8-2015

# Higgs Boson equation in de Sitter spacetime: Numerical investigation of bubbles using GPUs

Jacob N. Banda
*University of Texas-Pan American*

# HIGGS BOSON EQUATION IN DE SITTER SPACETIME: NUMERICAL INVESTIGATION OF BUBBLES USING GPUS

A Thesis

by

JACOB N. BANDA

Submitted to the Graduate School of
The University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2015

Major Subject: Mathematics

# HIGGS BOSON EQUATION IN DE SITTER SPACETIME: NUMERICAL INVESTIGATION OF BUBBLES USING GPUS

A Thesis
by
JACOB N. BANDA

COMMITTEE MEMBERS

Dr. Andras Balogh
Chair of Committee

Dr. Dambaru Bhatta
Committee Member

Dr. Anahit Galstyan
Committee Member

Dr. Karen Yagdjian
Committee Member

August 2015

## ABSTRACT

Banda, Jacob N., Higgs Boson Equation in de Sitter Spacetime: Numerical Investigation of Bubbles using GPUs. Master of Science (MS), August, 2015, 47 pp., 12 figures, 30 references, 24 titles.

The Higgs field, along with its corresponding boson, represent a milestone for modern day particle physics. In this work we consider the Higgs boson equation in de Sitter spacetime. Previous work by K. Yagdjian [23] has formulated sufficient conditions for the existence of the zeros of global solutions in the interior of their supports. In searching for such solutions, we turn to heterogeneous parallel computing, which allows for faster computation through graphical processing units (GPUs). Armed with general-purpose computation on graphics hardware (GPGPU) techniques and explicit numerical schemes, we approximate solutions of the equation for the Higgs boson along with the creation, growth, and interaction of the zeros, or bubbles.

# DEDICATION

I would very much like to dedicate this entire work to my immediate family members; each of them has offered me support throughout my graduate career and life in general. Thank you mom and dad for your unending care, love, and encouragement; to my sisters, thank you for your patience and always looking out for me.

## ACKNOWLEDGMENTS

In no way could this work have been possible without the guidance of Dr. Andras Balogh, the foundations of Dr. Karen Yagdjian and David Medina, and the disposition of Dr. Anahit Galstyan and Dr. Dambaru Bhatta. Specifically, I must thank Dr. Balogh for allowing me to collaborate with him on this topic. In addition to his department workload, he dedicated himself until the very end, ensuring that we could deliver this thesis as a contribution to the scientific community. Furthermore, the ideas reflected here are an application of work done by Mr. David Medina, and as such I wish to thank him as well. We also acknowledge the Experimental Algebra and Geometry Lab (EAGL) that provided the high-performance GPU machine for our work. This work was partially supported by the U.S. Department of Education UTPA-MATH GAANN program under grant number P200A120256.

TABLE OF CONTENTS

LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

In this work we will consider the partial differential equation

$$\phi_{tt} = -3\phi_t + e^{-2t}\Delta\phi + \mu^2\phi - \lambda\phi^3, \tag{1.1}$$

where $\phi = \phi(\mathbf{x},t)$, $\mathbf{x} \in \mathbb{R}^3$, $t > 0$ and parameters $\mu, \lambda > 0$, with initial conditions

$$\begin{aligned} \phi(\mathbf{x},0) &= \phi_0(\mathbf{x}), & (1.2)\\ \phi_t(\mathbf{x},0) &= \phi_1(\mathbf{x}). & (1.3) \end{aligned}$$

Equation (1.1) is a nonlinear wave equation with cubic nonlinearity and time dependent coefficient $e^{-2t}$. This equation represents the Higgs real-valued scalar field in de Sitter spacetime. Longtime existence of solution for this equation is an open problem. Research by Yagdjian [23] formulated sufficient conditions for the existence of the zeros of global solutions.

We wish to approximate a solution to equation (1.1) with finite differences in order to investigate the creation, growth, and interaction of the so-called bubbles, or zeros, of this solution. It should be noted that the Higgs field and boson play a vital role in the current standard model of particle physics. While it is not known that the solution exists for all time, the next approach would be to approximate the solution through numerical methods. In searching for such solutions, we turn to heterogeneous parallel computing, which allows for faster computation through graphical processing units (GPUs). We shall discuss the advantages of pursuing such an approach, along with the programming languages employed in the simulation. In the simulation study, we present two case studies under different initial conditions.

The rest of the thesis is organized in the following manner. In Chapter II. we discuss the role of the Higgs boson in the current standard model, along with the implications that arise from the Higgs mechanism. Furthermore, we describe the sufficient conditions for the existence of such bubbles, derived by K. Yagdjian [23]. In Chapter III. we discuss the choice of numerical method, the effects of scaling, the discretized form of equation (1.1), and the implementation of GPUs. In Chapter IV., the numerical results of the two case studies are discussed along with their interpretation. Finally, in Chapter V. we summarize our future work.

# CHAPTER II

# THEORETICAL RESULTS

## 2.1 Higgs boson history

In particle physics, the standard model describes elementary particles that have been discovered to date. We can employ a simplified classification scheme in order to understand the relationships that exist between the particles. The three classifications that we are interested in are 1) quarks, 2) leptons, and 3) bosons (force carriers) [1, 9]. It should be noted that all *visible* matter is made up of elementary particles.[1] The particles that make up matter are considered fermions; quarks and leptons fall in this category. While quarks can be combined to form hadrons, leptons can exist without binding to each other. The naming convention for hadrons describes the number of quarks that have been combined: a union of three quarks is known as a baryon and a union of two quarks is known as a meson. Figure 2.1 presents the six known quarks to date. Figure 2.2 presents the six known leptons to date.

[1]We do not concern ourselves with dark matter or dark energy in this work.

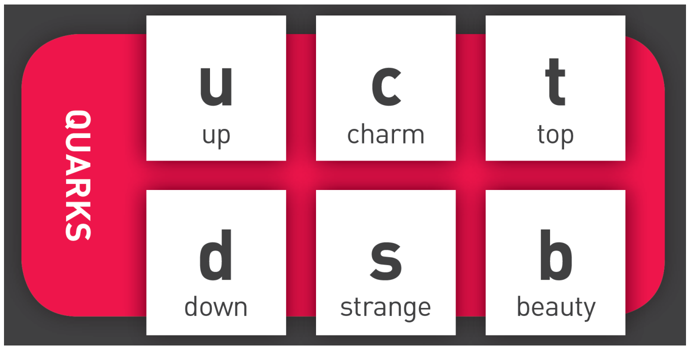


Figure 2.1: Standard model: quarks [21]

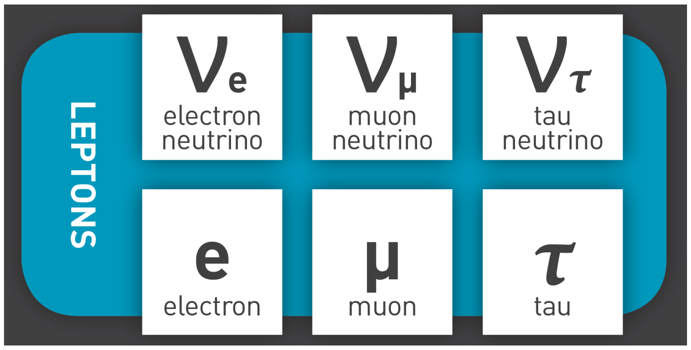


Figure 2.2: Standard model: leptons [21]

Note that the particles are related in pairs, or "generations". There are three generations for both the quarks and leptons (moving from left to right in Figures 2.1 and 2.2). The first generation is the lightest and most stable, whilst the second and third generations are heavier and less stable.

While we have accounted for twelve of the elementary particles of the standard model, the remaining particles fall under the classification of force carriers. There are four fundamental forces at work: 1) strong force, 2) weak force, 3) electromagnetic force, and 4) gravitational force. Both the strong and weak forces are short range forces, while both the electromagnetic and gravitational forces are long range forces. To date, three of the forces have been shown to result from the exchange of force-carrier particles, or "bosons". These are the strong, weak, and electromagnetic force. Their carrier particles are the gluon, W and Z bosons, and the photon, respectively. The gravitational force-carrier particle has come under great scrutiny these past few years and is posited to be the Higgs boson. Figure 2.3 presents these force-carrier particles. The standard model of particle physics enjoyed advances, but with some resulting issues. In 1956, Julian Schwinger [17] posited that a gauge theory might exist which could help understand the weak interactions. Such a gauge theory would involve a pair of mediating particles, or gauge bosons, called $W^+$ and $W^-$. Furthermore, he suggested that there might be a unified theory of weak and electromagnetic interactions. He claimed that such a theory would in fact involve some kind of symmetry between the three gauge bosons: the $W^+$, the $W^-$, and the photon. In 1961, Sheldon Glashow [8] added a fourth gauge boson, $Z$, to cure a problem with mirror symmetry.

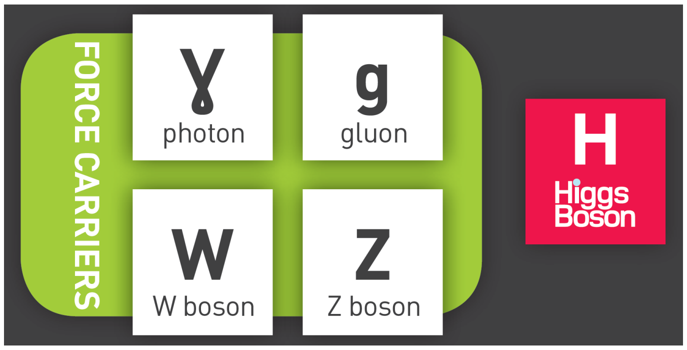


Figure 2.3: Standard model: force carriers [21]

