly, the program is designed to work for arbitrary potentials, different from (1). Other potentials $V(\phi)$, $U_i(\phi)$ can be implemented by modifying the corresponding lines in the model file. Aside from this file, the only other file that the user needs to modify is `parameters.h`, which contains all the parameters for a given run of the program. These include the number of grid points, the running time, and a number of other general variables specific to each run. There is also a parameter in this file that tells the program which type of model file to look for.

Given a specific model and set of parameters, the BRANECODE solves the system of equations of motions for the metric functions and the scalar field (3) along with the boundary conditions provided by the presence of the branes (5). The required functions are contained in the file `equations.cpp`.

The BRANECODE has built-in routines for outputting and plotting the metric fields, the scalar field, and derived quantities. These outputs are stored in ASCII files that can be read in and plotted by any standard plotting software. There are also options (set in `parameters.h`) to have the program call GNUPLOT to generate and display postscript plots of the data at runtime. (These options should only be chosen if the program is running on a computer with both GNU-PLOT and GHOSTVIEW.) An example of this graphical output is shown in Fig. 1.
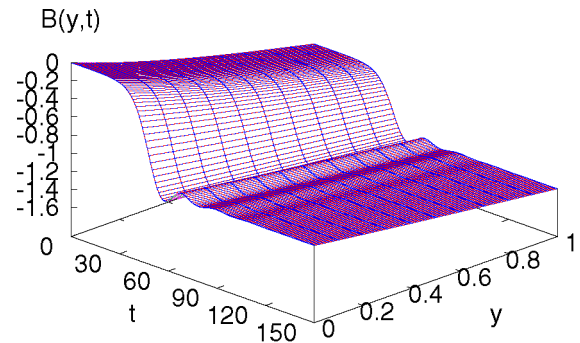

Fig. 1. An example of the graphical output generated by the BRANECODE. The plot shows the evolution of a metric component B (describing the interbrane separation) as a function of space and time. Physically, this evolution shows a transition from an unstable static warped geometry solution towards a stable static solution. During this non-linear reconfiguration the interbrane distance and the Hubble scale of the de Sitter geometry decreases.

Once all parameters have been set and you have modified or created a model file according to your wishes you simply compile and run the BRANECODE. The code is designed to be platform independent and should work with any C++ compiler. The makefile that comes with the distribution has entries and flags for the GNU `gcc` compiler and the INTEL `icc` compiler. You can select one of them or edit the makefile to invoke your favorite compiler.

3. Algorithm

The evolution equations solved by the BRANE-CODE are the set of Einstein/scalar field equations on an effectively 2-dimensional spacetime obtained after imposing homogeneity and isotropy on the non-compact spatial coordinates **x**. In [1] we showed that under these conditions it is always possible to choose coordinates that bring the metric to the form

$$
ds^2 = e^{2B(t,y)}(-dt^2 + dy^2) + e^{2A(t,y)}d\mathbf{x}^2,
\tag{2}
$$

with the two branes fixed at $y = 0, 1$, respectively. This choice is motivated by the fact that in this coordinate system the lattice size is time independent due to the fixed position of the branes and the equations simplify significantly. In this gauge, we have the following dynamical equations

$$\ddot{A} - A'' + 3\dot{A}^2 - 3A'^2 = \tfrac{2}{3}\mathrm{e}^{2B}V,$$

$$\ddot{B} - B'' - 3\dot{A}^2 + 3A'^2 + \tfrac{1}{2}\dot{\phi}^2 - \tfrac{1}{2}\phi'^2 = -\tfrac{1}{3}\mathrm{e}^{2B}V,$$

$$\ddot{\phi} - \phi'' + 3\dot{A}\dot{\phi} - 3A'\phi' = -\mathrm{e}^{2B}V_{,\phi}, \tag{3}$$

supplemented by the constraint equations

$$-A'\dot{A} + B'\dot{A} + A'\dot{B} - \dot{A}' = \tfrac{1}{3}\dot{\phi}\phi',$$

$$2A'^2 - A'B' + A'' - \dot{A}^2 - \dot{A}\dot{B}$$
$$\quad = -\tfrac{1}{6}\dot{\phi}^2 - \tfrac{1}{6}\phi'^2 - \tfrac{1}{3}\mathrm{e}^{2B}V. \tag{4}$$

