Co-Op Work Term Report:

Numerical Study of the Nonlinear Klein-Gordon Equation

John N. Homenuke

Phys 298

Department of Physics & Astronomy
University of British Columbia

Submitted August 30, 2004

John Homenuke

Numerical Study of the Nonlinear Klein-Gordon Equation

Author
John N. Homenuke
Student Number
26100032

Department of Physics & Astronomy
Faculty of Science
University of British Columbia

Coordinates
University of British Columbia, Hennings Building, Room 414
May 3 - August 27, 2004

Preface

This work is chiefly a numerical study of a particular form of the nonlinear Klein-Gordon (nlKG) equation. It is based on the previous work, *Resonant Structures within the Nonlinear Klein-Gordon Equations*, the thesis of Dr. Ethan P. Honda, a former student of Dr. Matthew W. Choptuik. Choptuik is a professor in the Department of Physics & Astronomy at the University of British Columbia in Vancouver, Canada, and the supervisor of the author.

The nlKG equation was solved numerically using finite-difference methods in one spatial dimension with a symmetric double well potential as its nonlinear term. As is the case with Honda's thesis, a coordinate system that contains an absorbing region is implemented to greatly increase solution accuracy over long time periods. In addition, the solutions were generated using a language called Rapid Numerical Prototyping Language, developed by Dr. Robert Marsa (UT Austin) and Dr. Choptuik. Prior to the actual solving of the nlKG equation are explanations of the numerical techniques used throughout this work and a solution to the wave equation, which provides many of the details skipped over in the nlKG equation chapter.

Summary

This work is a reproduction of some of the results of Ethan Honda's Ph. D. thesis, *Resonant Dynamics within the Nonlinear Klein-Gordon Equations*, namely, the solutions to the nonlinear Klein-Gordon (nlKG) equation with a symmetric double well potential. The solutions are localized and oscillatory and so termed *oscillons*. The lifetimes of these phenomena can be divided into three stages: an initial period of high radiation followed by a pseudo-stable quasi-periodic phase and a final stage of dispersal. Resonance states exist where the lifetimes are infinite and it is the critical behaviour surrounding these resonance states that is the focus of this work.

The coordinate system used is termed *monotonically increasingly boosted* coordinates. In the radial direction, it interpolates between regular, static coordinates and an outgoing null coordinate, freezing ingoing and outgoing radiation in the interpolating region. This "bunched up" radiation is then dissipated using a method developed by Kreiss and Oliger [6], leaving the interior region virtually free of contamination.

Prior to the major results are discussions of the numerical methods employed in solving the nlKG equation. Crank-Nicolson finite-difference schemes were programmed using Rapid Numerical Prototyping Language (RNPL), a language developed by Robert Marsa and Matthew Choptuik that is specically designed for solving time-dependent PDEs. As well, the solution to the wave equation is presented using the same methods as above (but in a regular coordinate system) to illustrate the finer details of the how the nlKG equation is solved.

Contents

Chapter 1

Introduction

The oscillating solutions to the Nonlinear Klein-Gordon (nlKG) equation (and sine-Gordon equation) were first discovered by Bogolyubskii and Mankhan'kov [1] in 1976, although they originally called them pulsons for their radiative properties. They were analyzed in more detail by Copeland *et al* in 1995 [2] who renamed them *oscillons* for their oscillatory bahaviour. They were not able to study the solutions in great detail due to a lack of computational resources. In 2002, Ethan Honda took advantage of better computational techniques and was able to do a more or less complete study of the oscillon solutions in his thesis, *Resonant Dynamics within the Nonlinear Klein-Gordon Equation* [5]. The goal of this work is to reproduce some of Honda's results, namely the resonant solutions within a symmetric double well potential.

Oscillons are thought of as (spherically symmetric or axisymmtric) bubbles separating two vacuum states ϕ_c and ϕ_0 with the bubble wall interpolating between them. These bubbles exhibit three-stage lifetimes: (1) Much of the bubble's mass is shed as energy that radiates outward; (2) This is followed by the pseudo-stable oscillion phase in which the core field value oscillates almost periodically and virtually no energy is radiated; (3) The oscillon either collapses quietly or with a final burst of energy. The most significant factor in the lifetime of the oscillon is its initial radius r_0. At certain initial radii, there exist resonance states in which the lifetime goes to inifinity. Three are shown and analyzed in this work, but there are many more as demonstrated by Honda [5].

Additionally, Honda used a novel coordinate system termed *montonically increasingly boosted* (MIB) coordinates as a means to reduce contamination of data. The need for this arises from the lack of an outgoing boundary condition for the massive scalar field. It can only be approximated by the condition for the massless case (wave equation). When radiation reaches the outer boundary in a regular coordinate system with an outgoing boundary condition, it is partially reflected back into the computational domain. This effect is several times greater if the equation of motion is massive compared to the massless case.

In MIB coordinates, the equations are solved in flat spacetime, but the points near the outer boundary travel outwards at nearly the speed of light while the points near the origin remain nearly motionless with an interpolating region between. Outgoing and ingoing radiation are trapped in this interpolating region and subsequently dissipated leaving the inner region virtually free of contamination.

Following this introduction is a chapter outlining the numerical methods employed to solve all the equations in this work. All the equations are solved using Crank-Nicolson finite-difference schemes and the programs for generating the solutions are written in Rapid Numerical Prototyping Language (RNPL).

Preceeding the work on the nlKG equation is a detailed solution to the wave equation.

Of particular interest is the regularity condition at the origin and the outgoing boundary condition at $r = r_{\mathrm{max}}$.

The major results of this work, unfortunately, bear only qualitative resemblence to Honda's due to a systematic computational error that could not be corrected. The resonances were shown to exist, but their positions in the parameter space r_0 and their lifetimes do not align with Honda's.

Chapter 2

Numerical Analysis

This chapter provides the foundation for understanding how differential equations are solved on a discrete domain and describes the tricks and tools for doing so.

2.1 Discretization

2.1.1 The Residual

A differential equation (or a system thereof) can be written generally as

$$\mathrm{L}(u) - f = 0 \tag{2.1}$$

where L is some differential operator, u is the yet undetermined solution, and f is possibly some function of u, the coordinates, and time. The corresponding difference equation can be written as

$$\hat{\mathrm{L}}(\hat{u}) - \hat{f} = 0 \tag{2.2}$$

where the $\hat{}$ notation indicates the corresponding discretized variable. Equation (2.2) can never be an exact replacement for (2.1) and $\hat{u}$ is not always an exact solution to (2.2). In the case where it cannot be made exact (nonlinear DE), a small term $\hat{r}$ is added to the equation to account for this and $\hat{u}$ is replaced by the approximation $\tilde{u}$,

$$\hat{\mathrm{L}}(\tilde{u}) - \hat{f} = \hat{r}. \tag{2.3}$$

Equation (2.3) is iterively re-solved as a nonlinear algebraic equation would be until $\hat{r} \to 0$ and necessarily $\tilde{u} \to \hat{u}$. If this occurs in the limit of infinite iteration, the finite-difference scheme used is said to be *convergent*.

2.1.2 Error

The difference in the discrete and continuum solutions is the solution error,

$$\hat{e} = \hat{u} - u, \tag{2.4}$$

which is more easily understood as a Richardson Expansion, a power series in h,

$$\hat{u} = u + e_1 h^1 + e_2 h^2 + e_3 h^3 + e_4 h^4 + \cdots . \tag{2.5}$$

If the scheme is centered, only even error terms will appear. If the scheme is not centered, then odd terms will appear as well.

The truncation error $\hat{\tau}$ is a measure of how well the continuum solution satisfies the discrete equations.

$$\hat{\tau} = \hat{\mathrm{L}}(u) - \hat{f} \tag{2.6}$$

The scheme is said to be a consistent representation of the continuum equation if the truncation error goes to zero as the grid spacing goes to zero. That is to say, the disrete and continuum solutions will be equal. In practice, the grid spacing can only be made so close to zero, resulting in

$$\lim_{h\to 0} \hat{\tau} = \mathrm{O}(h^p). \tag{2.7}$$

where p is some positive integer. This test reflects the accuracy of the scheme in reproducing the continuum solution.

2.1.3 Convergence

Most often, the purpose behind finding numerical solutions is that a continuum solution is very difficult to determine or cannot be found. So how can one be sure to have found the right discrete solution with nothing to compare it too? If the scheme is second-order accurate ($p = 2$) as are all the schemes in this work, the following condition should be met.

$$C_f = \frac{\hat{u}_{4h} - \hat{u}_{2h}}{\hat{u}_{2h} - \hat{u}_h} \to 4 \quad \text{as} \quad h \to 0 \tag{2.8}$$

C_f is called the convergence factor and $\hat{u}_{nh}$ are discrete solutions computed at different grid spacings. Visually, the successive solutions will asymptotically converge to a solution representing $h \to 0$.

2.2 Finite-difference Schemes

Finite difference schemes are the manner in which the differential equations are discretized. They are either explicit or implicit. Explicit schemes render difference equations that can be solved for the future time step completely in terms of previous time steps. For example, the forward Euler and Leap-frog schemes respectively are

$$u_j^{n+1} = u_j^n + kN_h(u_j^n), \tag{2.9}$$
$$u_j^{n+1} = u_j^{n-1} + 2kN_h(u_j^n), \tag{2.10}$$

where N_h is some function of the discrete variables, j and n index the spatial and temporal points in the domain, respectively, and k is the time step. Implicit schemes must be solved as a set of (potentially nonlinear) algebraic equations at each time step. For example, the backward Euler and Crank-Nicolson schemes respectively are

$$u_j^{n+1} = u_j^n + kN_h(u_j^{n-1}), \tag{2.11}$$
$$u_j^{n+1} - u_j^n = \frac{1}{2}k\left[N_h(u_j^n) + N_h(u_j^{n+1})\right] \tag{2.12}$$

This work was exclusively done using Crank-Nicolson schemes for their stability over long time periods and $\mathrm{O}(h^2)$ accuracy. This is due to the time averaging that occurs in the right-hand side of equation (2.12). These schemes were taken from Drazin [3].