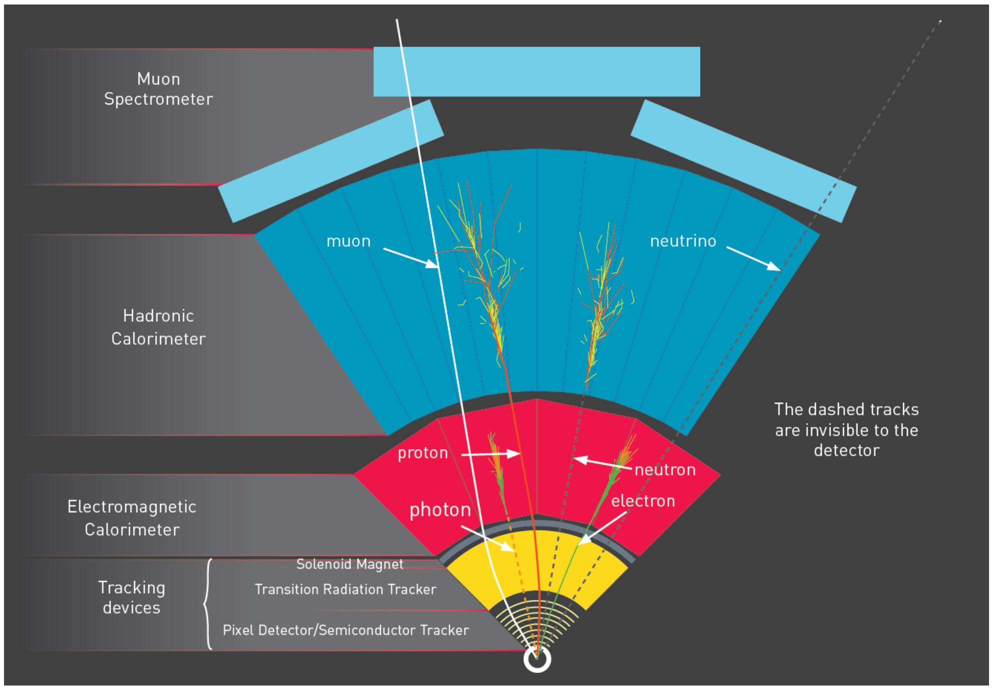
However, the electroweak unification theory in physics could not account for the non-zero masses of the W and Z bosons. In fact, to explain the short range and weakness of the weak interactions, it is imperative that these particles be very heavy. The measured values, $M_W = 80$ GeV and $M_Z = 91$ GeV are far from zero and cannot be considered as small effects. If there is some kind of symmetry between these four gauge bosons, it must be broken in some way. One cannot simply add masses to these bosons by hand. It is here that we introduce the Higgs mechanism [7, 10, 11]. In short, the Higgs field would allow elementary particles to obtain mass. The corresponding Higgs boson is posited to be the missing force-carrying particle for gravity in the current standard model.

Today, physicists are hard at work collecting data at the CERN laboratory in Geneva, Switzerland and employing tools such as the Large Hadron Collider (LHC) and the ATLAS particle detector experiment [19] in hopes of confirming that the newly discovered boson is in fact, without a doubt, the Higgs boson and complements the current standard model.

## 2.2 Mathematical results

In this section we briefly touch the sufficient conditions for zeros that have been proven by Dr. Karen Yagdjian. We refer the reader to his work [23] for a more detailed explanation.

**Theorem 1.** *Let* $\phi = \phi(\boldsymbol{x},t) \in C\left(\mathbb{R}^3 \times [0,\infty)\right)$ *be a weak global solution of the real field equation (1.1). Let* $\psi = \psi(x)$ *be an eigenfunction of the Laplace operator in* $\mathbb{R}^3$ *corresponding to the*



Muon Spectrometer
Hadronic Calorimeter
Electromagnetic Calorimeter
Tracking devices
Solenoid Magnet
Transition Radiation Tracker
Pixel Detector/Semiconductor Tracker
muon
neutrino
proton
neutron
photon
electron
The dashed tracks are invisible to the detector

Figure 2.4: ATLAS particle detector experiment: cross-sectional view [21]

*eigenvalue* $\nu$. *Denote by*

$$C_0(\phi,\psi) \equiv \int_{\mathbb{R}^3} \psi(x)\phi(x,0)dx, \quad C_1(\phi,\psi) \equiv \int_{\mathbb{R}^3} \psi(x)\phi_t(x,0)dx, \tag{2.1}$$

*the integrals (functionals) of its* $\psi$*- weighted initial values. Assume that*

$$\left(\sqrt{9+4\left(\mu^2+\nu\right)}+3\right)C_0(\phi,\psi)+2C_1(\phi,\psi)>0. \tag{2.2}$$

*Assume also that the* $\psi$*- weighted self-interaction functional* $-\lambda\int_{\mathbb{R}^3}\psi(x)\phi^3(x,t)dx$ *satisfies*

$$\int_{\mathbb{R}^3}\psi(x)\phi^3(x,t)dx \leq 0 \tag{2.3}$$

*for all t either outside of the sufficiently small neighborhood of zero if* $\mu^2+\nu>0$, *or inside of some neighborhood of infinity if* $\mu^2+\nu=0$.

*Then the global solution* $\phi=\phi(x,t)$ *cannot be an asymptotically time-weighted* $-\psi$ $L^3$*-signed with the weight* $\nu_\phi = e^{a_{\phi,\psi}t}t^{b_{\phi,\psi}}$, *where if* $\mu^2+\nu>0$, *then either* $a_{\phi,\psi}<\sqrt{9+4\left(\mu^2+\nu\right)}-3$ *or* $a_{\phi,\psi}=\sqrt{9+4\left(\mu^2+\nu\right)}-3$ *and* $b_{\phi,\psi}<-2$, *while* $a_{\phi,\psi}=0$ *and* $b_{\phi,\psi}\leq 4$ *if* $\mu^2+\nu=0$.

Yagdjian proves that the continuous global solution obtained by prolongation of a local solution must change a sign. Indeed, the global solution is an oscillating in time solution. Hence, from Theorem 1 (or Theorem 4.1 in Yagdjian's work) we find the following two sufficient conditions:

$$\left(\sqrt{9+4\left(\mu^2+\nu\right)}+3\right)C_0(\phi,\psi)+2C_1(\phi,\psi)>0 \tag{2.4}$$

$$\int_{\mathbb{R}^3}\psi(x)\phi^3(x,t)dx\leq 0 \tag{2.5}$$

# CHAPTER III

# NUMERICAL TREATMENT OF HIGGS BOSON

## 3.1 Choice of numerical method

In choosing the appropriate numerical method, it is obvious that there are an abundance of methods to choose from, each with their own strengths and drawbacks. Indeed, one could argue for or against a particular case. However, in searching for a feasible method, it was our goal to maintain simplicity while balancing three key factors: 1) accuracy, 2) storage, and 3) time. In effect, the designated method was finite difference approximation resulting in explicit, forward Euler method with second order accuracy implemented on graphical processing units (GPUs). Compared to explicit methods, the advantage of implicit methods is that they are absolute stable, while explicit methods are conditionally stable, requiring a smaller time step size for stability of the method. However, the GPU implementation of explicit methods supersedes this requirement. On the other hand, for the wave equation the implicit method creates more dispersion from the exact phase velocity than the explicit method. For this reason we are not considering implicit schemes.

It should be noted that implementing the calculations on GPUs allows for faster calculations under *certain* settings. Not all numerical schemes lend themselves to parallel calculations. It is up to the programmer to find these parallel routines and exploit them. By choosing an explicit, forward-in-time method we were able to exploit these parallel routines and not be held back by the time restrictions of serial computations. Such examples include parallel computations for ordinary differential equations or while using Runge-Kutta methods [2, 13]. In this way we were able to overcome the intense calculations over many data points that would appear in our simulations.

## 3.2 Cone of propagation and scaling

In the case of the linear wave equation

$$w_{tt} + c^2 \Delta w = 0 \tag{3.1}$$

the constant $c$ represents the speed of propagation. The initial signal $w$ supported in a finite region $\Omega$, i.e. with zero values outside, will travel with velocity $c$ in all direction of space forever. In the case of the modified linear wave equation

$$w_{tt} + e^{-2t} \Delta w = 0 \tag{3.2}$$

the time dependent expression $e^{-t}$ represents the speed of propagation. The initial signal $w$ supported in a finite region $\Omega$, i.e. with zero values outside, will travel with velocity $e^{-t}$ in all direction of space with total distance

$$\int_0^\infty e^{-t} dt = 1. \tag{3.3}$$

This finite traveled distance allows for the computations to be restricted to a finite region. Furthermore, this permits the use of zero boundary conditions.

The size of the computational domain depends on the size of the support of the initial conditions. In order to use the unit box for the computational domain we rescale equation (1.1) with the appropriate transformation $\mathbf{x} \to \frac{\mathbf{x}}{L}$. Therefore, the scaled equation would become

$$\phi_{tt} = -3\phi_t + \frac{e^{-2t}}{L^2} \Delta \phi + \mu^2 \phi - \lambda \phi^3, \quad \mathbf{x} \in [0,1]^3, t > 0 \tag{3.4}$$

for some constant $L > 0$. Rescaling the equation will allow us to restrict the wave from ever reaching the boundary given the correct initial conditions.

## 3.3 Finite Difference Discretization

We will provide a derivation for the discretized 3-D Higgs boson equation in the de Sitter spacetime. Research by Yagdjian [23] provides the corresponding equation for the Higgs real-valued scalar field in the de Sitter Spacetime. After scaling, we have

$$\phi_{tt} = -3\phi_t + \frac{e^{-2t}}{L^2} \Delta \phi + \mu^2 \phi - \lambda \phi^3, \quad \mathbf{x} \in [0,1]^3, t > 0 \tag{3.5}$$

We can express $\phi_t$, $\phi_{tt}$, $\phi_{xx}$, $\phi_{yy}$, and $\phi_{zz}$ using the definition of central finite difference quotients [3, 12]. For example,

$$\phi_t(t,x,y,z) = \frac{\phi(t+\Delta t,x,y,z) - \phi(t-\Delta t,x,y,z)}{2\Delta t} + \mathscr{O}(\Delta t^2) \tag{3.6}$$

Then the second partial time derivative is as follows.

$$\phi_{tt}(t,x,y,z) = \frac{\phi(t+\Delta t,x,y,z) - 2\phi(t,x,y,z) + \phi(t-\Delta t,x,y,z)}{(\Delta t)^2} + \mathscr{O}(\Delta t^2) \tag{3.7}$$

Similarly,

$$\phi_{xx}(t,x,y,z) = \frac{\phi(t,x+\Delta x,y,z) - 2\phi(t,x,y,z) + \phi(t,x-\Delta x,y,z)}{(\Delta x)^2} + \mathscr{O}(\Delta x^2) \tag{3.8}$$

$$\phi_{yy}(t,x,y,z) = \frac{\phi(t,x,y+\Delta y,z) - 2\phi(t,x,y,z) + \phi(t,x,y-\Delta y,z)}{(\Delta y)^2} + \mathscr{O}(\Delta y^2) \tag{3.9}$$

$$\phi_{zz}(t,x,y,z) = \frac{\phi(t,x,y,z+\Delta z) - 2\phi(t,x,y,z) + \phi(t,x,y,z-\Delta z)}{(\Delta z)^2} + \mathscr{O}(\Delta z^2) \tag{3.10}$$