Dots and primes denote derivatives with respect to the time t and the coordinate y along the bulk, respectively. In addition, the program imposes the following junction (Israel) conditions at the positions of the branes

$$A' = -\tfrac{1}{6}U\mathrm{e}^B, \qquad B' = -\tfrac{1}{6}U\mathrm{e}^B,$$

$$\phi' = \tfrac{1}{2}\mathrm{e}^B U_{,\phi}. \tag{5}$$

These junction conditions are equivalent to extending the space beyond the two branes, and imposing $\mathbb{Z}_2$ symmetry across the branes.

In the coordinate system (2), the characteristic propagation speed of the dynamical equations is always 1. This is advantageous from a numerical viewpoint as the size of the time step can be optimized uniformly by setting it equal to spatial grid separation. We describe below our implementation of the *leapfrog* discretization scheme, which is stable, second order accurate, and non-dissipative.

The program discretizes the 2-dimensional $\{t, y\}$ space and computes the value of the three functions B, A, ϕ at each grid point. For any fixed time, the grid is made of $N + 1$ points equally spaced along the bulk, with 0 and N corresponding to the locations of the two branes. The value of N is set in para-meters.h. The same grid spacing is taken in the y- and t-directions. The initial conditions are in the form $(A(y), B(y), \phi(y), \dot{A}(y), \dot{B}(y), \dot{\phi}(y))$, and they can be chosen arbitrarily subject to the fact that they satisfy (4) and (5) at the initial time. Of particular interest are configurations with an initially static bulk, since the program can be used to verify their stability and to study the dynamics when they are unstable as described in Section 4.

The discretized initial data $(A_i, \dot{A}_i, B_i, \dot{B}_i, \phi_i, \dot{\phi}_i)$, $i = 0, \ldots, N$, has to be converted to a form suitable for the *leapfrog* algorithm. Instead of having the fields and their velocities at the time $t = t_0$, the algorithm needs the spatial profiles of the fields (A_i, B_i, ϕ_i) at two subsequent moments in time $t = t_0$ and $t = t_0 + \epsilon$. In some special cases the profiles at the two initial times can be calculated analytically, but in general the program takes a Runge–Kutta step using the initial derivatives to calculate the field values at $t_0 + \epsilon$.

The field values at all subsequent time steps are calculated as follows. (Note that we will use f here to denote a generic field, i.e. for equations that apply to all three fields A, B, and ϕ.) In the gauge chosen, the bulk equations only contain derivatives in the form $\ddot{f} - f''$, and $\dot{f}\dot{g} - f'g'$. Recall that our grid spacing $\epsilon = 1/N$ is equal to our time step. Thus, to second order accuracy, we can write the derivatives at a given point in spacetime in terms of the values of its neighbors as

$$\ddot{f} - f'' = \frac{1}{\epsilon^2}(f_{\mathrm{up}} + f_{\mathrm{dn}} - f_{\mathrm{lt}} - f_{\mathrm{rt}}) + \mathrm{O}(\epsilon^2),$$

$$\dot{f}\dot{g} - f'g' = \frac{1}{4\epsilon^2}\big[(f_{\mathrm{up}} - f_{\mathrm{dn}})(g_{\mathrm{up}} - g_{\mathrm{dn}}) \tag{6}$$
$$\quad - (f_{\mathrm{rt}} - f_{\mathrm{lt}})(g_{\mathrm{rt}} - g_{\mathrm{lt}})\big] + \mathrm{O}(\epsilon^2),$$

where the indices label relative grid positions as defined in Fig. 2. In this way, the three differential equations (3) become three algebraic equations for the three unknown quantities $(B_{\mathrm{up}}, A_{\mathrm{up}}, \phi_{\mathrm{up}})$. Notice that this can be done independently site by site in the bulk.