Operator		Expansion		Definition	
$\Delta_t^{\mathrm{f}}(u_j^n)$	$=$	$\partial_t u\big\vert_j^{n+1/2} + \mathrm{O}(h^2)$	$=$	$\frac{1}{\Delta t}\left(u_j^{n+1} - u_j^n\right)$	(2.13)
$\mu_t^{\mathrm{f}}(u_j^n)$	$=$	$u\big\vert_j^{n+1/2} + \mathrm{O}(h^2)$	$=$	$\frac{1}{2}\left(u_j^n + u_j^{n+1}\right)$	(2.14)
$\Delta_r^{\mathrm{c}}(u_j^n)$	$=$	$\partial_r u\big\vert_j^{n} + \mathrm{O}(h^2)$	$=$	$\frac{1}{2\,\Delta r}\left(u_{j+1}^n - u_{j-1}^n\right)$	(2.15)
$\Delta_{r^3}^{\mathrm{c}}(u_j^n)$	$=$	$\partial_{r^3} u\big\vert_j^{n} + \mathrm{O}(h^2)$	$=$	$\frac{u_{j+1}^n - u_{j-1}^n}{(r_{j+1/2})^3 - (r_{j-1/2})^3}$	(2.16)
$\Delta_r^{\mathrm{f}}(u_j^n)$	$=$	$\partial_r u\big\vert_{j+1}^{n} + \mathrm{O}(h^2)$	$=$	$\frac{1}{2\,\Delta r}\left(-3u_j^n + 4u_{j+1}^n - u_{j+2}^n\right)$	(2.17)
$\Delta_r^{\mathrm{b}}(u_j^n)$	$=$	$\partial_r u\big\vert_{j-1}^{n} + \mathrm{O}(h^2)$	$=$	$\frac{1}{2\,\Delta r}\left(3u_j^n - 4u_{j-1}^n + u_{j-2}^n\right)$	(2.18)
$\mu_t^{\mathrm{diss}}(u_j^n)$	$=$	$\Delta r^3 \partial_r^4 u\big\vert_j^{n} + \mathrm{O}(h^2)$	$=$	$-\frac{\epsilon}{16\,\Delta t}\left(u_{j-2}^n - 4u_{j-1}^n + 6u_j^n + 4u_{j+1}^n + u_{j+2}^n\right)$	(2.19)

Table 2.1: Finite-difference Operators

2.2.1 Finite-difference Operators

There is a straight-forward natural way to generate finite-difference equations from the differential equations. The differential operators are replaced with their corresponding difference operators. Time-averaging operators are also inserted to create Crank-Nicolson schemes. The difference operators are generated by combining different Taylor series expansions of derivitives and truncating terms that are second order in h. See Appendix A for some of the derivations. Table 2.1 lists the operators used in this work with their definitions.

2.3 Rapid Numerical Prototyping Language (RNPL)

The programming behind this work was done almost entirely in Rapid Numerical Prototyping Langauge. RNPL is a high-level programming language developed by Dr. Robert Marsa (UT Austin) and Dr. Matthew Choptuik for the purpose of solving systems of differential equations using finite-difference techniques. Its usefulness lies in its natural, operator-based syntax, relatively compact source code, and built-in check-pointing mechanism.

The source code consists entirely of declaration statements including parameters, the grid(s), grid functions, operators, and the FD equations. The operators are defined in the code essentially as they are in table 2.1 and the FD equations are generated as described in section 2.2.1 above. Well-commented source code for the spherically symmetric case in the symmetric double well potential (SDWP) is in Appendix B.

The RNPL compiler does not do all the work. It simply writes either the C or Fortran source code, depending on the user's preference, that one would otherwise write in order to solve the same equations in C or Fortran. These are then compiled and linked to form the executable. The executable takes one argument, a parameter file that supercedes the default parameters in the source code so they can be changed easily.

Chapter 3

Wave Equation

This chapter details the preliminary work done prior to reproducing Honda's data. It is essentially a "practice run" for the author, but also a simple example for the reader and provides many details for understanding the work in the following chapter that aren't included therein.

3.1 Wave Equation in Spherical Symmetry

The wave equation in spherical coordinates with unit speed and no angular dependence is

$$\frac{\partial^2 \phi}{\partial t^2} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial \phi}{\partial r}\right). \tag{3.1}$$

This, however, is not the preferred form of the wave equation if a numerical solution is sought after.

3.1.1 Regularity at the Origin

The boundary condition at $r = 0$ is to be Neumann, so ϕ should be an even function as $r \to 0$.

$$\lim_{r\to 0} \phi(t, r) = \phi_0(t) + \phi_2(t)r^2 + \phi_4(t)r^4 + \mathrm{O}(r^6) \tag{3.2}$$

Casting the right hand side of (3.1) in this form,

$$\lim_{r\to 0} \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial \phi}{\partial r}\right) = 6\phi_2(t) + 20\phi_4(t)r^2 + \mathrm{O}(r^4). \tag{3.3}$$

According to equation (3.2), $\partial_r \phi = \mathrm{O}(r)$ to leading order. Thus, $r^2 \partial_r \phi = \mathrm{O}(r^3)$ to leading order. Then using the following relation,

$$\frac{\partial}{\partial r} = \frac{\partial(r^3)}{\partial r}\frac{\partial}{\partial(r^3)} = 3r^2 \frac{\partial}{\partial(r^3)}, \tag{3.4}$$

equation (3.1) becomes

$$\frac{\partial^2 \phi}{\partial t^2} = 3\frac{\partial}{\partial(r^3)}\left(r^2 \frac{\partial \phi}{\partial r}\right). \tag{3.5}$$

Although this derivitive applied to $r^2\partial_r\phi$ also returns an expression whose leading term is O(1) in the limit just as much as the form of equation (3.1) does, when (3.2) at fixed time is substituted into the discrete form of (3.5), (see below for definitions of discretized variables)

$$3\frac{r^2_{j+\frac{1}{2}}\left(u^n_{j+1}-u^n_j\right)-r^2_{j-\frac{1}{2}}\left(u^n_j-u^n_{j-1}\right)}{\Delta r\left(r^3_{j+\frac{1}{2}}-r^3_{j-\frac{1}{2}}\right)}, \tag{3.6}$$

the following results.

$$6\phi_2-\frac{402}{13}\phi_4\Delta r^2 \tag{3.7}$$

This is precisely equation (3.3) to leading order. When (3.2) is substituted into the discrete form of (3.1),

$$\frac{r^2_{j+1/2}(\phi^n_{j+1}-\phi^n_j)-r^2_{j-1/2}(\phi^n_j-\phi^n_{j-1})}{r^2_j}, \tag{3.8}$$

the result is

$$\frac{27}{4}\phi_2\Delta r^2+\frac{67}{2}\phi_4\Delta r^4, \tag{3.9}$$

which is not equation (3.3) to leading order.

3.1.2 Equations of Motion

The solution scheme implemented here is written in RNPL. It is best utilized when the equations to be solved are split into a first order system because the language only allows one initial condition to be specified for each equation. Given the definition of Π in equation (3.10), the initial conditions can be specified as they normally would when solving the wave equation, that is, with an initial profile $\phi(0,r)$ and initial time derivitive $\Pi(0,r)$.

$$\frac{\partial\phi}{\partial t}=\Pi \tag{3.10}$$

$$\frac{\partial\Pi}{\partial t}=3\frac{\partial}{\partial(r^3)}\left(r^2\frac{\partial\phi}{\partial r}\right) \tag{3.11}$$

3.1.3 Sommerfeld Outgoing Boundary Condition

This equation is to be solved on the domain

$$0\le x\le r_{\max},\qquad t\ge 0.$$

with a Neumann conditions at $r=0$ (left side) and Sommerfeld conditions at $r=r_{\max}$ (right side).

$$\frac{\partial\phi}{\partial r}(t,0)=0 \tag{3.12}$$

$$\frac{\partial\Pi}{\partial r}(t,0)=0 \tag{3.13}$$

Sommerfeld conditions at the right boundary allow waves to be transmitted through it from the left as if there was no boundary, but no waves are allowed to enter the domain from the right. Determining this boundary condition in the cartesian case for simplicity, we begin with the general solution to the wave equation,

$$\phi(t,x)=f(t+x)+g(t-x), \tag{3.14}$$

where $f(t+x)$ represents any wave function travelling to the left and $g(t-x)$ is any wave function travelling to the right. If no left-moving waves are to enter the domain through the boundary, then necessarily $f=0$. Letting $u=t-x$, the general solution (3.14) becomes

$$\phi(t,x) = g(u) \tag{3.15}$$

Differentiating $\phi(t,x)$ with respect to t and x,

$$\frac{\partial\phi}{\partial t} = \frac{\mathrm{d}g}{\mathrm{d}u}\frac{\partial u}{\partial t}, \tag{3.16}$$

$$\frac{\partial\phi}{\partial x} = \frac{\mathrm{d}g}{\mathrm{d}u}\frac{\partial u}{\partial x}. \tag{3.17}$$

Adding (3.16) and (3.17),

$$\frac{\partial\phi}{\partial t} + \frac{\partial\phi}{\partial x} = \frac{\mathrm{d}g}{\mathrm{d}u}\left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\right) = 0, \tag{3.18}$$

a condition on $\phi(t,x)$ results that applies to the right endpoint only.