Substituting our results into equation (3.5), we obtain the second order approximation

$$\begin{aligned}\frac{\phi(t+\Delta t,x,y,z) - 2\phi + \phi(t-\Delta t,x,y,z)}{(\Delta t)^2} &= -3\left(\frac{\phi(t+\Delta t,x,y,z) - \phi(t-\Delta t,x,y,z)}{2\Delta t}\right) \\ &+ \frac{e^{-2t}}{L^2}\left(\frac{\phi(t,x+\Delta x,y,z) - 2\phi + \phi(t,x-\Delta x,y,z)}{(\Delta x)^2} + \frac{\phi(t,x,y+\Delta y,z) - 2\phi + \phi(t,x,y-\Delta y,z)}{(\Delta y)^2}\right. \\ &\left. + \frac{\phi(t,x,y,z+\Delta z) - 2\phi + \phi(t,x,y,z-\Delta z)}{(\Delta z)^2}\right) + \mu^2\phi - \lambda\phi^3. \end{aligned} \tag{3.11}$$

Let $\Delta x = \Delta y = \Delta z$. Furthermore, designate $\phi^i_{jkl}$ to represent the i-th time step along the three dimensions j,k, and l at their respective intervals. Then equation (3.11) becomes

$$\begin{aligned}\frac{\phi^{i+1}_{jkl} - 2\phi^i_{jkl} + \phi^{i-1}_{jkl}}{(\Delta t)^2} &= -3\left(\frac{\phi^{i+1}_{jkl} - \phi^{i-1}_{jkl}}{2\Delta t}\right) + \frac{e^{-2t}}{L^2}\left(\frac{\phi^i_{j+1,k,l} - 2\phi^i_{jkl} + \phi^i_{j-1,k,l}}{(\Delta x)^2}\right. \\ &\left. + \frac{\phi^i_{j,k+1,l} - 2\phi^i_{jkl} + \phi^i_{j,k-1,l}}{(\Delta x)^2} + \frac{\phi^i_{j,k,l+1} - 2\phi^i_{jkl} + \phi^i_{j,k,l-1}}{(\Delta x)^2}\right) + \mu^2\phi - \lambda\phi^3.\end{aligned}$$

Solving for $\phi^{i+1}_{jkl}$.

$$\begin{aligned}\phi^{i+1}_{jkl} - 2\phi^i_{jkl} + \phi^{i-1}_{jkl} &= \frac{-3\Delta t}{2}\left(\phi^{i+1}_{jkl} - \phi^{i-1}_{jkl}\right) + \frac{e^{-2t}}{L^2}\left(\frac{\Delta t}{\Delta x}\right)^2\left(\phi^i_{j+1,k,l} - 2\phi^i_{jkl} + \phi^i_{j-1,k,l}\right. \\ &\left. + \phi^i_{j,k+1,l} - 2\phi^i_{jkl} + \phi^i_{j,k-1,l} + \phi^i_{j,k,l+1} - 2\phi^i_{jkl} + \phi^i_{j,k,l-1}\right) + (\Delta t)^2[\mu^2\phi^i_{jkl} - \lambda(\phi^i_{jkl})^3].\end{aligned} \tag{3.12}$$

Continuing,

$$\phi_{jkl}^{i+1} + \frac{3\Delta t}{2}\phi_{jkl}^{i+1} = (\Delta t)^2\left[\mu^2\phi_{jkl}^{i} - \lambda(\phi_{jkl}^{i})^3\right] + 2\phi_{jkl}^{i} + \left(\frac{3\Delta t}{2} - 1\right)\phi_{jkl}^{i-1} + \frac{e^{-2t}}{L^2}\left(\frac{\Delta t}{\Delta x}\right)^2\left(\phi_{j+1,k,l}^{i} + \phi_{j-1,k,l}^{i} + \phi_{j,k+1,l}^{i} + \phi_{j,k-1,l}^{i} + \phi_{j,k,l+1}^{i} + \phi_{j,k,l-1}^{i} - 6\phi_{jkl}^{i}\right). \quad (3.13)$$

Finally,

$$\phi_{jkl}^{i+1} = \frac{2(\Delta t)^2}{2+3\Delta t}\left[\mu^2\phi_{jkl}^{i} - \lambda(\phi_{jkl}^{i})^3\right] + \frac{2}{2+3\Delta t}\left[2\phi_{jkl}^{i} + \left(\frac{3\Delta t}{2} - 1\right)\phi_{jkl}^{i-1}\right] + \frac{2e^{-2t}}{L^2(2+3\Delta t)}\left(\frac{\Delta t}{\Delta x}\right)^2\left(\phi_{j+1,k,l}^{i} + \phi_{j-1,k,l}^{i} + \phi_{j,k+1,l}^{i} + \phi_{j,k-1,l}^{i} + \phi_{j,k,l+1}^{i} + \phi_{j,k,l-1}^{i} - 6\phi_{jkl}^{i}\right), \quad (3.14)$$

for $j,k,l = 1,\ldots,N$ and for $i = 1,2,\ldots$. Equation (3.14) is explicit, forward-in-time, Euler method. This method requires $\phi_{jkl}^{0}$ and $\phi_{jkl}^{1}$ specified. In all our simulations we set the initial time derivative $\phi_t(\mathbf{x},0) \equiv 0$. This is implemented as $\phi_{jkl}^{1} = \phi_{jkl}^{0}$.

## 3.4 GPU Implementation

When comparing the power of the graphics processing unit (GPU) over the central processing unit (CPU) or vice versa, many claims exist as to how much more powerful the GPU has become when processing data intensive tasks. At the very least, Lee [15] has shown that a substantial speedup of around 2.5x on average is achievable. Indeed, while the CPU is often referred to as the "brains" of a personal computer, it is composed of only a few cores with a great deal of cache memory. This arrangement lends itself to handling a few software threads at a time. In contrast, the GPU can handle thousands of threads simultaneously due to its composition of hundreds of cores. There was a time when GPUs were used mainly for 3D game rendering. However, as the field of general purpose computation on graphics processing units (GPGPU) has grown, GPUs are also used for data intensive tasks with parallel structure, such as financial modeling, scientific research and oil and gas exploration.

One of the goals of this work was to reduce compute time and seek efficient use of our resources. Naturally, accessing the GPU and repurposing it for our needs would require a new set of programming standards. For this we turn to heterogeneous parallel computing [5, 14] with OpenCL, an open, royalty-free standard for cross-platform, parallel programming of modern processors found in personal computers, servers and handheld/embedded devices [18]. In essence, it allows access to hardware components that would otherwise have been restricted.

Many different standards exist for heterogeneous computing, with some that are even vendor specific (NVidia, Intel, etc.). All these application programming interfaces (APIs) require intimate knowledge of the hardware and significantly more syntax than traditional scientific computations. Figure 3.1 shows a typical "Hello World" program segment written in OpenCL.

Since coding in OpenCL would require more time, we opted to include the OCCA2 library as a unified threading model [16, 22]. OCCA was developed at Rice University and allows for communication with the different API standards set forth by different vendors. The front end remains accessible to the end user, providing coding options in C, C++, MATLAB and other familiar environments. The OCCA library can be found at `https://github.com/tcew/OCCA2`. Figure 3.2 gives a representation of the role that OCCA played in our work. The program code implementing our algorithm can be found in Appendix A. Note the simplicity of setting up the GPU and executing the kernel module.

```
/* Create OpenCL context */
context = clCreateContext(NULL, 1, &device_id, NULL, NULL,
          &ret);
/* Create Command Queue */
command_queue = clCreateCommandQueue(context, device_id, 0,
          &ret);
/* Create Memory Buffer */
memobj = clCreateBuffer(context, CL_MEM_READ_WRITE,
         MEM_SIZE * sizeof(char), NULL, &ret);
/* Create Kernel Program from the source */
program = clCreateProgramWithSource(context, 1,
          (const char **)&source_str,
          (const size_t *)&source_size, &ret);
/* Build Kernel Program */
ret = clBuildProgram(program, 1, &device_id, NULL, NULL, NULL);
/* Create OpenCL Kernel */
kernel = clCreateKernel(program, "hello", &ret);
/* Set OpenCL Kernel Parameters */
ret = clSetKernelArg(kernel, 0, sizeof(cl_mem), (void *)&memobj);
/* Execute OpenCL Kernel */
ret = clEnqueueTask(command_queue, kernel, 0, NULL,NULL);
/* Copy results from the memory buffer */
ret = clEnqueueReadBuffer(command_queue, memobj, CL_TRUE, 0,
MEM_SIZE * sizeof(char),string, 0, NULL, NULL);
```

Figure 3.1: "Hello World" program segment written in OpenCL

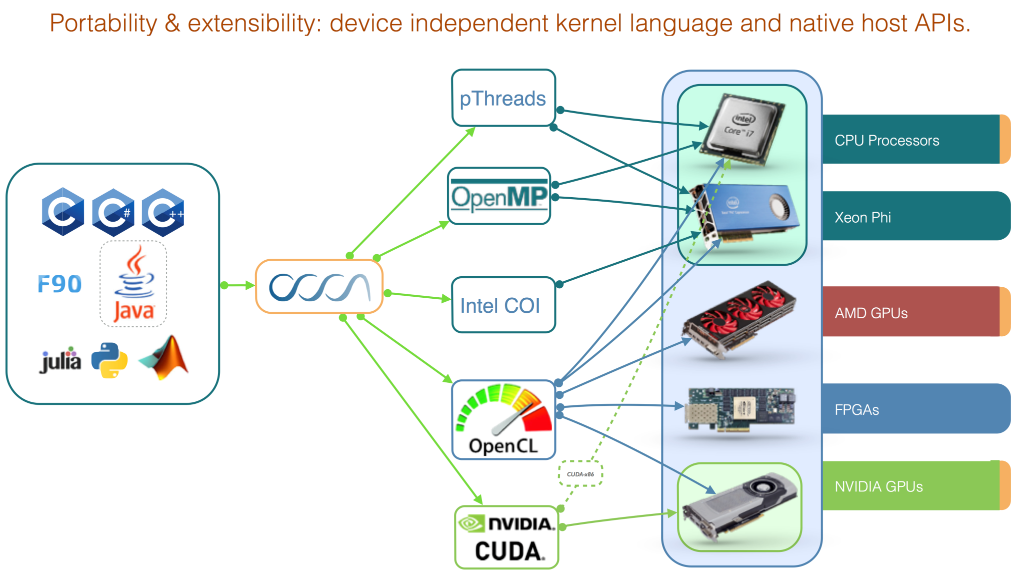


Figure 3.2: OCCA2 - unified threading model [22]

# CHAPTER IV

# SIMULATION STUDY

## 4.1 Visualization with VisIt

The task of visualizing the data was left to VisIt: an open source, interactive, animation and analysis tool [4]. Some of VisIt's key features include visualizing small to large data sets, animating them through time, manipulating them with mathematical expressions and operators, and exporting the resulting images and animations. VisIt accepts over 120 scientific data formats, including the format we chose to use, Hierarchical Data Format 5 [20]. It was typical for us to handle up to 200 data sets, each measuring in size about 4 megabytes (MB), totaling to almost one gigabyte. VisIt also provided distributed architecture capabilities that we implemented for a more streamlined approach. The Viewer application was installed on a Windows or Mac machine, which in turn communicated through SSH with the VisIt Engine running on the Linux server.