Once the bulk field values at the new time have been determined, we can apply the junction conditions (5) to advance the boundary field values. We discuss only the computation at the first brane ($i = 0$). The treatment for the second brane is analogous. Only first derivatives in y enter in (5). To preserve second order accuracy, we use an "asymmetric" discretization of the derivatives

$$f_0' = \frac{1}{2\epsilon}(-3f_0 + 4f_1 - f_2) + \mathrm{O}(\epsilon^2), \tag{7}$$

where subscripts indicate the grid position of each field value (see Fig. 2). The junction conditions (5) thus become a set of three algebraic equations in terms of three unknowns. We can eliminate A_0 and B_0 in favor of ϕ_0, and write an equation in terms of the only unknown quantity ϕ_0,

$$4\phi_1 - \phi_2 - 3\phi_0 - \epsilon\,\mathrm{e}^{B_0(\phi_0)}U_0'(\phi_0) = 0. \tag{8}$$
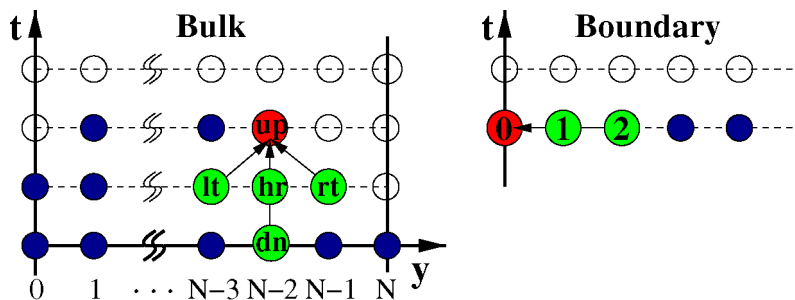
Fig. 2. Numerical evolution scheme. See text for details.

For specific brane potentials, these equations can be solved analytically. However, as the code is designed for arbitrary bulk/brane potentials, Eq. (8) is solved numerically using the iterative Newton's method. Once ϕ_0 is determined, the remaining unknowns B_0 and A_0 are trivially computed through the remaining junction conditions.

4. Initial conditions

In general, the specification of initial conditions, i.e. the determination of the initial spatial profiles for the fields and their velocities $A(y)$, $B(y)$, $\phi(y)$, $\dot{A}(y)$, $\dot{B}(y)$, and $\dot{\phi}(y)$ is a nontrivial task. These function cannot be chosen independently, but rather are subject to the constraint equations (4) and boundary conditions (5). However, the physical instability of static de Sitter configurations [1,3] provides the possibility to generate interesting braneworld dynamics with static solutions as initial conditions that we can recommend. All the static solutions for a given model can be exhaustively classified by the phasespace analysis of the dynamics of the gravity/scalar system performed in [2].

One simple way to generate static initial conditions is to consider a configuration of the form $B = B(y)$, $\phi = \phi(y)$, $A = B + Ht$ (de Sitter branes in a static bulk). The first of Eqs. (4) is then trivially satisfied. For a sufficiently simple model, the remaining constraint equation can be solved analytically. Otherwise we provide a MATHEMATICA notebook and MAPLE worksheet to solve them numerically for a given bulk potential and generate appropriately formatted initial data. For simplicity, the code determines two of the three parameters of the brane potentials (1) in such a way that the boundary conditions (5) are fulfilled initially. The third parameter, either the tension on the brane λ_i or the minimum of the brane potential σ_i, is set by the user in the file model_numeric.h. If the user instead wants to set up initial conditions for a given choice of brane potentials, he can make use of a shooting method operating in the phasespace of the static solutions. Namely, he can start with initial conditions chosen such that they fulfill the boundary conditions on one of the branes, compute the corresponding bulk configuration, and keep varying the initial conditions (e.g., with a numerical scan) until the solution also satisfies the junction conditions at the second boundary. For any given set of bulk and brane potentials, there can be none, one, or more than one solution to the boundary value problem. The latter case opens up the possibility for interesting dynamics of transitions as investigated in [1].