It need only be shown that $r\phi(t,r)$ is a solution to the 1-d cartesian wave equation and the boundary condition (3.18) may be applied in the spherical case. Beginning with

$$\frac{\partial^2(r\phi)}{\partial t^2} = \frac{\partial^2(r\phi)}{\partial r^2}, \tag{3.19}$$

it will be shown that equation (3.1) follows.

$$\begin{aligned} \frac{\partial^2}{\partial t^2}(r\phi) &= \frac{\partial^2}{\partial r^2}(r\phi) \\ r\frac{\partial^2\phi}{\partial t^2} &= \frac{\partial}{\partial r}\left(\phi + r\frac{\partial\phi}{\partial r}\right) \\ \frac{\partial^2\phi}{\partial t^2} &= \frac{r}{r^2}\frac{\partial}{\partial r}\left(\phi + r\frac{\partial\phi}{\partial r}\right) \end{aligned} \tag{3.20}$$

From here, it is trivial to show that

$$r\frac{\partial}{\partial r}\left(\phi + r\frac{\partial\phi}{\partial r}\right) = \frac{\partial}{\partial r}\left(r^2\frac{\partial\phi}{\partial r}\right). \tag{3.21}$$

Given that (3.21) is true, it can be substituted into (3.20) resulting in (3.1), which shows that it follows from (3.19) and vice versa because the algebaic steps can be performed in reverse. Therefore, the boundary conditions on the right can safely be stated as

$$\frac{\partial(r\phi)}{\partial t}(t,r_{\max}) = -\frac{\partial(r\phi)}{\partial r}(t,r_{\max}), \tag{3.22}$$

$$\frac{\partial(r\Pi)}{\partial t}(t,r_{\max}) = -\frac{\partial(r\Pi)}{\partial r}(t,r_{\max}). \tag{3.23}$$

3.1.4 Initial Conditions

The initial condition is chosen to be a Gaussian pulse, shown in figure 3.1,

$$\phi(0,r) = a\,\exp\left[-\left(\frac{r-R}{\Delta}\right)^2\right], \tag{3.24}$$

$$\Pi(0,r) = \frac{\sigma}{r}\frac{\mathrm{d}}{\mathrm{d}r}\phi(0,r). \tag{3.25}$$

where $\sigma = 1, 0, -1$ causing the pulse to initially move left, time-symmetrically, or right, respectively.

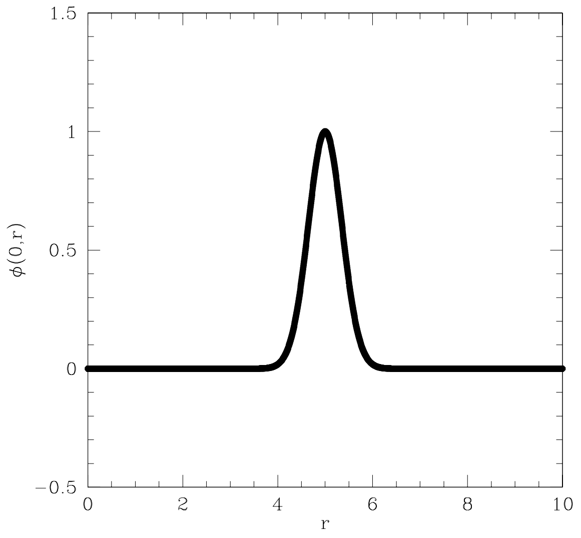

Figure 3.1: Gaussian initial profile of wave equation with parameters $a = 1.0$, $R = 5.0$, $\Delta = 0.5$, and $\sigma = 1.0$.

3.1.5 Discretization

The domain needs to be converted into a discrete set of points (or grid), upon which the solution ϕ_j^n is defined. The grid here is defined as the set of points (t^n, r_j) such that

$$t^n = n\,\Delta t,$$
$$r_j = (j-1)\Delta r,$$

for $j = 1, 2, \cdots, J$ and $n = 0, 1, 2, \cdots$, and

$$\Delta r = \frac{r_{\max}}{J-1},$$
$$\Delta t = \lambda\,\Delta r,$$

where λ is an adjustable parameter called the Courant factor. The discrete solutions are

$$\phi_j^n = \phi(t^n, x_j), \qquad \Pi_j^n = \Pi(t^n, x_j).$$

Including the boundary conditions, the equations of motion in the discrete domain are

$$\Delta_t^{\mathrm{f}}(\phi_j^n) = \mu_t^{\mathrm{f}}(\Pi_j^n), \qquad j = 1, 2, \cdots, J \tag{3.26}$$
$$\Delta_r^{\mathrm{f}}(\Pi_1^{n+1}) = 0, \tag{3.27}$$
$$\Delta_t^{\mathrm{f}}(\Pi_j^n) = \mu_t^{\mathrm{f}}\delta_{r^3}^{\mathrm{c}}(\phi_j^n), \qquad j = 2, 3, \cdots, J-1 \tag{3.28}$$
$$\Delta_t^{\mathrm{f}}(r_J\Pi_J^n) + \mu_t^{\mathrm{f}}\Delta_r^{\mathrm{b}}(r_J\Pi_J^n) = 0. \tag{3.29}$$

The initial conditions in their discrete form are

$$\phi_j^0 = 0, \qquad j = 1, J \tag{3.30}$$
$$\phi_j^0 = a\,\exp\left[-\left(\frac{r_j - R}{\Delta}\right)^2\right], \qquad j = 2, 3, \cdots, J-1 \tag{3.31}$$
$$\Pi_j^0 = \frac{\sigma}{r_j}\Delta_r^{\mathrm{c}}(r_j\phi_j^0), \qquad j = 1, 2, \cdots, J. \tag{3.32}$$

Equations (3.26) through (3.32) are the discrete analogues of equations (3.10) through (3.13) and (3.22) through (3.25). The differential operators are replaced with the corresponding finite difference operators from table 2.1 and continuous variables with the corresponding discrete variables.

3.2 Solutions and Analysis

The full solutions and convergence tests of the spherically symmetric wave and sine-Gordon equations as well as similar solutions to the 1-d cartesian wave equation can be viewed in `mpg` format at [5].

Figure 3.2 shows the time evolution of the Gaussian pulse (3.24) according to the spherically symmetric wave equation. The profile is initially left-moving, $\sigma = 1$. It reflects off the left boundary and escapes through the right boundary as was intended.

The solutions on the less dense grids show a small reflection off the right side that becomes more visible as it approaches the origin. The more dense grids reduce the magnitude of this phenomenon monotonically. The time taken for that small pulse to travel from one end and back is called the *light crossing time* as the speed of these waves is very close to unity. This error invariably contaminates solution data if evolved for longer than a crossing time.

Additionally, a small piece of the initial profile splits off and moves to the right. This, again, is due to computational imprecision (and the nature of solutions to the wave equation) and can be minimized by using a denser grid.

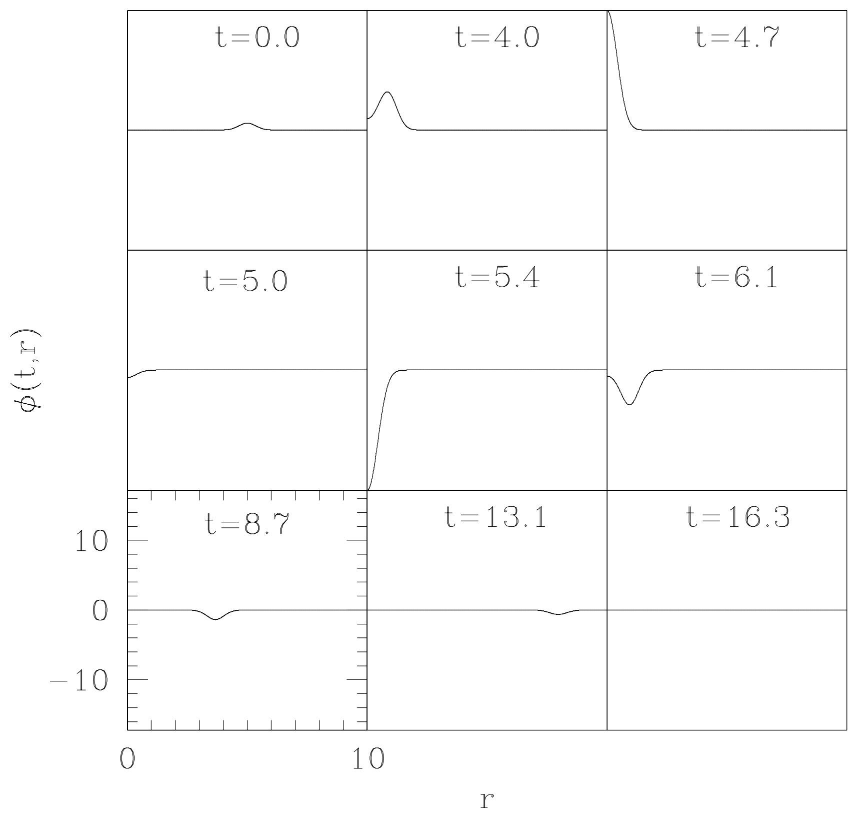

Figure 3.2: **Solution to wave equation**. The pulse begins in the center, reflects off the left boundary, and escapes through the right boundary.

Chapter 4

ϕ^4 Klein-Gordon Equation

4.1 MIB Coordinates

The coordinate system here is said to be *monotonically increasingly boosted* (MIB). The points that constitute the domain move outward according a monotonic velocity function $f(r)$ relative to the origin. The function sets up three regions in the domain. The innermost region is a typical radial coordinate, the outermost is an outgoing null coordinate and the middle region interpolates smoothly between the two. It has the definition

$$f(r) = \frac{1}{2}\tanh\left(\frac{r - r_w}{\delta}\right) + \frac{1}{2}\tanh\left(\frac{r_w}{\delta}\right) \tag{4.1}$$

and therefore the following properties.

$$f(r) \begin{cases} = 0, & r = 0 \\ \approx 0, & r \ll r_w \\ \approx 1, & r \gg r_w \end{cases}$$

The middle region is where the usefulness of the coordinate system lies. Radiation entering it from either direction is compressed in the radial direction and its speed is brought to near zero. Dissipation added to the equations everywhere dampens the radiation proportionally to its wavenumber. This dampening is therefore only effective in the middle region, leaving the interior solution virtually free of contamination and still based in a regular spherical coordinate system.