When considering *how* to visualize scalar data in three space dimensions and one time dimension, we must emphasize that the matter is not as trivial as the reader might expect. Indeed, even physics textbooks have difficulty portraying sound standing waves of air columns [24]. With VisIt we were able to use volume visualization on the Higgs scalar field in three space dimensions. Using the slice and contour tools we were able to visualize a hierarchy of bubbles embedded in each other.

## 4.2 Case studies

The discretized equation (3.14) is implemented using the C++ code found in Section 1.2 of Appendix A, and the kernel code in Section 1.1 of Appendix A. The spatial domain consists of the unit cube $\mathbf{x} \in [0,1]^3 \subset \mathbb{R}^3$, with $N = 100$ grid points in each of the three spatial directions, resulting in a grid of size $10^6$ and $\Delta x = 0.01$. The scaling parameter, along with other parameters, were set as follows: $L = 2, \mu^2 = 0.1, \lambda = 0.1, \Delta t = 0.005$. The final time was $T = 0.995$. We present numerical simulations for two sets of initial conditions. The first set is one bell-shaped function while the second set is the sum of two bell-shaped functions. For both cases of initial conditions we present three sets of images using different visualization techniques.

The volume plot displays the scalar values using colors in the cube. These images are found in Figures 4.1 and 4.4. The color scale was chosen carefully to provide distinguishable images and the opacity transfer function was chosen to ensure the visibility of both positive and negative values by hiding the values close to zero. The volume plot configuration XML file is in Section 1.5 of Appendix A.

VisIt's contour plot option was used to generate an isosurface indicating the places of zero values where there is a sign change in the Higgs boson scalar field (the so-called bubbles). This was actually combined with the volume plot to generate the images in Figures 4.2 and 4.5. The color green represents these zero values. The contour plot configuration XML file is in Section 1.6 of Appendix A.

The contour plot did not allow us to see the embedded bubbles. Therefore, VisIt's slicing operator was utilized on the combined volume and contour plots to see inside the outer bubbles, using a diagonal cut through the cube. These images can be found in Figures 4.3 and 4.6. The slice operator configuration XML file is in Section 1.7 of Appendix A.

In all cases VisIt's smoothing option was enabled.

### 4.2.1 One bell

The following initial conditions were used in the case of one bell.

$$\phi(\mathbf{x},0) = \phi_0(\mathbf{x}) \quad = \quad 2e^{-100|\mathbf{x}-(0.5,0.5,0.5)|^2} \tag{4.1}$$

$$\phi_t(\mathbf{x},0) = \phi_1(\mathbf{x}) \quad = \quad 0 \tag{4.2}$$

for $\mathbf{x} \in [0,1]^3 \subset \mathbb{R}^3$. The boundary conditions are $\phi(\mathbf{x},t) = \phi_0(\mathbf{x})$ for $\mathbf{x} \in \partial[0,1]^3$ and $t \geq 0$. The exponential function is never zero, strictly speaking, but numerically on the boundary of the unit box the value of the function is not larger than $2e^{-50} \approx 1.93 \times 10^{-22}$. This value, although very small, is still not zero in floating point arithmetic. Our simulation keeps the boundary values constant (Dirichlet boundary conditions). Although the boundary conditions were not zero, we did not notice any changes around the boundary over the time of the simulation. In fact, non-zero boundary values are extremely important for VisIt's contour plot to function correctly, so that the isosurfaces of zero values pertain to the sign changing regions only. The initial function could be considered to have compact support in the simulation since its value goes below single precision very fast.

In Figure 4.1, we present several time steps of the aforementioned initial conditions using only the volume plot. Notice that after the initial positive only values, we also observe negative values. The wave amplitude decreases with time. The wave spreads towards the boundaries.

In Figure 4.2, we present several time steps of the aforementioned initial conditions using the volume plot combined with an isosurface plot. The radius of the (green) bubble increases with time as the amplitude decreases as expected.

In Figure 4.3, we present several time steps of the aforementioned initial conditions using the volume plot combined with an isosurface plot, but with a slice down the diagonal of the unit box. The green circle denotes the bubble. At time $t = 0.17$ we can see the outline of a bubble begin to appear with the negative scalar values colored blue. We do not observe a second bubble as the wave continues in time. Note the white portion of the wave which appears due to the opacity transfer function configuration.

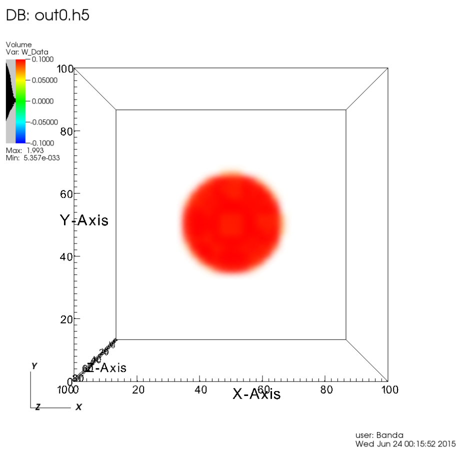
### 4.2.2 Two bells

The following initial conditions were used in the case of two bells.

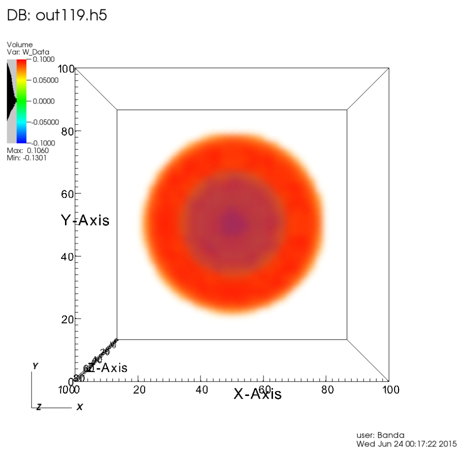
$$\phi(\mathbf{x},0) = \phi_0(\mathbf{x}) \quad = \quad 2\left[e^{-100|\mathbf{x}-(0.4,0.4,0.4)|^2} + e^{-100|\mathbf{x}-(0.6,0.6,0.6)|^2}\right] \tag{4.3}$$

$$\phi_t(\mathbf{x},0) = \phi_1(\mathbf{x}) \quad = \quad 0 \tag{4.4}$$

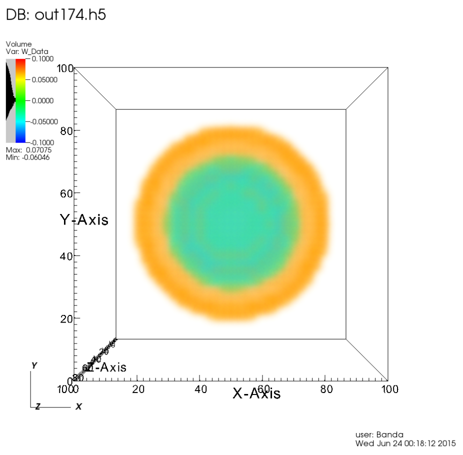
for $\mathbf{x} \in [0,1]^3 \subset \mathbb{R}^3$. The boundary conditions are $\phi(\mathbf{x},t) = \phi_0(\mathbf{x})$ for $\mathbf{x} \in \partial[0,1]^3$ and $t \geq 0$. The exponential function is never zero, strictly speaking, but numerically on the boundary of the unit box the value of the function is not larger than $2\left(e^{-60} + e^{-40}\right) \approx 8.5 \times 10^{-18}$. This value, although



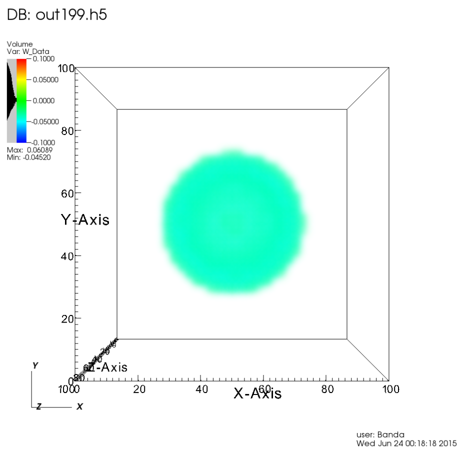
(a) $t = 0$



(b) $t = 0.595$

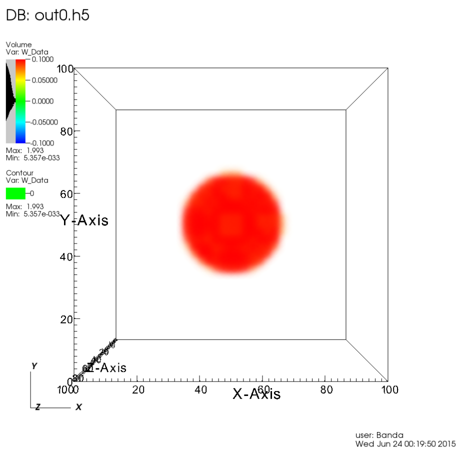


(c) $t = 0.87$

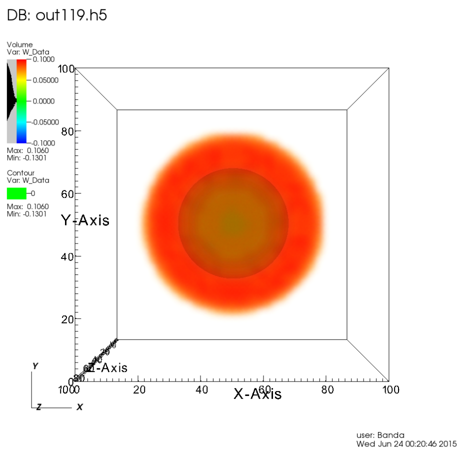


(d) $t = 0.995$

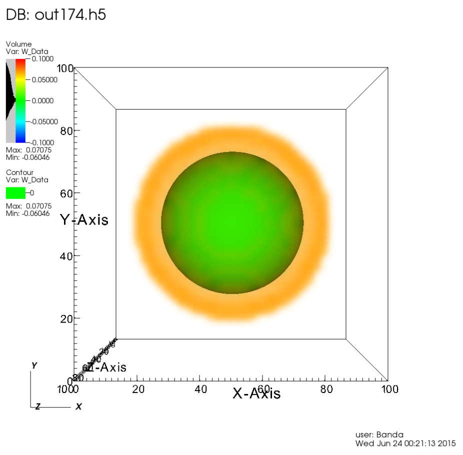
Figure 4.1: Three dimensional scalar rendering through time of one bell