5. Output

The main outputs of the program are the values of the three functions B, A, and ϕ at different bulk sites and time steps. Two parameters inside parameters.h control how many points (both in the y- and t-directions) are to be saved. From these quantities, one can construct some outputs of immediate physical relevance. One is the physical interbrane distance. The branes are at a fixed coordinate distance in our gauge and their physical separation is encoded in the metric coefficient B

$$D(t) \equiv \int_0^1 \mathrm{d}y \sqrt{g_{55}} = \int_0^1 \mathrm{d}y\, e^{B(t,y)}. \qquad (9)$$

Another interesting quantity is the Hubble parameter as computed by observers on each of the two branes

$$H_i \equiv \frac{1}{a} \frac{\mathrm{d}a}{\mathrm{d}\tau}\bigg|_i = \mathrm{e}^{-B_i} \dot{A}_i, \tag{10}$$

where τ is the physical time measured by the observers, defined by $D\tau \equiv \mathrm{e}^B \mathrm{d}t$. Quite interestingly, the gauge choice used in the algorithm (see the previous section) does not exhaust the gauge freedom of the problem. Residual gauge transformations have been described in [1], and one can show that they do not change the values of D and H_i, which therefore have physical meaning. Besides these physical quantities the BRANECODE also computes the Ricci scalar and the square of the Weyl tensor in the bulk. Other quantities of interest can easily be obtained from the "raw" values of B, A, ϕ.

6. Conclusions

The main motivation for our work was to extend the knowledge of braneworld dynamics beyond the few situations where it was known analytically. Apart from these situations (characterized by a static or slowly evolving bulk), approximate methods based on effective 4-dimensional computations are unreliable. In this short note we have presented an algorithm, together with its C++ implementation, designed for numerical computations in this framework. First results obtained from this code were presented in [1,4]. We could show that some bulk configurations which are stable at low energy (low value of the expansion rate H of the two branes) become unstable as H increases, in agreement with the analytical calculations of [3]. This can be interpreted as a part of a more general phenomenon of gravitational instability of compactification to four-dimensional de Sitter geometry [5]. The numerical integration allowed us to follow the evolution of the system starting from the unstable configuration. For certain bulk/brane potentials the system may evolve towards another static, but stable configuration, characterized by a lower value of H. The transition is typically a process of quick bulk reconfiguration. In many cases, however, the second configuration does not exist or cannot be reached. The two branes then either move apart to infinite distance, or they collide. Brane collisions is a very interesting subject by itself.

The study of the numerical evolution led us to the conclusion that the asymptotic geometry is given by the universal Kasner-type that is typical of homogeneous but anisotropic strong gravity regimes. We did not investigated the case of branes departing from each other when one of them in general approaches a naked singularity in the bulk configuration. We also did not studied the possibility of the formation of an apparent horizon between the branes.

Our algorithm is focused on the simplest possible set-up which allows for brane stabilization based on a generalized Goldberger–Wise mechanism. Only one bulk scalar field has been considered, although with arbitrary potentials in the bulk and at the two branes. The inclusion of more scalar fields would be straightforward. In particular, one could consider other fields, which are confined to the branes, and which are coupled to the bulk fields, e.g., through the brane potentials. (The interplay between bulk/brane fields may lead to novel interesting features not considered in [1].) Another easy generalization would be the inclusion of perfect fluids on the branes (for example, describing standard matter and radiation). Less trivial but more interesting extensions could be the inclusion of other types of fields (form fields in the bulk, for instance), or evolution with more dimensions included. For example, relaxing the hypothesis of homogeneity

```
*************************************
* Potential parameters on the branes *
*************************************
            Brane0         Brane1
VEVs    :  9.990000e-02    5.371387e-01
Zetas   :  1.000000e+03    1.000000e+03
Lambdas:  6.186450e+00   -7.707347e+00


***************************
* The Branecode is running *
***************************

Directory data/ existed, will be moved
  to data_sav/
New data will be stored in directory:
  data/

Start at coordinate time: 0.000000
End at coordinate time  : 150.000000
Current coordinate time :+1.010000e+02
```

Fig. 3. Output during the simulation.

and isotropy along the ordinary dimensions would allow the study of inhomogeneous perturbations of the system.