The geometry of the spacetime here is flat, but it will be helpful to write the metric and the equations in a (3+1) form useful in general relativity.

$$\mathrm{d}\tilde{s}^2 = -\mathrm{d}\tilde{t}^2 + \mathrm{d}\tilde{r}^2 + r^2\mathrm{d}\tilde{\Omega}^2, \tag{4.2}$$

where $\mathrm{d}\tilde{\Omega}^2 = \mathrm{d}\tilde{\theta}^2 + \sin^2(\tilde{\theta})\mathrm{d}\tilde{\phi}^2$. The coordinate transformations are

$$\tilde{t} = t, \qquad \tilde{r} = r + f(r)t, \qquad \tilde{\Omega} = \Omega. \tag{4.3}$$

With these in place, the metric becomes

$$\begin{aligned} \mathrm{d}s^2 = &(-1 + f(r)^2)\mathrm{d}t^2 + 2f(r)(1 + f'(r)t)\mathrm{d}t\mathrm{d}r \\ &+ (1 + f'(r)t)^2\mathrm{d}r^2 + (r + f(r)t)^2\mathrm{d}\Omega^2. \end{aligned} \tag{4.4}$$

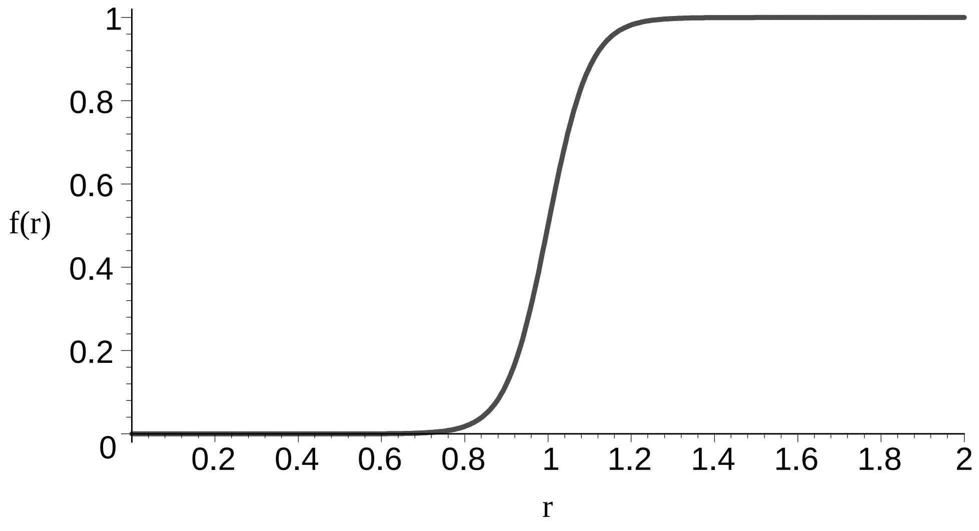

Figure 4.1: This function, $f(r) = \frac{1}{2}\tanh\left(\frac{r-r_w}{\delta}\right)+\frac{1}{2}\tanh\left(\frac{r_w}{\delta}\right)$ in units where $r_w = 1$, smoothly interpolates between the outgoing null coordinate and regular interior region. The characteristic velocities are zero in both directions in the interpolation region.

Using the definitions,

$$\begin{aligned} a(t,r) &= 1 + f'(r)t & b(t,r) &= 1 + \frac{f(r)t}{r} \\ \alpha(t,r) &= 1 & \beta(t,r) &= \frac{f(r)}{1+f'(r)t} \end{aligned} \tag{4.5}$$

the metric can be written in the (3+1) form,

$$\mathrm{d}s^2 = (-\alpha^2 + a^2\beta^2)\mathrm{d}t^2 + 2a^2\beta\mathrm{d}t\mathrm{d}r + a^2\mathrm{d}r^2 + r^2b^2\mathrm{d}\Omega^2. \tag{4.6}$$

In this form, the metric, called it $\mathbf{g}$, has determinant g, given by $\sqrt{-g} = \alpha a r^2 b^2 \sin(\theta)$. Its inverse is

$$\mathrm{inv}\,\mathbf{g} = \begin{bmatrix} -1/\alpha^2 & \beta/\alpha^2 & 0 & 0 \\ \beta/\alpha^2 & 1/a^2 - \beta/\alpha^2 & 0 & 0 \\ 0 & 0 & 1/r^2b^2 & 0 \\ 0 & 0 & 0 & 1/r^2b^2\sin^2(\theta) \end{bmatrix}. \tag{4.7}$$

4.2 Theory

The nonlinear Klein-Gordon equation with unit speed is most generally written as

$$\nabla^2\phi - \frac{\partial^2\phi}{\partial t^2} = \frac{\partial V}{\partial \phi} \tag{4.8}$$

where V is a some potential that depends on ϕ. In this case we are examining a symmetric double well potential (see fig. 4.2),

$$V_S(\phi) = \frac{1}{4}\left(\phi^2 - 1\right)^2. \tag{4.9}$$

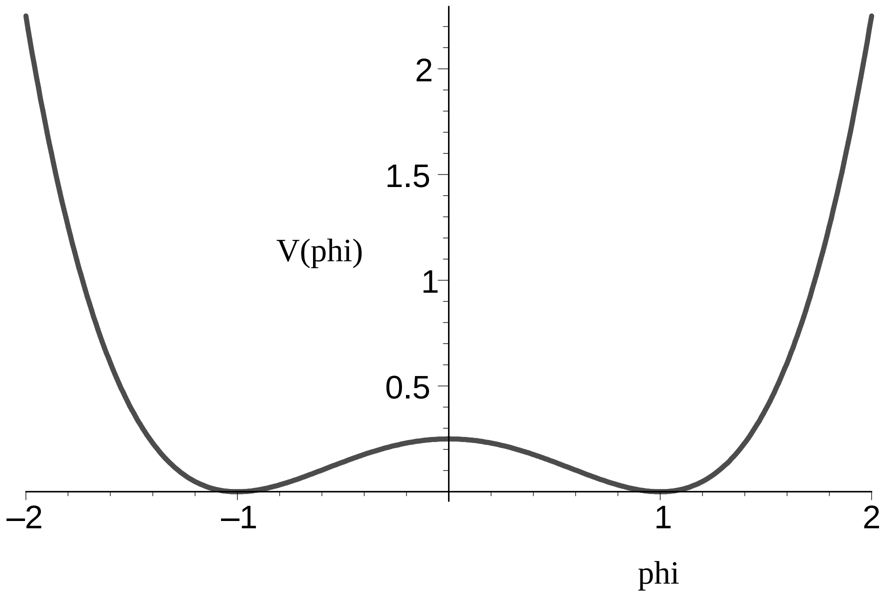

Figure 4.2: This is the symmetric double well potential, $V(\phi) = \frac{1}{4}(\phi^2-1)^2$. The two vacuum states are the minima of the function, 1 and -1.

4.2.1 Equations of Motion

The spherically symmetric action for a massive scalar field is

$$S(\phi) = \int \sqrt{-g}\left(-\frac{1}{2}g^{\mu\nu}\,\nabla_\mu\phi\,\nabla_\nu\phi - V(\phi)\right)\mathrm{d}^4x. \tag{4.10}$$

Since ϕ is a scalar field,

$$\nabla_\mu\phi = \frac{\partial\phi}{\partial x^\mu},$$

and the Lagrangian is

$$\mathcal{L} = \sqrt{-g}\left(-\frac{1}{2}g^{\mu\nu}\frac{\partial\phi}{\partial x^\mu}\frac{\partial\phi}{\partial x^\nu} - V(\phi)\right) \tag{4.11}$$

where $g_{\mu\nu}$ are the components of the flatspace metric $\mathbf{g}$ in spherically symmetric MIB coordinates. The equation of motion is found by substituting the Lagrangian into a particular form of the Euler-Lagrange equation,

$$\nabla_\mu\left(\frac{\partial\mathcal{L}}{\partial(\nabla_\mu\phi)}\right) - \frac{\partial\mathcal{L}}{\partial\phi} = 0, \tag{4.12}$$

resulting in

$$\nabla_\mu\left(-g^{\mu\nu}\nabla_\nu\phi\right) + \frac{\partial V}{\partial\phi} = 0. \tag{4.13}$$

This reduces to

$$g^{\mu\nu}\nabla_\mu\nabla_\nu\phi - \frac{\partial V}{\partial\phi} = 0,$$
$$\nabla^\nu\nabla_\nu\phi - \frac{\partial V}{\partial\phi} = 0.$$

Then from any text,

$$\frac{1}{\sqrt{-g}}\partial_\mu \left(\sqrt{-g}\, g^{\mu\nu}\partial_\nu \phi\right) - \frac{\partial V}{\partial \phi} = 0. \tag{4.14}$$

This equation of motion can be written as a set of two equations that are first order in time using the definition

$$\Pi = \frac{a}{\alpha}\left(\partial_t \phi - \beta \partial_r \phi\right). \tag{4.15}$$

The equations of motion become

$$\dot{\Pi} = \frac{1}{r^2 b^2}\left(r^2 b^2 \left(\frac{\alpha}{a}\Pi + \beta\phi'\right)\right)' - 2\frac{\dot{b}}{b}\Pi - \alpha a \phi(\phi^2 - 1) \tag{4.16}$$

$$\dot{\phi} = \frac{\alpha}{a}\Pi + \beta\phi' \tag{4.17}$$

in which the dot and prime notations are time and radial derivitives, respectively. As a means of confirming that these equations produce the right solutions, a different implementation of the same equation of motion is needed for comparison. Here, a set of three equations is devoloped with the definitions

$$\Pi = \frac{a}{\alpha}\left(\partial_t \phi - \beta \partial_r \phi\right), \tag{4.18}$$

$$\Phi = \partial_r \phi, \tag{4.19}$$

giving us

$$\dot{\Pi} = \frac{1}{r^2 b^2}\left(r^2 b^2 \left(\frac{\alpha}{a}\Pi + \beta\Phi\right)\right)' - 2\frac{\dot{b}}{b}\Pi - \alpha a \phi(\phi^2 - 1) \tag{4.20}$$

$$\dot{\Phi} = \left(\frac{\alpha}{a}\Pi + \beta\Phi\right)' \tag{4.21}$$

$$\dot{\phi} = \frac{\alpha}{a}\Pi + \beta\Phi \tag{4.22}$$

There exists no outoing boundary condition for the massive scalar field, so the OBC for the massless scalar field (wave equation) is used as an approximation.