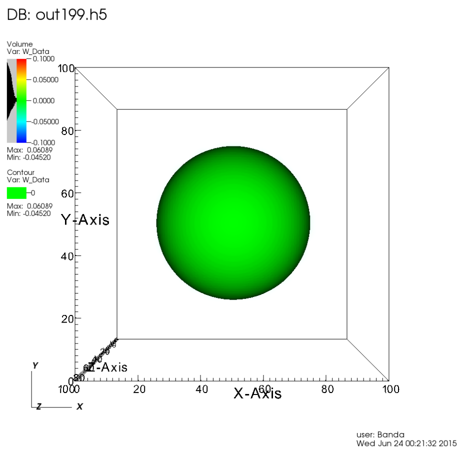
(a) $t = 0$



(b) $t = 0.595$

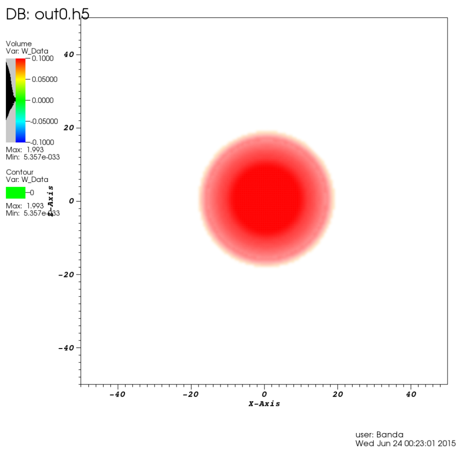


(c) $t = 0.87$

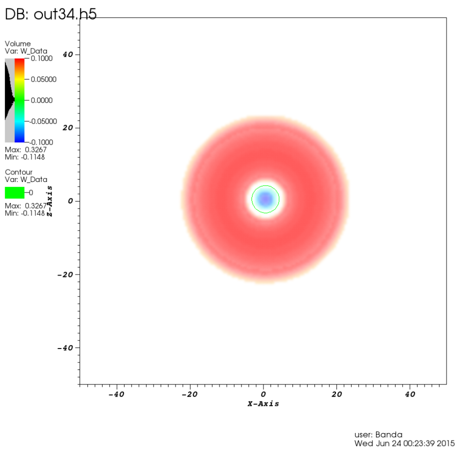


(d) $t = 0.995$

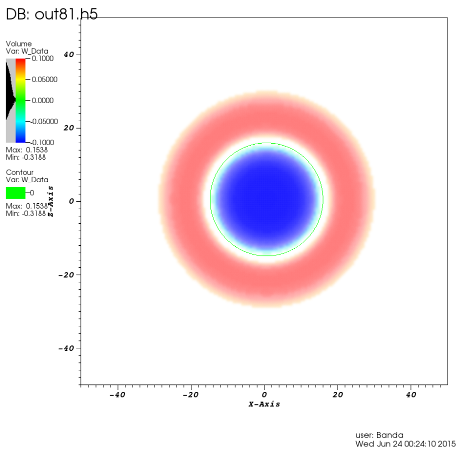
Figure 4.2: Three dimensional scalar rendering through time with isosurface



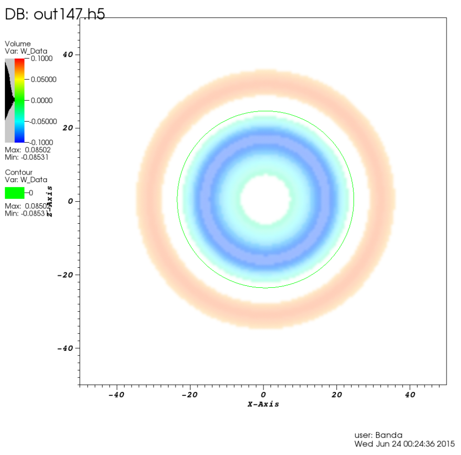
(a) $t = 0$



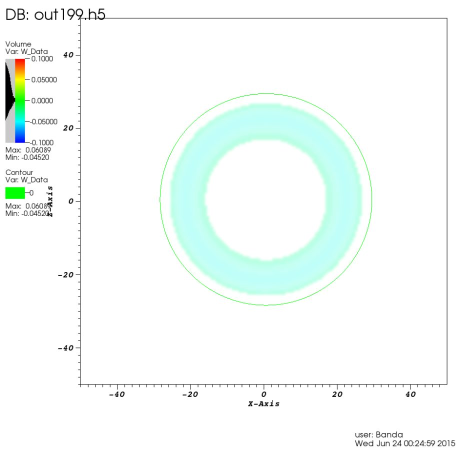
(b) $t = 0.17$



(c) $t = 0.405$



(d) $t = 0.735$



(e) $t = 0.995$

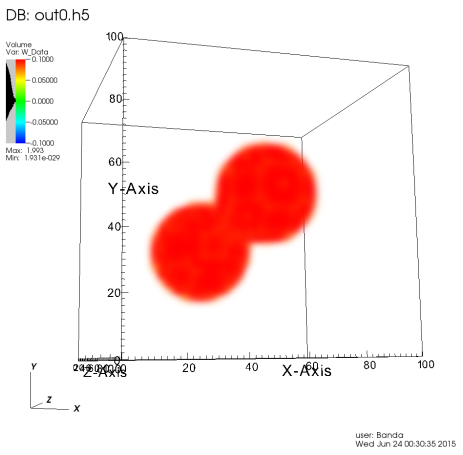
Figure 4.3: Cross section, or "slice", of three dimensional scalar rendering through time with isosurface

very small, is still not zero in floating point arithmetic. Our simulation keeps the boundary values constant (Dirichlet boundary conditions). Although the boundary conditions were not zero, we did not notice any changes around the boundary over the time of the simulation. The initial function could be considered to have compact support in the simulation since its value goes below single precision very fast.

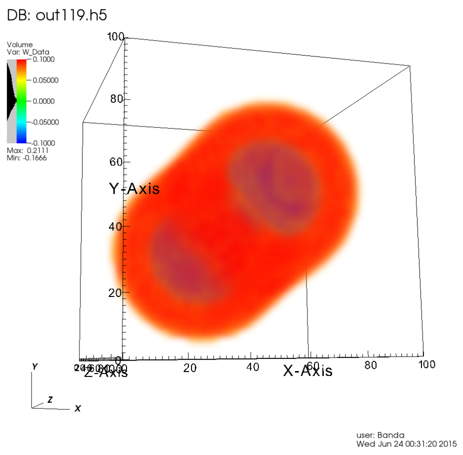
In Figure 4.4, we present several time steps of the aforementioned initial conditions using the volume plot. In addition to observing the wave propagate outward, we see an interaction taking place in which the two bell functions used in the initial conditions create a ring of high values, while three regions of negative values appear.

In Figure 4.5, we present several time steps of the aforementioned initial conditions using the volume plot combined with an isosurface plot. The bubbles are represented by the color green. At $t = 0.455$ in Figure 4.5b, we can start to see the bubbles forming inside their respective regions. At $t = 0.655$ in Figure 4.5c, we notice the bubbles reaching out towards each other, which leads us to $t = 0.84$ in Figure 4.5d. Here we see that the two bubbles have already split apart after coming together to create a new, third bubble in between them. Finally, in Figure 4.5e the three bubbles are clearly defined while a region of high values exists around the middle bubble. We also observe concavity forming on one side of the outer bubbles mirroring the interior bubble.

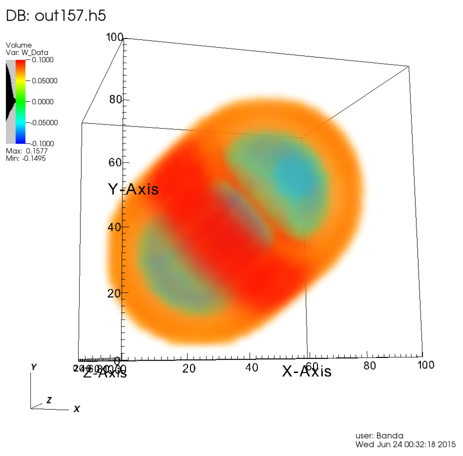
In Figure 4.6, we present several time steps of the aforementioned initial conditions using the volume plot combined with an isosurface plot, but with a slice down the diagonal of the unit box. The surface of the bubbles is represented by a green curve. In Figure 4.6b, we observe the bubbles expanding towards each other. In Figure 4.6c, it is evident that the bubbles are reaching out to each other as part of their interaction. In Figure 4.6d, the bubbles connect and introduce a new region of negative values in the middle region. In Figure 4.6e we notice increasing negative values in the middle, bringing the local minimum values to three. In addition, the regions between the local minimums have increasing values, which give way to holes inside the larger bubble in Figure 4.6f. We consider these holes as new bubbles. In Figure 4.6g we observe all the bubbles expanding. However, in Figure 4.6h we see that the faster growth of the inner bubbles leads to