Appendix A. Test run output

Once the program is started the message captured in Fig. 3 is printed on the screen. It provides information about the parameters that are computed by the program, the location where the generated data is stored and the current time of calculation together with the starting time and the time of termination.

```
Data stored on disk in directory data/

*************************************
* Calculation finished successfully *
*************************************
Elapsed real time: 0 d:0 h:0 min:22 sec.

Execute: gnuplot radion.gp && gv -antialias -geometry 643x430+0+20 radion.eps &
Execute: gnuplot B.gp && gv -antialias -geometry 643x430+645+470 B.eps &
Execute: gnuplot A.gp && gv -antialias -geometry 643x430+0+470 A.eps &
Execute: gnuplot phi.gp && gv -antialias -geometry 643x430+645+20 phi.eps &
```
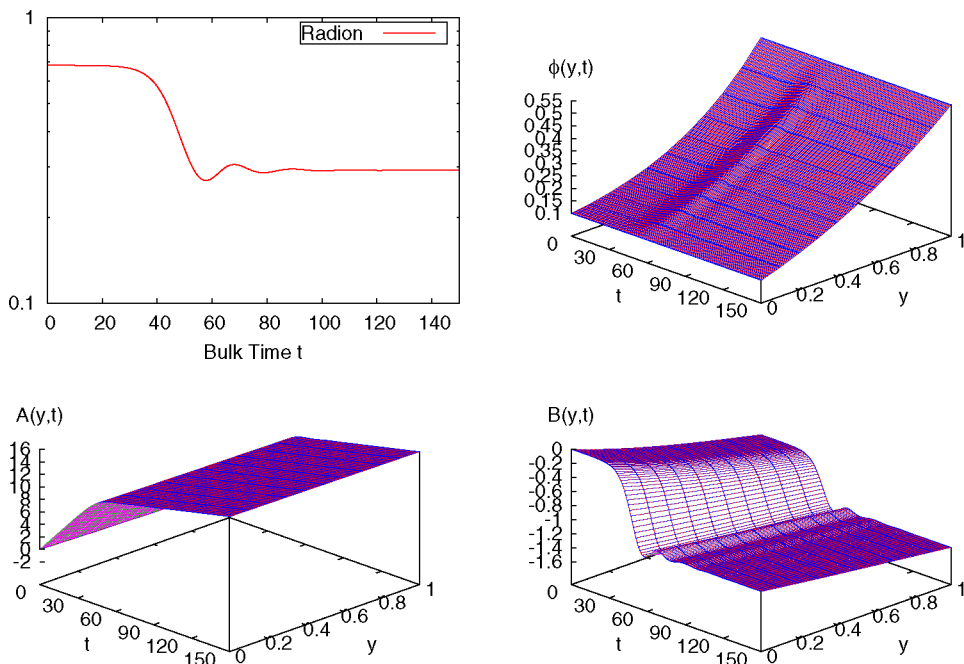
Once the calculation is finished and the data is stored the message in Fig. 4 is printed. It includes the real time that it took to perform the simulation as well as additional system commands for the graphical representation of the data if selected in the header file `parameters.h`, cf. Section 2.

In the example provided with the distribution the main variables of the simulation are plotted using standard LINUX packages GNUPLOT and GV, cf. Fig. 5.

All model and simulation parameters of a particular run are stored in the file `info.dat`. An example for its content is given in Fig. 6.

Fig. 4. Output after the completion of a simulation.

Fig. 5. Output example of the BRANECODE.

```
Grid Resolution
500
OUTPUT_SKIP
50
OUTPUT_INTERVAL
10
Starting time TMIN
0.000000e+00
Final time TMAX
1.500000e+02
Bulk Mass of the Scalar Field
5.000000e-01
Cosmological Constant in the Bulk
-6.000000e+00
Effective Mass of the Scalar Field on the Brane at y=0
1.000000e+03
Expectation Value of the Scalar Field on the Brane at y=0
9.990000e-02
Effective Mass of the Scalar Field on the Brane at y=1
1.000000e+03
Expectation Value of the Scalar Field on the Brane at y=1
5.371387e-01
Tension of the Brane at y=0
6.186450e+00
Tension of the Brane at y=0
-7.707347e+00
Bulk Hubble Parameter
2.500000e-01
Error status
0
```