4.2.2 Finite-difference Equations

For each equation of motion, there are five FD equations representing different regions of the discrete domain. These regions are $j = 1$, $j = 2$, $j = 3, \ldots, N-2$, $j = N-1$, and $j = N$. The outermost points are for the boundary conditions, and the remaining points are for the equations of motion. The five regions facilitate the dissipation operator, which spans five spatial points, so can only be applied in the $j = 3, \ldots, N-2$ region.

$$\Delta_r^{\mathrm{f}}(\Pi_1^{n+1}) = 0 \tag{4.23}$$

$$\Delta_t^{\mathrm{f}}(\Pi_2^n) = \mu_t^{\mathrm{f}} \left(\frac{3}{(1+f(r)t/r)^2} \Delta_{r^3}^{\mathrm{c}} \left(\frac{(r+f(r)t)^2(\Phi + f(r)\Pi)}{1+t\Delta_r^{\mathrm{f}} f(r)} \right) - \frac{2f(r)\Pi}{r+f(r)t} - \phi^3 + \phi \right) \tag{4.24}$$

$$\Delta_t^{\mathrm{f}}(\Pi_j^n) = \mu_t^{\mathrm{f}} \left(\frac{3}{(1+f(r)t/r)^2} \Delta_{r^3}^{\mathrm{c}} \left(\frac{(r+f(r)t)^2(\Phi + f(r)\Pi)}{1+t\Delta_r^{\mathrm{c}} f(r)} \right) - \frac{2f(r)\Pi}{r+f(r)t} - \phi^3 + \phi \right) + \mu_t^{\mathrm{diss}} \Pi_j^n, \; j = 3, \ldots, N-2 \tag{4.25}$$

$$\Delta_t^{\mathrm{f}}(\Pi_{N-1}^n) = \mu_t^{\mathrm{f}} \left(\frac{3}{(1+f(r)t/r)^2} \Delta_{r^3}^{\mathrm{c}} \left(\frac{(r+f(r)t)^2(\Phi + f(r)\Pi)}{1+t\Delta_r^{\mathrm{b}} f(r)} \right) - \frac{2f(r)\Pi}{r+f(r)t} - \phi^3 + \phi \right) \tag{4.26}$$

$$\Delta_t^{\mathrm{f}}(r\Pi_N^n) = -\mu_t^{\mathrm{f}} \left(\Delta_r^{\mathrm{b}}(r\Pi_N^n) \right) \tag{4.27}$$

$$\Phi_1^{n+1} = 0 \tag{4.28}$$

$$\Delta_t^{\mathrm{f}}(\Phi_j^n) = \mu_t^{\mathrm{f}} \, \Delta_r^{\mathrm{f}} \left(\frac{\Pi + f(r)\Phi}{1+t\Delta_r^{\mathrm{c}} f(r)} \right), \qquad j = 2, \, N-1 \tag{4.29}$$

$$\Delta_t^{\mathrm{f}}(\Phi_j^n) = \mu_t^{\mathrm{f}} \, \Delta_r^{\mathrm{f}} \left(\frac{\Pi + f(r)\Phi}{1+t\Delta_r^{\mathrm{c}} f(r)} \right) + \mu_t^{\mathrm{diss}} \Phi_j^n, \qquad j = 3, \ldots, N-1 \tag{4.30}$$

$$\Delta_t^{\mathrm{f}}(r\Phi_N^n) = -\mu_t^{\mathrm{f}} \left(\Delta_r^{\mathrm{b}}(r\Phi_N^n) \right) \tag{4.31}$$

$$\Delta_r^{\mathrm{f}}(\phi_1^{n+1}) = 0 \tag{4.32}$$

$$\Delta_t^{\mathrm{f}}(\phi_j^n) = \mu_t^{\mathrm{f}} \left(\frac{\Pi + f(r)\Phi}{1+t\Delta_r^{\mathrm{c}} f(r)} \right), \qquad j = 2, \, N-1 \tag{4.33}$$

$$\Delta_t^{\mathrm{f}}(\phi_j^n) = \mu_t^{\mathrm{f}} \left(\frac{\Pi + f(r)\Phi}{1+t\Delta_r^{\mathrm{c}} f(r)} \right) + \mu_t^{\mathrm{diss}} \phi_j^n, \qquad j = 3, \ldots, N-2 \tag{4.34}$$

$$\Delta_t^{\mathrm{f}}(r\phi_N^n) = -\mu_t^{\mathrm{f}} \left(\Delta_r^{\mathrm{b}}(r\phi_N^n) \right) \tag{4.35}$$

4.3 Solutions and Analysis

4.3.1 Performance of the MIB Code

The coordinates appear to do their job in halting and dissipating the outgoing radiation. The outgoing characteristic $\lambda_+ = 0$ in the absorbing region and beyond it at all times. The dissipation operator acts only on non-smooth regions. The absorbing region slows the radiation, causing it to "bunch up" and become more susecptible to the dissipation.

Unfortunately, the implementation of the MIB coordinate system here does not perform entirely as expected. Any ingoing radiation should be absorbed in addition to all the outgoing radiation. Immediately upon executing the evolutions, small oscillations take place at the outer boundary (see fig. 4.3). The oscillations propagate inward and evidently are not halted by the absorbing region. This failure of the absorbing region is possibly due its time-dependent nature. The characteristic velocity in the negative direction, λ_-, does not reach zero in the absorbing region until long after the start time of the evolution, allowing the early ingoing radiation to penetrate into the interior of the domain where the oscillon exists.

This problem was more or less continuous throughout the development phase of the MIB code and could not be reconciled. Solving the wave equation in MIB coordinates showed an instantaneous relationship between the solution at the inner boundary and at the outer boundary. Clearly this is a not a physical problem that can be blamed on the model, but instead it is an error in programming that seemed to always attach itself to the implementation of the MIB coordinates.

The effect of this is contaminated data after only one crossing time. Figure 4.4 demonstrates this behaviour at the origin.

4.3.2 Resonant Structure of Oscillons

Fortunately, the qualitative behaviour of the solutions is still intact, despite the inaccuracies. Here, the oscillon phase of the solutions is explored. The initial conditions are designed to omit the initial collapse of the bubble and start in the oscillon phase using

$$\phi(0,r) = \phi_0 + (\phi_c - \phi_0)\exp\left(-r^2/{r_0}^2\right) \tag{4.36}$$

to essentially give the bubble no excess volume. We call r_0 the initial radius of the bubble calculate the lifetimes of the oscillons over this parameter space. The vacuum states inside and outside the bubble respectively are $\phi_c = 1$ and $\phi_0 = -1$.

The oscillon lifetimes asymptote at certain values of r_0 (resonance states). The first three are plotted in figure 4.7. Additionally, there is a scaling law for each resonance. That is, the lifetimes depend linearly on $\log|r_0 - r_0^*|$ where r_0^* is the value of r_0 at the resonance (or as close to it as can be estimated). This is shown in figure 4.8.

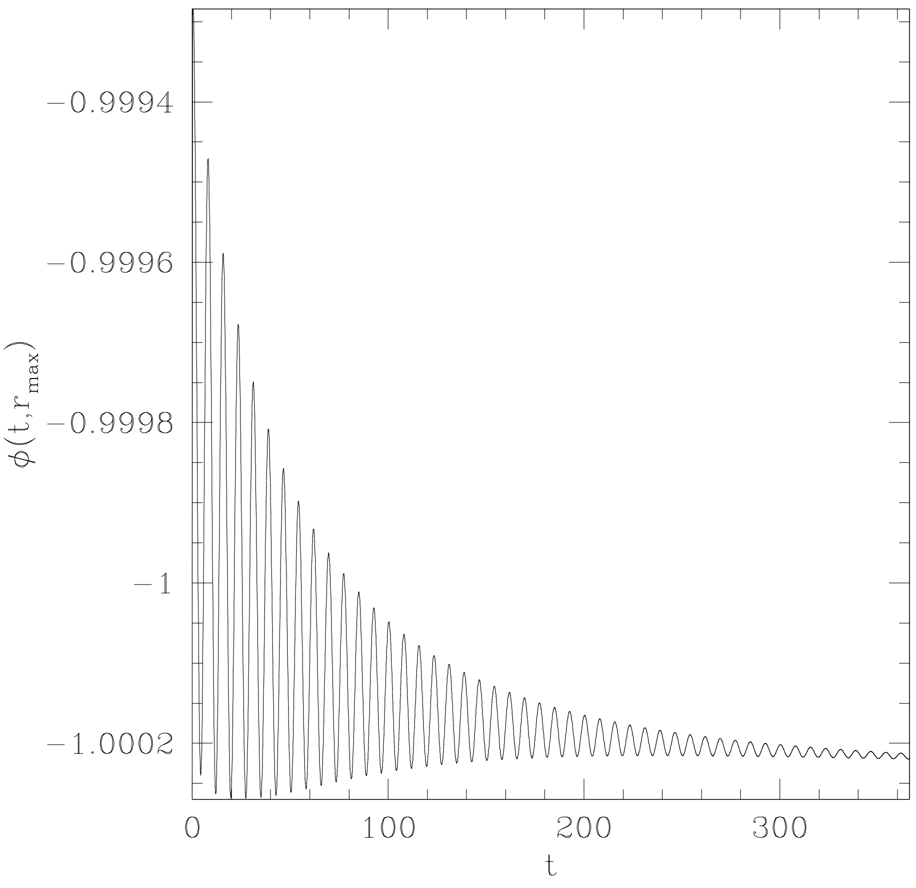

Figure 4.3: **Outer boundary.** These are the (unintended) small oscillations that occur at the outer boundary. The amplitude is $O(10^{-4})$ whereas the interior solutions are $O(1)$.