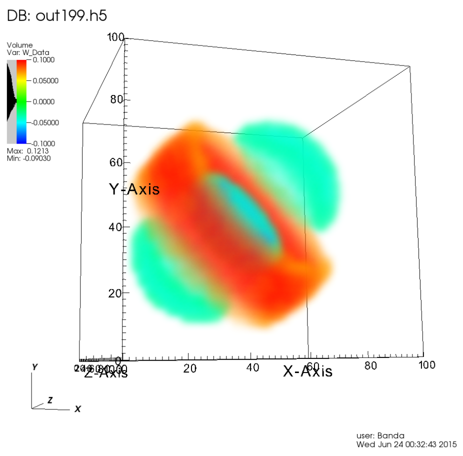
(a) $t = 0$



(b) $t = 0.595$

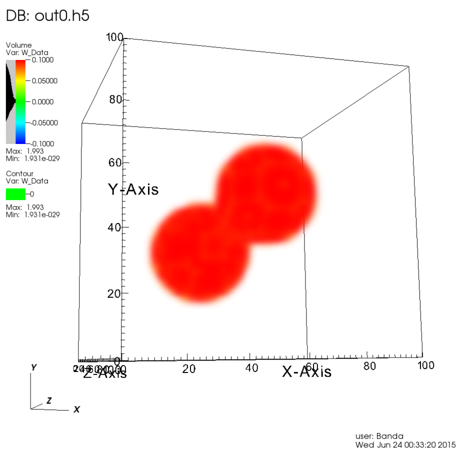


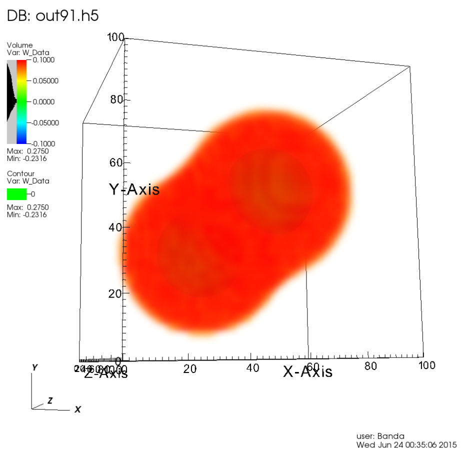
(c) $t = 0.785$

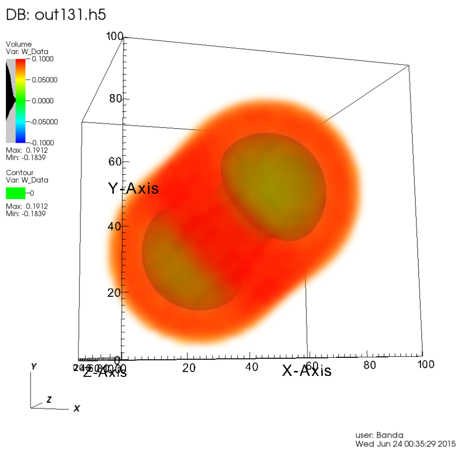


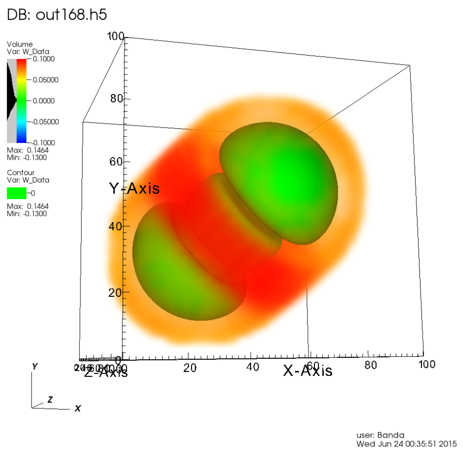
(d) $t = 0.995$

Figure 4.4: Three dimensional scalar rendering through time of two bells


(a) $t = 0$


(b) $t = 0.455$


(c) $t = 0.655$


(d) $t = 0.84$

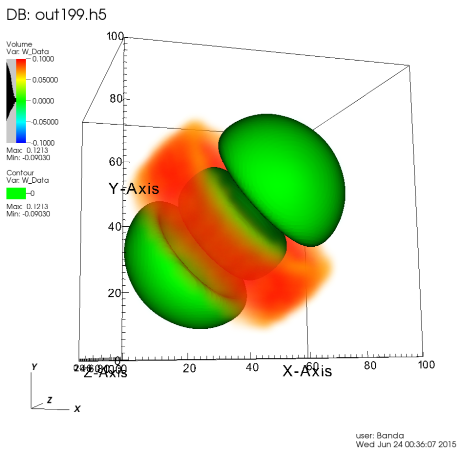
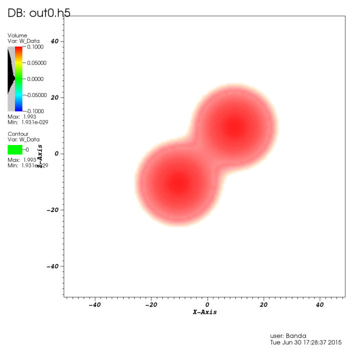
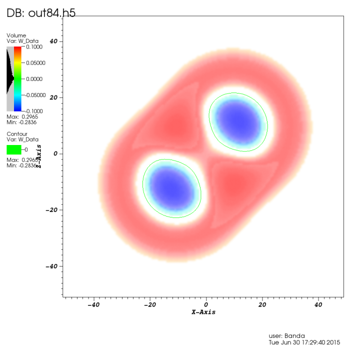

(e) $t = 0.995$

Figure 4.5: Three dimensional scalar rendering through time with isosurface

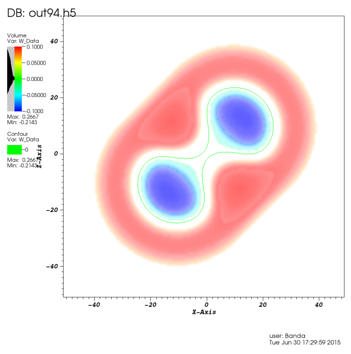
the creation of three, disjoint bubbles. Finally, in Figure 4.6i the bubbles separate further while maintaining their shape. We can also observe the aforementioned concavity forming on one side of the outer bubbles mirroring the interior bubble.



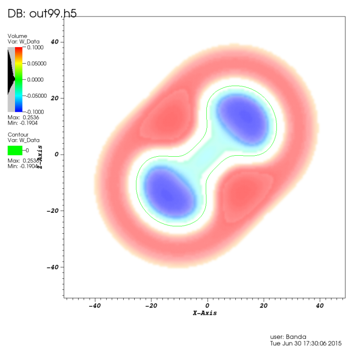
(a) $t = 0$



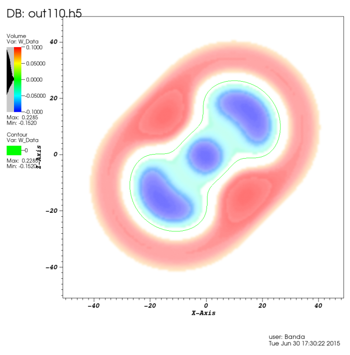
(b) $t = 0.42$



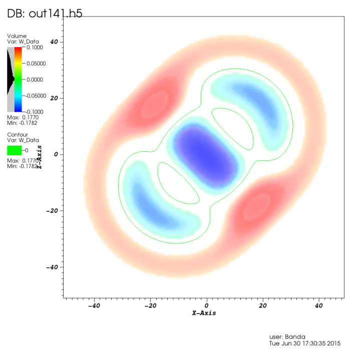
(c) $t = 0.47$



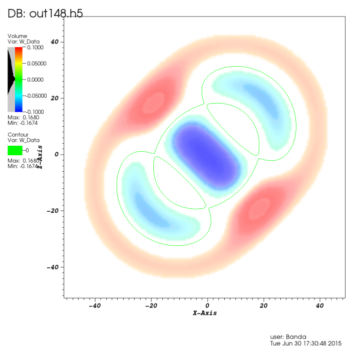
(d) $t = 0.495$



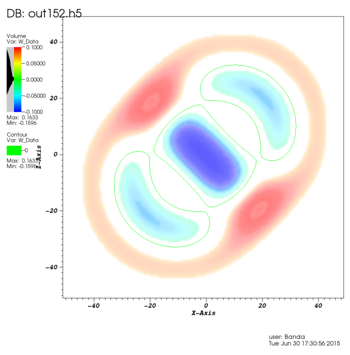
(e) $t = 0.55$



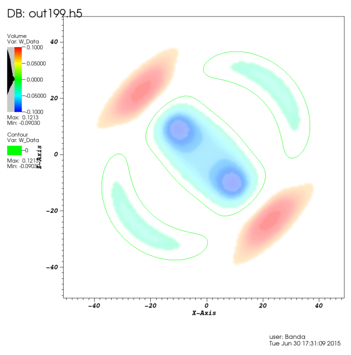
(f) $t = 0.705$



(g) $t = 0.74$



(h) $t = 0.76$



(i) $t = 0.995$

Figure 4.6: Cross section, or "slice", of three dimensional scalar rendering through time with isosurface

# CHAPTER V

# CONCLUSION

In this work we investigated how the bubbles of the Higgs boson equation in de Sitter spacetime are created and how they interact with each other. We were able to achieve this by utilizing high-power parallel computations and effective numerical simulations.

For future work we would like to continue examining the different possibilities that arise from modifying the parameters of the equation, namely $\mu$ and $\lambda$. Furthermore, it would be of interest to consider the interaction of the Higgs boson with other elementary particles to investigate the proposed effects of the Higgs field on quarks, leptons, or other hadrons. Specifically, the proposed effects of the Higgs field on the W and Z bosons would be of primary importance.

With regards to equation (1.1), we would also like to investigate whether the solution exists for all time or if it blows up at a finite time. This would imply that the solution becomes not smooth, or perhaps a derivative would become not smooth. The wave might blow up at a given finite time, say $t_0$ for example.

For advances in computing, the Department of Mathematics at the University of Texas-Pan American recently obtained a workstation with four top of the line NVIDIA Tesla K8 GPUs. We hope to obtain higher resolution using this new equipment. The current simulations are using single precision arithmetic. With the stronger GPUs, we expect to run the simulations with double precision arithmetic.

Finally, it would also be of interest to integrate the wave to consider the energy contained in the simulation, i.e. numerically calculating the energy. Perhaps it might grow without bound and blow up in finite time. Although we did not observe this scenario, it could happen for some very special initial conditions that we might look for in the future. For the focusing cubic nonlinear Klein-Gordon equation, such initial conditions corresponding to so-called negative energy were discussed by Donninger and Schlag [6].

We hope that this project will serve as the basis for many future high-performance computations.

# BIBLIOGRAPHY

[1] *The higgs boson*, Science, 338 (2012), pp. 1558–1559.

[2] K. BURRAGE, *Parallel and Sequential Methods for Ordinary Differential Equations*, Clarendon Press, New York, NY, USA, 1995.

[3] W. CHENEY AND D. KINCAID, *Numerical Mathematics and Computing*, International Thomson Publishing, 4th ed., 1998.

[4] H. CHILDS, E. BRUGGER, B. WHITLOCK, J. MEREDITH, S. AHERN, D. PUGMIRE, K. BIAGAS, M. MILLER, C. HARRISON, G. H. WEBER, H. KRISHNAN, T. FOGAL, A. SANDERSON, C. GARTH, E. W. BETHEL, D. CAMP, O. RÜBEL, M. DURANT, J. M. FAVRE, AND P. NAVRÁTIL, *VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data*, in High Performance Visualization–Enabling Extreme-Scale Scientific Insight, Oct 2012, pp. 357–372.

[5] J. DONGARRA AND A. L. LASTOVETSKY, *High Performance Heterogeneous Computing*, Wiley-Interscience, New York, NY, USA, 2009.

[6] R. DONNINGER AND W. SCHLAG, *Numerical study of the blowup/global existence dichotomy for the focusing cubic nonlinear klein-gordon equation*, Nonlinearity, 24 (2011), p. 2547.

[7] F. ENGLERT AND R. BROUT, *Broken symmetry and the mass of gauge vector mesons*, Phys. Rev. Lett., 13 (1964), pp. 321–323.