Fig. 6. Example of the parameters stored in the file info.dat.

```
  #        time  proper time       Hubble          C2

   9.980000e-01 9.999987e-01 2.499949e-01 1.763024e-12
   1.998000e+00 1.999995e+00 2.499937e-01 1.114797e-10
   2.998000e+00 2.999982e+00 2.499929e-01 4.309199e-11
```

Fig. 7. Example of the data format of the generated file brane0.dat.

```
 #        time              A

 0.000000e+00 0.000000e+00 -9.844431e-02 -1.886905e-01 -2.721424e-01
 2.000000e-03 4.999965e-04 -9.794431e-02 -1.881905e-01 -2.716424e-01
 1.000000e+00 2.499959e-01 1.515521e-01 6.130630e-02 -2.214513e-02
 2.000000e+00 4.999891e-01 4.015455e-01 3.112998e-01 2.278485e-01
```

Fig. 8. Example of the data format of the generated file A.dat.

The data generated during a simulation consists of discretized functions of one or two independent variables t and y. The induced quantities on either branes, e.g., the proper time τ or the expansion parameter H, only depend on the coordinate time t. They are collected in the files brane0.dat and brane1.dat,

```
#       time           y            phi           A              B

0.000000e+00 0.000000e+00 1.000000e-01  0.000000e+00  0.000000e+00
0.000000e+00 2.000000e-01 1.276959e-01 -1.886905e-01 -1.886905e-01
0.000000e+00 4.000000e-01 1.753732e-01 -3.499188e-01 -3.499188e-01
0.000000e+00 6.000000e-01 2.513273e-01 -4.920424e-01 -4.920424e-01
0.000000e+00 8.000000e-01 3.662943e-01 -6.214045e-01 -6.214045e-01
0.000000e+00 1.000000e+00 5.350111e-01 -7.442296e-01 -7.442296e-01

2.000000e-03 0.000000e+00 1.000000e-01  4.999965e-04 -3.518520e-09
2.000000e-03 2.000000e-01 1.276959e-01 -1.881905e-01 -1.886905e-01
2.000000e-03 4.000000e-01 1.753732e-01 -3.494188e-01 -3.499188e-01
2.000000e-03 6.000000e-01 2.513273e-01 -4.915424e-01 -4.920424e-01
2.000000e-03 8.000000e-01 3.662943e-01 -6.209045e-01 -6.214045e-01
2.000000e-03 1.000000e+00 5.350111e-01 -7.437296e-01 -7.442296e-01
```

Fig. 9. Example of the data format of the generated file `gnu.dat`.

cf. Fig. 7. For the functions that depend on the two independent variables t and y there are two possible formats for the output. The first possible choice of formatting prints the function in an array. Each line starts with the value of time for which the values of the function are plotted along the one-dimensional lattice $y \in [0, 1]$ in equidistant steps. Fig. 8 shows the first lines of the data file A.dat that contains the discretized version of the metric function $A(t, y)$. Alternatively, the data can be stored in a format that is adjusted to GNUPLOT. Here the first two columns collect the values of the independent functions t and y. The values of the dependent functions are contained in the subsequent columns. Fig. 9 shows the first lines of the data file gnu.dat that stores the numerical data of the discretized fields $\phi(t, y)$, $A(t, y)$, $B(t, y)$.

The number of grid points and the number of time steps that are stored in the data files can be adjusted with the corresponding control parameters in the header file parameter.h.

References

[1] J. Martin, G.N. Felder, A.V. Frolov, M. Peloso, L. Kofman, braneworld dynamics with the BraneCode, Phys. Rev. D, submitted for publication, hep-th/0309001.

[2] G.N. Felder, A.V. Frolov, L. Kofman, Warped geometry of brane worlds, Class. Quant. Grav. 19 (2002) 2983, hep-th/0112165.

[3] A.V. Frolov, L. Kofman, Can inflating braneworlds be stabilized, Phys. Rev. D 69 (2004) 044021, hep-th/0309002.

[4] L. Kofman, J. Martin, M. Peloso, Exact identification of the radion and its coupling to the observable sector, hep-ph/0401189.

[5] C. Contaldi, L. Kofman, M. Peloso, Gravitational instability of de Sitter compactifications, hep-th/0403270.