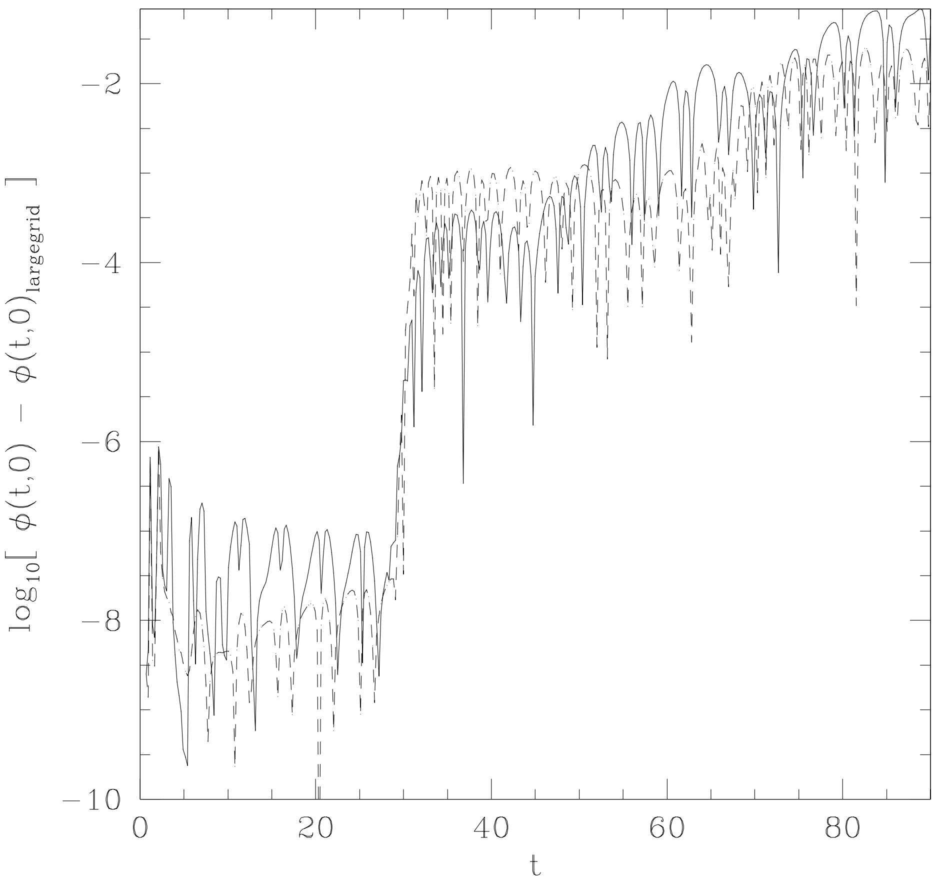

Figure 4.4: **Solution error.** The above plot compares the error in a solution using MIB coordinates (solid line) and a solution in regular (t, r) coordinates with Sommerfeld outgoing boundary conditions (dashed line). This error is difference in the either of the two solutions to a solution on a grid large enough to prevent any reflections off the outer boundary. The MIB solution exhibits behaviour no better than the OBC solution due to the computational problems at the outer boundary. The crossing time here is 30.

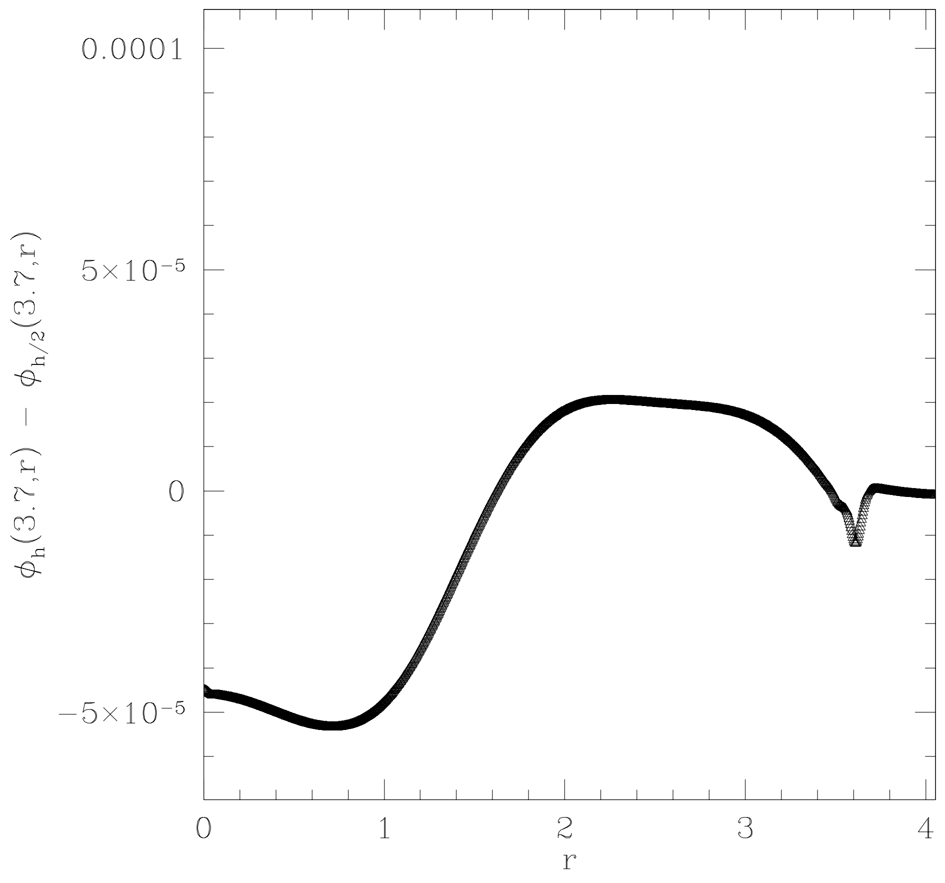

Figure 4.5: **Convergence test.** Although not easily distinguished, there are two functions, $\phi_h - \phi_{h/2}$ (solid line) and $4(\phi_{h/2} - \phi_{h/4})$ (triangles). Their approximate equality demonstrates that the code is convergent to $\mathrm{O}(h^2)$ accuracy. This "snapshot" of the solution is taken at $t = 3.7$.

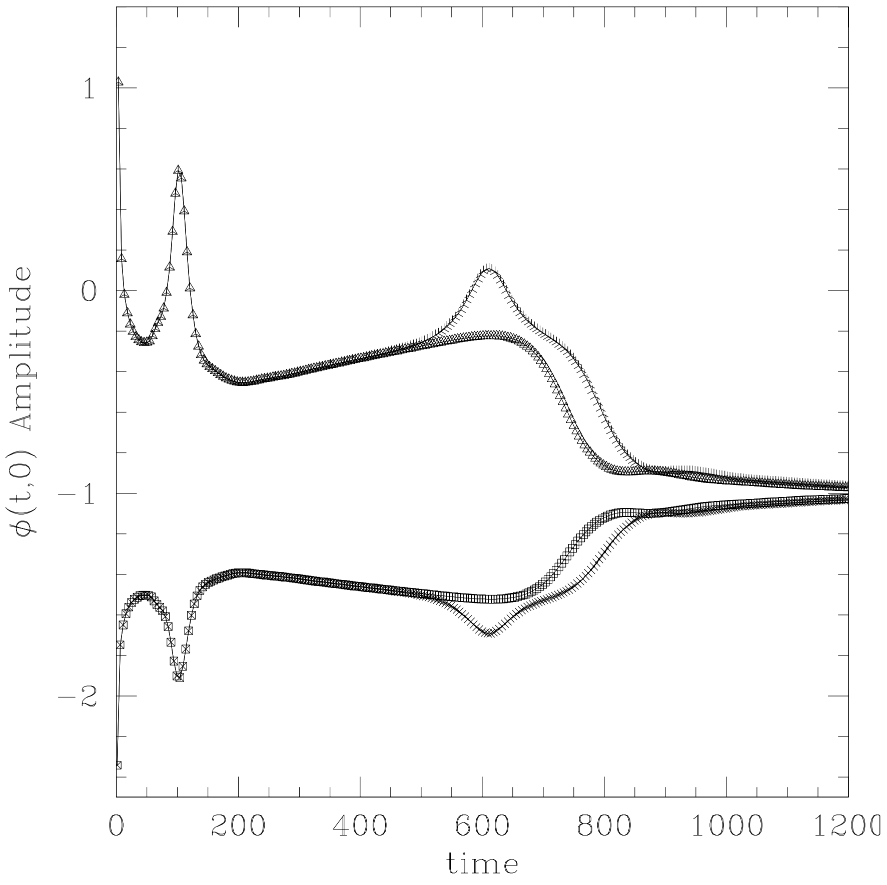

Figure 4.6: Above shows the difference in supercritical and subcritical behaviour of the oscillon. Barely above resonance, the oscillon disperses after one final modulation for a total of two modulations. Barely below, no final modulation is seen. This particular example is the second resonance in fig. 4.7.

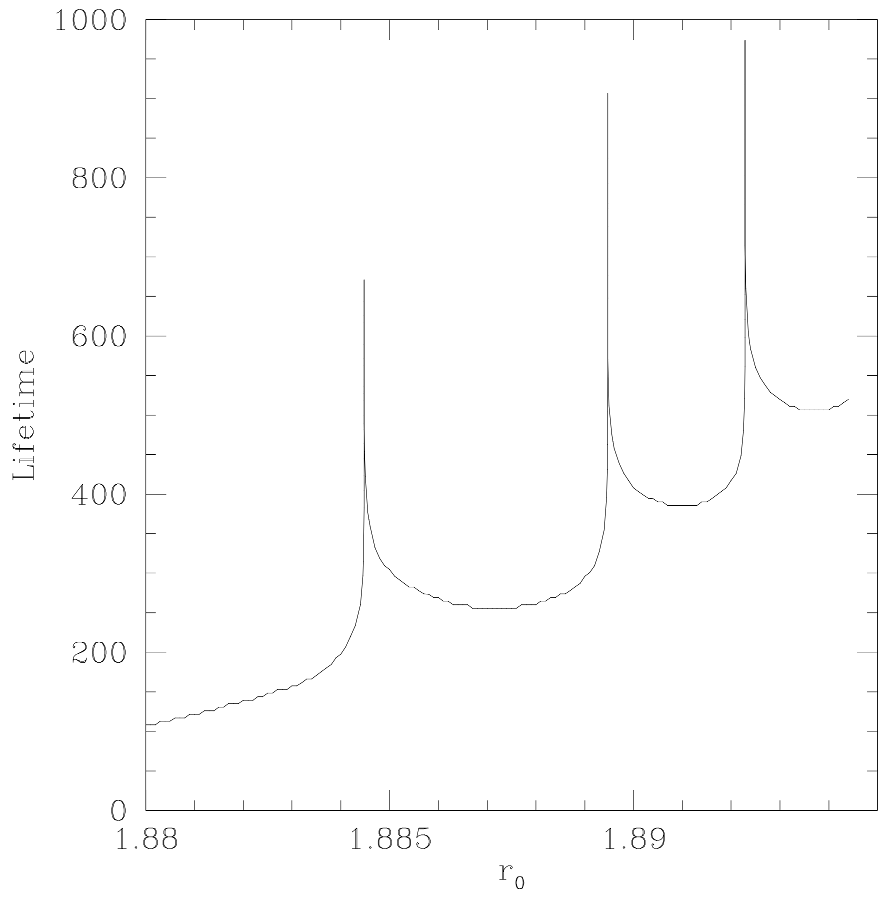

Figure 4.7: This is the lifetime of the oscillon as a function of its initial radius r_0. The three peaks indicate the resonance states that the oscillon may exist in and are shown to exihibit scaling laws of the form $\tau = \gamma \log |r_0 - r_0^*|$. The resonances are at $r_0 = 1.8833766, 1.88946771, 1.89228353$. Below the first resonance, there are no modulations in the amplitude of the field at the origin (see fig. 4.6). Between the first and the second, there is one modulation. Between the second and the third, there are two, and so on.