[8] S. L. GLASHOW AND M. GELL-MANN, *Gauge theories of vector particles*, Annals of Physics, 15 (1961), pp. 437 – 460.

[9] K. HAGIWARA, K. HIKASA, K. NAKAMURA, M. TANABASHI, ET AL., *Review of particle physics*, 66 (2002).

[10] P. W. HIGGS, *Broken symmetries and the masses of gauge bosons*, Phys. Rev. Lett., 13 (1964), pp. 508–509.

[11] ——, *Spontaneous symmetry breakdown without massless bosons*, Phys. Rev., 145 (1966), pp. 1156–1163.

[12] N. KAW, KALU, *Numerical Methods with Applications*, http://www.autarkaw.com, 2nd ed., 2011.

[13] M. KORCH AND T. RAUBER, *Parallel implementation of runge-kutta integrators with low storage requirements*, in Euro-Par 2009 Parallel Processing, H. Sips, D. Epema, and H.-X. Lin, eds., vol. 5704 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 785–796.

[14] A. LASTOVETSKY, *Adaptive parallel computing on heterogeneous networks with mpc*, Parallel Computing, 28 (2002), pp. 1369 – 1407.

[15] V. W. LEE, C. KIM, J. CHHUGANI, M. DEISHER, D. KIM, A. D. NGUYEN, N. SATISH, M. SMELYANSKIY, S. CHENNUPATY, P. HAMMARLUND, R. SINGHAL, AND P. DUBEY, *Debunking the 100x gpu vs. cpu myth: An evaluation of throughput computing on cpu and gpu*, SIGARCH Comput. Archit. News, 38 (2010), pp. 451–460.

[16] D. S. MEDINA, A. ST.-CYR, AND T. WARBURTON, *OCCA: A unified approach to multi-threading languages*, CoRR, abs/1403.0968 (2014).

[17] J. SCHWINGER, *Dynamical theory of k mesons*, Phys. Rev., 104 (1956), pp. 1164–1172.

[18] J. E. STONE, D. GOHARA, AND G. SHI, *Opencl: A parallel programming standard for heterogeneous computing systems*, IEEE Des. Test, 12 (2010), pp. 66–73.

[19] THE ATLAS COLLABORATION, *A particle consistent with the higgs boson observed with the atlas detector at the large hadron collider*, Science, 338 (2012), pp. 1576–1582.

[20] THE HDF GROUP, *Hierarchical Data Format, version 5*, 1997-2015. http://www.hdfgroup.org/HDF5/.

[21] THE ROYAL SOCIETY, *Understanding the higgs boson*. Online, 6 Carlton House Terrace, London SW1Y 5AG, 10 2013. http://www.understanding-the-higgs-boson.org/.

[22] T. WARBURTON, *OCCA2: extensible many-core programming*. Online, 9 2014. https://github.com/tcew/OCCA2.

[23] K. YAGDJIAN, *On the global solutions of the higgs boson equation*, Communications in Partial Differential Equations, 37 (2012), pp. 447–478.

[24] L. ZENG, C. SMITH, G. H. POELZER, J. RODRIGUEZ, E. CORPUZ, AND G. YANEV, *Illustrations and supporting texts for sound standing waves of air columns in pipes in introductory physics textbooks*, Phys. Rev. ST Phys. Educ. Res., 10 (2014), p. 020110.

## APPENDIX A

# APPENDIX A

# OCCA2 IMPLEMENTATION

## 1.1 Kernel code

```
%\begin{lstlisting}[language=C++,frame=single]
occaKernel void fdKernel(const int N, const float rt, const float dt, const float
    mu_2, const float lambda, float *w_i, float *w_im1, float *w_ip1)
{
   for(int groupZ = 0; groupZ < N ; groupZ += 8; outer2)
   {   // <-- parallelized by threads (GPU: blocks)
      for(int groupY = 0; groupY < N ; groupY += 8; outer1)
      {   // <-- parallelized by threads (GPU: blocks)
         for(int groupX = 0; groupX < (N + 2); groupX += 8; outer0)
         {  //
            for(int l = groupZ; l < (groupZ + 8); ++l; inner2)
            {       // <-- serial for each thread (GPU: threads)
               for(int k = groupY; k < (groupY + 8); ++k; inner1)
               {       // <-- serial for each thread (GPU: threads)
                  for(int j = groupX; j < (groupX + 8); ++j; inner0)
                  {     //
                     if((0 < l) && (l < N - 1) && (0 < k) && (k < N - 1) && (0 < j)
                        && (j < N - 1))
                     {
                        w_ip1[jkl] = rt * (w_i[jkl + 1] - 6*w_i[jkl] + w_i[jkl - 1]
                            + w_i[jkl + N] + w_i[jkl - N] + w_i[jkl + N*N] + w_i[jkl
                            - N*N]) + ((2.0*dt*dt)/(2.0+3.0*dt))*(mu_2*w_i[jkl] -
                            lambda*w_i[jkl]*w_i[jkl]*w_i[jkl]) +
                            (2.0/(2.0+3.0*dt))*(2.0*w_i[jkl] +
                            (1.5*dt-1)*w_im1[jkl]);
                      }
                    }
                  }
                }
              }
            }
  }
}
```

## 1.2 C++ file

```
%\begin{lstlisting}[language=C++,frame=single]
#include <iostream>
#include <sstream>
#include <string>
#include <cmath>
#include "occa.hpp"
#include "H5Cpp.h"

occa::device device;    // creating devices
occa::kernel fdKernel;  // kernel name
occa::memory w_i, w_im1, w_ip1; // variables
```

```
float dx, dt, rt, r, dt_2, mu_2, lambda, L;
int N;


#ifndef H5_NO_NAMESPACE
using namespace H5;
#endif

const H5std_string DATASET_NAME( "W_Data" );
using namespace std;

void setupGrid();
void setupCL();
void solve();
int writeHDF(int iter);

int main(int argc, char **argv)
{
   setupGrid();
   setupCL();
   solve();
   return 0;
}

void setupGrid()
{
   N  = 100;
   L  = 2.0;
   dx = 1.0/N;
```

```
  dt = 0.005;

  dt_2 = dt*dt;

  mu_2 = 0.1;

  lambda = 0.1;

}


void setupCL()

{

  int platformID = 0;

  int deviceID  = 0;


  device.setup("mode = OpenCL, platformID=0, deviceID=0"); //locating
     device/GPU***REVIEW

  // device.setup("OpenCL", platformID, deviceID); //locating device/GPU***REVIEW

  // device.setup("OpenMP");     // For OpenMP

  // device.setup("CUDA", deviceID); // For CUDA


  fdKernel = device.buildKernelFromSource("fdKernel.okl", "fdKernel"); //generate
     the openCL func from kernel


  w_i = device.malloc(N*N*N*sizeof(float));

  w_im1 = device.malloc(N*N*N*sizeof(float));

  w_ip1 = device.malloc(N*N*N*sizeof(float));

}


void solve()

{

  vector<float> initVector(N*N*N, 0.0);

  //vector<double> zeroVector(N*N*N, 0.0);
```

```
    int j, k, l;

    //exponential
    for (l = 0; l < N; l++)
      for (k = 0; k < N; k++)
        for (j = 0; j < N; j++)
          initVector[l*N*N+k*N+j] =
             exp(-100.0*(pow(l*dx-0.4,2.0)+pow(k*dx-0.4,2.0)+pow(j*dx-0.4,2.0)))+
             exp(-100.0*(pow(l*dx-0.6,2.0)+pow(k*dx-0.6,2.0)+pow(j*dx-0.6,2.0)));

    w_i.copyFrom(&(initVector[0])); // copy vectors to gpu
    w_im1.copyFrom(&(initVector[0]));
    w_ip1.copyFrom(&(initVector[0]));

    //time steps
    for(int iter = 0; iter < 200; iter++)
    {     //Reduced Iterations
      rt=2.0*exp(-2.0*iter*dt)*dt*dt/(dx*dx*(2.0+3.0*dt)*L*L);
      fdKernel(N, rt, dt, mu_2, lambda, w_i, w_im1, w_ip1);

      if (iter % 1 ==0)
        writeHDF(iter/1);

      w_im1.swap(w_i);
      w_i.swap(w_ip1);
    }
    device.finish();
}
```

```
int writeHDF(int iter)
{
  vector<float> soln(N*N*N);
  w_ip1.copyTo(&(soln[0]));
  std::stringstream ss;
  ss << "out" << iter << ".h5";
  std::string filename=ss.str();
  H5std_string FILE_NAME( filename );

  /* Data initialization. */
  const int  RANK = 3;
  int j, k, l;
  float data[N][N][N];       // buffer for data to write

  for (l = 0; l < N; l++)
    for (k = 0; k < N; k++)
    {
      for (j = 0; j < N; j++) data[l][k][j] = soln[l*N*N+k*N+j];
    }

  // Try block to detect exceptions raised by any of the calls inside it
  try
  {
    /* Turn off the auto-printing when failure occurs so that we can handle the
        errors appropriately */
    Exception::dontPrint();

    /* Create a new file using H5F_ACC_TRUNC access, default file creation
        properties, and default file access properties.  */
```

```
   H5File file( FILE_NAME, H5F_ACC_TRUNC );

   /* Define the size of the array and create the data space for fixed size
      dataset. */
   hsize_t    dimsf[3];            // dataset dimensions

   dimsf[0] = N;
   dimsf[1] = N;
   dimsf[2] = N;
   DataSpace dataspace( RANK, dimsf );
   /* Define datatype for the data in the file. We will store little endian INT
      numbers. */
   //     IntType datatype( PredType::NATIVE_INT );
   FloatType datatype( PredType::NATIVE_FLOAT );
   //     datatype.setOrder( H5T_ORDER_LE );
   /* Create a new dataset within the file using defined dataspace and datatype
      and default dataset creation properties. */
   DataSet dataset = file.createDataSet( DATASET_NAME, datatype, dataspace );

   /* Write the data to the dataset using default memory space, file space, and
      transfer properties. */
   dataset.write( data, PredType::NATIVE_FLOAT );
}  // end of try block

// catch failure caused by the H5File operations
catch( FileIException error )
{
   error.printError();
   return -1;
   }

   // catch failure caused by the DataSet operations
   catch( DataSetIException error )
   {
      error.printError();
      return -1;
   }

   // catch failure caused by the DataSpace operations
   catch( DataSpaceIException error )
   {
      error.printError();
      return -1;
   }

   // catch failure caused by the DataSpace operations
   catch( DataTypeIException error )
   {
      error.printError();
      return -1;
   }
   return 0; // successfully terminated
}
```