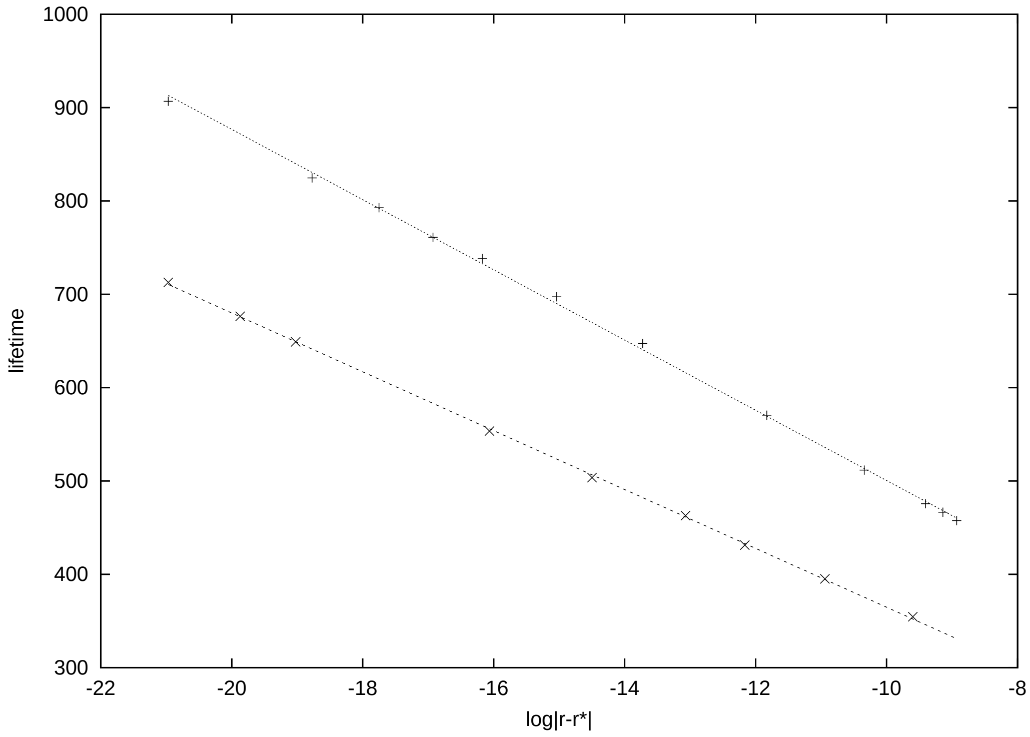

Figure 4.8: The linearity of this data demonstrates their is a time scaling law associated with associated each resonance. These slopes are the left and right side of resonance number two as in fig. 4.7 and respectively are -37 and -31.

Chapter 5

Conclusions

The solutions to the nonlinear Klein-Gordon (nlKG) equation implemented here show that oscillons are localized, have a finite lifetimes, and can exist in resonance states (infinite lifetime) depending on the their initial bubble radii. The critical behaviour of the resonances is characterized by a time scaling law, $\tau = \gamma \log |r_0 - r_0^*|$, where r_0 is the inital radius, r_0^* is the initial radius of a resonant oscillon, τ is the lifetime, and γ is a constant of proportionality that is different for each side of each resonance.

The major source of error here is the computational problem at the outer boundary in which small oscillations with no physical significance occur. These waves propagate toward the interior region and contaminate the solution there, no longer conserving energy. Although they are small enough to maintain the qualitative behaviour of the solution, it cannot be trusted for it accuracy Figure 4.3 shows that the MIB code implemented here is no better than a regular coordinate system with outgoing boundary conditions. The typical bahaviour of the oscillon is to radiate energy at certain stages in its lifetime. The energy travelling in from the outer boundary likely had the opposite effect and fed the oscillon, lengthening its lifetime.

Honda's [5] first three resonance values are $r_0 = 2.2805$, 2.2838, 2.2876. Here they are $r_0 = 1.8833$, 1.8895, 1.8923, a difference of roughly 0.4 in each case. The lifetimes calculated here are generally longer, but agree within an order of magnitude. The scaling law exponents, γ, are similar depending on the resonance. Honda showed for the resonance at $r_0 = 2.335$ that $\gamma = -30$ for both sides of the resonance. Here $\gamma = -31$, -37 for a different (the second) resonance. This difference in the scaling exponent is likely due to the difference in precision of the resonance positions in the parameter space r_0. Honda's were tuned to one part in 10^{14}, whereas here they are tuned to one part in 10^7.

References

[**1**] I. L. Bogolyugskii, V. G. Makhan;kov, *JETP Letters* 24, Page: 12. 1976.

[**2**] E. J. Copeland, M. Gleiser, H.-R. Müller. *Oscillons: Resonant Configurations during Bubble Collapse.* Physical Review D: Vol. 52, No. 4. 15 August 1995.

[**3**] P. G. Drazin, R. S. Johnson. *Solitons: An Introduction.* Cambridge University Press: Cambridge, 1989. ISBN: 0521 33389 X. Page: 183.

[**4**] J. N. Homenuke. *Animated Solutions to the Wave and Sine-Gordon Equations.* `http://laplace.physics.ubc.ca/People/jhomenuk/waveSolutions.html`

[**5**] E. P. Honda. Ph. D. Dissertation. University of Texas at Austin, 2000.

[**6**] H.-O. Kreiss, J. Oliger. "Methods for Approximate Solution of Time Dependent Problems". Global Atmospheric Research Program Publication No. 10. World Meteorological Organization. Case Potsdale No. 1, CH-1211 Geneva 20, Switzerland, 1973.

Appendix A

Derivation of Finite-difference Operators

Below are the derivations of some of the finite-difference (FD) operators used in this work. Taylor series expansions are combined then truncated to $\mathrm{O}(h^2)$ accuracy where h is the discretization scale, a generalization of Δt or Δr. Subscripts are used to denote partial differentiation.

Forward First Time Derivitive

Forward can be interpretted as an expansion about a point ahead of t^n. In this case, it is $t^{n+1/2}$.

$$\begin{aligned} u(t+\Delta t) &= u\left(\left(t+\frac{\Delta t}{2}\right)+\frac{\Delta t}{2}\right) \\ &= u\left(t+\frac{\Delta t}{2}\right)+u_t\left(t+\frac{\Delta t}{2}\right)\cdot\left(\frac{\Delta t}{2}\right)+\frac{1}{2!}\cdot u_{tt}\left(t+\frac{\Delta t}{2}\right)\cdot\left(\frac{\Delta t}{2}\right)^2 \\ &\quad+\frac{1}{3!}\cdot u_{ttt}\left(t+\frac{\Delta t}{2}\right)\cdot\left(\frac{\Delta t}{2}\right)^3+\mathrm{O}(\Delta t^4) \end{aligned} \tag{A.1}$$

$$\begin{aligned} u(t) &= u\left(\left(t+\frac{\Delta t}{2}\right)-\frac{\Delta t}{2}\right) \\ &= u\left(t+\frac{\Delta t}{2}\right)-u_t\left(t+\frac{\Delta t}{2}\right)\cdot\left(\frac{\Delta t}{2}\right)+\frac{1}{2!}\cdot u_{tt}\left(t+\frac{\Delta t}{2}\right)\cdot\left(\frac{\Delta t}{2}\right)^2 \\ &\quad-\frac{1}{3!}\cdot u_{ttt}\left(t+\frac{\Delta t}{2}\right)\cdot\left(\frac{\Delta t}{2}\right)^3+\mathrm{O}(\Delta t^4) \end{aligned} \tag{A.2}$$

Subtracting equation (A.2) from (A.1) and dividing by Δt,

$$\frac{u(t+\Delta t)-u(t)}{\Delta t}=u_t\left(t+\frac{\Delta t}{2}\right)+\underbrace{\frac{1}{24}\cdot u_{ttt}\left(t+\frac{\Delta t}{2}\right)\cdot\Delta t^2}_{\text{LEADING ORDER ERROR TERM}}+\mathrm{O}(\Delta t^4).$$

Then re-writing this in terms of n and j,

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = u_t\Big|_j^{n+1/2} + \mathrm{O}(\Delta t^2) = \Delta_t^{\mathrm{f}}(u_j^n). \tag{A.3}$$

Foward and Backward First Spatial Derivitives

The choice of expansions is slightly different here. A system of three Taylor expansions is solved for $u_r(r)$ such that the next term is $\mathrm{O}(h^2)$.

$$u(r) = u(r) \tag{A.4}$$

$$u(r+\Delta r) = u(r) + u_r(r)\Delta r + \frac{1}{2!}u_{rr}(r)\Delta r^2 + \frac{1}{3!}u_{rrr}(r)\Delta r^3 + \mathrm{O}(\Delta r^4) \tag{A.5}$$

$$u(r+2\Delta r) = u(r) + 2u_r(r)\Delta r + \frac{4}{2!}u_{rr}(r)\Delta r^2 + \frac{9}{3!}u_{rrr}(r)\Delta r^3 + \mathrm{O}(\Delta r^4) \tag{A.6}$$

Solving for $u_r(r)$,

$$\frac{-3u(r) + 4u(r+\Delta r) - u(r+2\Delta r)}{2\Delta r} = u_r(r) - \underbrace{\frac{1}{4}u_{rrr}(r)\Delta r^2}_{\text{LEADING ORDER ERROR TERM}} + \mathrm{O}(\Delta r^3)$$

Re-writing this in terms of n and j, the forward derivative is

$$\frac{-3u_j^n + 4u_{j+1}^n - u_{j+2}^n}{2\Delta r} = u_r\Big|_j^n + \mathrm{O}(\Delta t^2) = \Delta_r^{\mathrm{f}}(u_j^n). \tag{A.7}$$

The backward derivative is found by reversing the minus signs in equations (A.5) and (A.6).

$$\frac{3u(r) - 4u(r+\Delta r) + u(r+2\Delta r)}{2\Delta r} = u_r(r) + \underbrace{\frac{1}{4}u_{rrr}(r)\Delta r^2}_{\text{LEADING ORDER ERROR TERM}} + \mathrm{O}(\Delta r^3)$$

$$\frac{3u_j^n - 4u_{j-1}^n + u_{j-2}^n}{2\Delta r} = u_r\Big|_j^n + \mathrm{O}(\Delta t^2) = \Delta_r^{\mathrm{f}}(u_j^n) \tag{A.8}$$

Centered First Spatial Derivitive

The centered spatial derivitive operator is determined as follows.

$$u(r+\Delta r) = u(r) + u_r(r)\Delta r + \frac{1}{2!}u_{rr}(r)\Delta r^2 + \frac{1}{3!}u_{rrr}(r)\Delta r^3 + \mathrm{O}(\Delta r^4) \tag{A.9}$$

$$u(r-\Delta r) = u(r) - u_r(r)\Delta r + \frac{1}{2!}u_{rr}(r)\Delta r^2 - \frac{1}{3!}u_{rrr}(r)\Delta r^3 + \mathrm{O}(\Delta r^4) \tag{A.10}$$

Then subtracting equation (A.10) from (A.9) and dividing by $2\Delta r$,

$$\frac{u(r+\Delta r) - u(r-\Delta r)}{2\Delta r} = u_r(r) + \underbrace{\frac{1}{3}u_{rrr}(r)\Delta r^2}_{\text{LEADING ORDER ERROR TERM}} + \mathrm{O}(\Delta r^4)$$

Re-writing this in terms of n and j,

$$\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta r} = u_r\Big|_j^n + \mathrm{O}(\Delta r^2) = \Delta_r^{\mathrm{c}}(u_j^n). \tag{A.11}$$

Forward Time Average

The forward time averaging operator (expanded around the point $t^{n+1/2}$) is developed identically to Δ_t^{f}, except that (A.2) and (A.1) are averaged.

$$\frac{u(t+\Delta t)+u(t)}{2} = u\left(t+\frac{\Delta t}{2}\right) + \underbrace{\frac{1}{8}u_{tt}\left(t+\frac{\Delta t}{2}\right)\Delta t^2}_{\substack{\text{LEADING ORDER}\\ \text{ERROR TERM}}} + \mathrm{O}(\Delta t^4)$$

This is re-written as

$$\frac{u_j^{n+1}+u_j^n}{2} = u\Big|_j^{n+1/2} + \mathrm{O}(\Delta t^2) = \mu_t^{\mathrm{f}}(u_j^n). \tag{A.12}$$

Appendix B

RNPL Source Code

```
###########################################################
# Solution to spherically symmetric KG Equation
# MIB coordinates, symmetric double well potential
# 3 equations of motion
#
# John Homenuke
# June 17, 2004
###########################################################


###########################################################
# Parameters
###########################################################

# Maximum memory allowance. Necessary for Fortran code.

system parameter int memsiz:= 1000000

# Physical Inner and outer boundaries

parameter float rmin    := 0.0
parameter float rmax    := 20.0

# Other various parameters

parameter float pie     := 3.14159265358979
parameter float rw      := 16.67
parameter float r0      := 2.0
parameter float phi0    := -1.0
parameter float phic    := 1.0
parameter float delr    := 0.0893
parameter float sigma   := 1.0
parameter float epsilon := 0.5


###########################################################
# Grid Definition
###########################################################
```

```
# 'spherical' is a user-specified name of the coordinate system.
# t and r are the independent variables.  The first one is always
# the time coordinate.

spherical coordinates t,r

# 'uniform' means the grid points are evenly spaced.  A 'nonuniform'
# option has yet to be introduced into the language.  'g1' is the
# name of the grid.  '[1:Nr]' indexes the spatial grid points from
# 1 to Nr.  In the temporal direction, the grid is bounded at run-
# time.

uniform spherical grid g1 [1:Nr] {rmin:rmax}

# 'phi', 'pi', etc. are the names of the grid functions.  'at 0,1'
# indicates the time-steps that the equations span at any one point
# in time, i.e. they contain terms indexed at n and n+1.  If they
# included terms at n-1, n, and n+1, then the code would say,
# 'at -1,0,1'.  The function 'ff' is the function f(r).  It is not
# evolved in time so the 'at 0,1' is left out. '{out_gf := 1}' means
# the value function of the function is written to file while the
# computations happen.  '0' means they are not.

float phi on g1 at 0,1      {out_gf := 1}
float pi  on g1 at 0,1      {out_gf := 0}
float pp  on g1 at 0,1      {out_gf := 0}
float ff  on g1             {out_gf := 0}


############################################################
# Finite-difference Operators
############################################################

# Again, a relative notation is used here.  The pointy brackets
# enclose the time-step relative to n and the square brackets
# enclose the spatial step relative to j.  '<1>f[0]' is the
# same as saying 'f evaluated at n+1 and j'.  The square brackets
# can enclose a vector for equations of higher spatial dimension.
# dt and dr are the spacings between neighbouring points.

operator DF(f,t) := (<1>f[0] - <0>f[0])/dt
operator MF(f,t) := (<1>f[0] + <0>f[0])/2
operator DC(f,r) := (<0>f[1] - <0>f[-1])/(2*dr)
operator DF(f,r) := (-3*<0>f[0] + 4*<0>f[1]  - <0>f[2])/(2*dr)
operator DB(f,r) := ( 3*<0>f[0] - 4*<0>f[-1] + <0>f[-2])/(2*dr)
operator D3(f,r) := (<0>f[1] - <0>f[-1])/(r[1]^3 - r[-1]^3)
operator MD(f,t) := -epsilon*(6*<0>f[0] + <0>f[-2] + <0>f[2] - 4*(<0>f[-1]
                    + <0>f[1]))/(16*dt)


############################################################
# Equations of motion with BCs
############################################################
```

```
# Here, the discrete equations of motion are entered, telling the machine
# how to calulate the residual.  The operators are used by replacing 'f'
# with a grid function or independent variable.  The array indices at the
# left denote which grid points to apply the equations at the right to.
# The <> and [] need not be included if they are both 0.

evaluate residual pi {

   [1:1]         := DF(<1>pi[0],r) = 0;
   [2:2]         := DF(pi,t) = MF( 3/(1+ff*t/r)^2 * D3( (r+ff*t)^2*(pp+ff*pi)/
                              (1+DF(ff,r)*t) ,r) - 2*ff*pi/(r+ff*t) - phi^3 + phi ,t);
   [3:Nr-2]      := DF(pi,t) = MF( 3/(1+ff*t/r)^2 * D3( (r+ff*t)^2*(pp+ff*pi)/
                              (1+DC(ff,r)*t) ,r) - 2*ff*pi/(r+ff*t) - phi^3 + phi ,t)
                              + MD(pi,t);
   [Nr-1:Nr-1]   := DF(pi,t) = MF( 3/(1+ff*t/r)^2 * D3( (r+ff*t)^2*(pp+ff*pi)/
                              (1+DB(ff,r)*t) ,r) - 2*ff*pi/(r+ff*t) - phi^3 + phi ,t);
   [Nr:Nr]       := DF(r*pi,t) + MF(DB(r*pi,r),t) = 0;
}

evaluate residual pp{

   [1:1]         := <1>pp[0] = 0;
   [2:2]         := DF(pp,t)  = MF( DF( ( pi + ff*pp ) / ( 1 + DC(ff,r)*t ) ,r) ,t);
   [3:Nr-2]      := DF(pp,t)  = MF( DC( ( pi + ff*pp ) / ( 1 + DC(ff,r)*t ) ,r) ,t)
                               + MD(pp,t);
   [Nr-1:Nr-1]   := DF(pp,t)  = MF( DB( ( pi + ff*pp ) / ( 1 + DC(ff,r)*t ) ,r) ,t);
   [Nr:Nr]       := DF(r*pp,t) + MF(DB(r*pp,r),t) = 0;
}

evaluate residual phi {

   [1:1]         := DF(<1>phi[0],r) = 0;
   [2:2]         := DF(phi,t) = MF( ( pi + ff*pp ) / ( 1 + DC(ff,r)*t ) ,t);
   [3:Nr-2]      := DF(phi,t) = MF( ( pi + ff*pp ) / ( 1 + DC(ff,r)*t ) ,t) + MD(phi,t);
   [Nr-1:Nr-1]   := DF(phi,t) = MF( ( pi + ff*pp ) / ( 1 + DC(ff,r)*t ) ,t);
   [Nr:Nr]       := DF(r*phi,t) + MF(DB(r*phi,r),t) = 0;
}


###########################################################
# Initial Conditions
###########################################################

# These declarations follow the same rules as the residuals, but are expressions
# rather than equations.

initialize phi {
   [1:Nr]    := phi0 + (phic-phi0)*exp(-r^2/r0^2);
}

initialize ff {
   [1:Nr]    := (tanh((r-rw)/(delr*rw))+tanh(rw/(delr*rw)))/2;
}

initialize pi {
   [1:1]    := 0;
   [2:Nr-1] := sigma*DC(phi,r);
   [Nr:Nr]  := 0
}

initialize pp {
   [1:1]    := 0;
   [2:Nr-1] := DC(phi,r);
   [Nr:Nr]  := 0
}


###########################################################
# Instructions
###########################################################

# Here the machine is told how to run the computations.  'looper iterative'
# is only necessary when the scheme is implicit.  The initial conditions
# are always contained in a separate .sdf file, so can be supplied by the
# user.  'auto initialize' automatically writes the file for the user from
# the 'initialize' command above.  'auto update' is the command that evolves
# the solution in time.

looper iterative
auto initialize phi, pi, pp, ff
auto update phi, pi, pp


###########################################################
# END of file
###########################################################
```