## 1.3 Makefile

```
%\begin{lstlisting}[language=make,frame=single]
ifndef OCCA_DIR
```

```
   OCCA_DIR = ../../
endif

include ${OCCA_DIR}/scripts/makefile

#---[ COMPILATION ]-------------------------------
headers = $(wildcard $(iPath)/*.hpp) $(wildcard $(iPath)/*.tpp)
sources = $(wildcard $(sPath)/*.cpp)

objects = $(subst $(sPath)/,$(oPath)/,$(sources:.cpp=.o))

examples = main

links += -lhdf5_cpp -lhdf5

executables: $(examples)

main: $(objects) $(headers) gpu-wave.cpp
   $(compiler) $(compilerFlags) -o main $(flags) $(objects) gpu-wave.cpp $(paths)
       $(links)

$(oPath)/%.o:$(sPath)/%.cpp $(wildcard $(subst
    $(sPath)/,$(iPath)/,$(<:.cpp=.hpp))) $(wildcard $(subst
    $(sPath)/,$(iPath)/,$(<:.cpp=.tpp)))
$(compiler) $(compilerFlags) -o $@ $(flags) -c $(paths) $<

clean:
   rm -f $(oPath)/*;
   rm -f main;
   rm -f *h5;
#==================================================
```

## 1.4 OCCA environment variables

```
%\begin{lstlisting}[language=bash,frame=single]
export PATH=$PATH:/usr/local/visit/bin:/usr/local/visit/2.8.0/linux-x86_64/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH::/usr/local/OCCA2/lib

export OCCA_DIR=/usr/local/OCCA2
export OCCA_PTHREADS_ENABLED=1
export OCCA_OPENMP_ENABLED=1
export OCCA_OPENCL_ENABLED=1
export OCCA_CUDA_ENABLED=1
```

## 1.5 VisIt volume settings

```
%\begin{lstlisting}[language=XML,frame=single]
<?xml version="1.0"?>
<Object name="VolumeAttributes">
    <Field name="legendFlag" type="bool">true</Field>
    <Field name="lightingFlag" type="bool">false</Field>
    <Object name="colorControlPoints">
        <Object name="ColorControlPointList">
            <Object name="ColorControlPoint">
                <Field name="colors" type="unsignedCharArray" length="4">0 0 255
                    255 </Field>
                <Field name="position" type="float">0</Field>
            </Object>
            <Object name="ColorControlPoint">
            <Field name="colors" type="unsignedCharArray" length="4">0 255 255
               255 </Field>
            <Field name="position" type="float">0.25</Field>
         </Object>
         <Object name="ColorControlPoint">
            <Field name="colors" type="unsignedCharArray" length="4">0 255 0
               255 </Field>
            <Field name="position" type="float">0.5</Field>
         </Object>
         <Object name="ColorControlPoint">
            <Field name="colors" type="unsignedCharArray" length="4">255 255 0
               255 </Field>
            <Field name="position" type="float">0.75</Field>
         </Object>
         <Object name="ColorControlPoint">
            <Field name="colors" type="unsignedCharArray" length="4">255 0 0
               255 </Field>
            <Field name="position" type="float">1</Field>
         </Object>
         <Field name="smoothing" type="string">Linear</Field>
         <Field name="equal" type="bool">false</Field>
         <Field name="discrete" type="bool">false</Field>
         <Field name="category" type="string"></Field>
      </Object>
   </Object>
   <Field name="opacityAttenuation" type="float">1</Field>
   <Field name="opacityMode" type="string">FreeformMode</Field>
   <Object name="opacityControlPoints">
      <Object name="GaussianControlPointList">
```

```
    </Object>
</Object>
<Field name="resampleFlag" type="bool">true</Field>
<Field name="resampleTarget" type="int">60000</Field>
<Field name="opacityVariable" type="string">W_Data</Field>
<Field name="compactVariable" type="string">default</Field>
<Field name="freeformOpacity" type="unsignedCharArray" length="256">255 255
    255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
    255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
    255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
    255 255 255 255 255 255 255 255 255 255 255 251 248 244 244 238 234 231
    227 224 221 214 207 204 200 197 193 190 190 187 180 176 176 173 169 166
    163 163 159 159 149 142 139 135 132 125 118 112 108 105 101 95 91 84 81 78
    71 64 57 54 50 74 61 40 33 30 27 33 0 0 0 0 0 0 0 0 0 0 0 0 27 30 33 37 40
    44 54 61 64 67 71 57 61 64 64 67 71 71 74 78 78 78 81 81 84 88 88 91 91 91
    91 95 98 98 101 105 105 105 108 108 108 112 112 112 112 112 112 115 118
    118 122 125 125 125 125 125 125 125 125 125 129 129 132 135 139 139 139
    139 139 142 146 149 149 153 156 159 163 163 163 166 166 169 173 176 176
    180 183 183 187 190 193 197 204 204 207 207 210 214 217 221 224 224 227
    231 234 234 238 241 248 251 255 255 255 255 255 255 255 255 255 255
    </Field>
<Field name="useColorVarMin" type="bool">true</Field>
<Field name="colorVarMin" type="float">-0.1</Field>
<Field name="useColorVarMax" type="bool">true</Field>
<Field name="colorVarMax" type="float">0.1</Field>
<Field name="useOpacityVarMin" type="bool">true</Field>
<Field name="opacityVarMin" type="float">-1</Field>
<Field name="useOpacityVarMax" type="bool">true</Field>
<Field name="opacityVarMax" type="float">1</Field>
```

```
    <Field name="smoothData" type="bool">true</Field>
    <Field name="samplesPerRay" type="int">500</Field>
    <Field name="rendererType" type="string">Splatting</Field>
    <Field name="gradientType" type="string">SobelOperator</Field>
    <Field name="num3DSlices" type="int">200</Field>
    <Field name="scaling" type="string">Linear</Field>
    <Field name="skewFactor" type="double">1</Field>
    <Field name="limitsMode" type="string">OriginalData</Field>
    <Field name="sampling" type="string">Rasterization</Field>
    <Field name="rendererSamples" type="float">3</Field>
    <Field name="transferFunctionDim" type="int">1</Field>
    <Field name="lowGradientLightingReduction" type="string">Lower</Field>
    <Field name="lowGradientLightingClampFlag" type="bool">false</Field>
    <Field name="lowGradientLightingClampValue" type="double">1</Field>
    <Field name="materialProperties" type="doubleArray" length="4">0.4 0.75 0 15
        </Field>
</Object>
```

## 1.6 VisIt contour settings

```
%\begin{lstlisting}[language=XML,frame=single]
<?xml version="1.0"?>
<Object name="ContourAttributes">
<Object name="defaultPalette">
<Object name="ColorControlPointList">
<Field name="compactColors" type="unsignedCharVector">255 0 0 255 0 255 0 255 0 0
    255 255 0 255 255 255 255 0 255 255 255 255 0 255 255 135 0 255 255 0 135 255
    168 168 168 255 255 68 68 255 99 255 99 255 99 99 255 255 40 165 165 255 255
    99 255 255 255 255 99 255 255 170 99 255 170 79 255 255 150 0 0 255 0 150 0
    255 0 0 150 255 0 109 109 255 150 0 150 255 150 150 0 255 150 84 0 255 160 0
    79 255 255 104 28 255 0 170 81 255 68 255 124 255 0 130 255 255 130 0 255 255
    </Field>
<Field name="compactPositions" type="floatVector">0 0.034 0.069 0.103 0.138 0.172
    0.207 0.241 0.276 0.31 0.345 0.379 0.414 0.448 0.483 0.517 0.552 0.586 0.621
    0.655 0.69 0.724 0.759 0.793 0.828 0.862 0.897 0.931 0.966 1 </Field>
<Field name="smoothing" type="string">None</Field>
<Field name="equal" type="bool">true</Field>
<Field name="discrete" type="bool">true</Field>
<Field name="category" type="string">Standard</Field>
</Object>
</Object>
<Field name="colorType" type="string">ColorBySingleColor</Field>
<Field name="colorTableName" type="string">Default</Field>
<Field name="legendFlag" type="bool">true</Field>
<Field name="lineStyle" type="int">0</Field>
<Field name="lineWidth" type="int">0</Field>
<Object name="singleColor">
<Object name="ColorAttribute">
<Field name="color" type="unsignedCharArray" length="4">255 0 0 255 </Field>
</Object>
</Object>
<Field name="contourNLevels" type="int">10</Field>
<Field name="contourValue" type="doubleVector">0 </Field>
<Field name="contourPercent" type="doubleVector"></Field>
<Field name="contourMethod" type="string">Value</Field>
<Field name="minFlag" type="bool">false</Field>
<Field name="maxFlag" type="bool">false</Field>
<Field name="min" type="double">0</Field>
```

```
<Field name="max" type="double">1</Field>
<Field name="scaling" type="string">Linear</Field>
<Field name="wireframe" type="bool">false</Field>
</Object>
```

## 1.7 VisIt slice settings

```
%\begin{lstlisting}[language=XML,frame=single]
<?xml version="1.0"?>
<Object name="SliceAttributes">
    <Field name="originType" type="string">Point</Field>
    <Field name="originPoint" type="doubleArray" length="3">51 51 51 </Field>
    <Field name="originIntercept" type="double">0</Field>
    <Field name="originPercent" type="double">0</Field>
    <Field name="originZone" type="int">0</Field>
    <Field name="originNode" type="int">0</Field>
    <Field name="normal" type="doubleArray" length="3">0 -1 0 </Field>
    <Field name="axisType" type="string">Arbitrary</Field>
    <Field name="upAxis" type="doubleArray" length="3">0 0 1 </Field>
    <Field name="project2d" type="bool">true</Field>
    <Field name="interactive" type="bool">true</Field>
    <Field name="flip" type="bool">false</Field>
    <Field name="originZoneDomain" type="int">0</Field>
    <Field name="originNodeDomain" type="int">0</Field>
    <Field name="meshName" type="string">mesh_100x100x100</Field>
    <Field name="theta" type="double">0</Field>
    <Field name="phi" type="double">0</Field>
</Object>
```

## BIOGRAPHICAL SKETCH

Jacob Nathaniel Banda was born and raised in the Rio Grande Valley of Texas in 1988. He graduated from Weslaco High School in 2006, received his Bachelor's of Science in Electrical Engineering from the University of Texas-Pan American in May of 2010, and his Master's of Science in Applied Mathematics from the same university in August of 2015. During his graduate study, he served as a GAANN Fellow and continued to reach out to the community as the first President and founder of the nationally recognized SPIE Student Chapter at UTPA, now UTRGV. He specializes in programming and numerical methods, with interests in particle physics, amplifier design, speaker dynamics, and live music production/recording. He may be contacted by email at banda_jn@hotmail.com or by physical correspondence at 715 Silva Dr, Weslaco, TX 78596.