Applying Cluster Computing With Raspberry Pis to Numerical General Relativity

Zack Chester, Shanshan Rodriguez, and Leo Rodriguez

Department of Physics, Worcester Polytechnic Institute, Worcester, MA 01609

April 25, 2018

Abstract

Cluster computing is an emerging field with an immense amount of applications in parallel processing and low-budget computer builds with a need for large amounts of processing power. Problems in numerical relativity often have complex computations and solutions that lend themselves to being very easily put into a parallel format. The practicality of combining the two is explored in this paper. Specifically, this paper discusses the benefits that cluster computing provides when assessing potential formulation of Kerr black-holes. Also, it asses potential applications in simulating gravitational waves. Both applications serve to demonstrate the benefits that cluster computing may offer to numerical relativity for more specific/advanced use.

Special Thanks

I would like to thank both Professor Shanshan Rodriguez and Professor Leo Rodriguez for all that they taught me and all of the help that they gave me with this project. I would also like to thank Dominic Chang for his help in teaching me general relativity.

Introduction

Numerical relativity has recently gained the attention of the public eye with the research that has been conducted on black holes and gravitational waves. Despite all the hype that the field has recently seen, a lot of the concepts that are being discussed are neither new or unexpected. In fact, the concept of gravitational waves goes back to some of the famous predictions made by Albert Einstein.

Before delving into the specific uses of numerical relativity for this project, it is necessary to introduce some details that are different and not often discussed in classical mechanics. First, it is necessary to consider gravity as a geometry, this is actually the main difference between general relativity and classical mechanics. Other than that, for the purposes of this project, the only real difference is that some constants will be renamed in a general relativity basis, an issue that will be discussed later.

A good way to conceptualize gravity as a geometry is to imagine a piece of fabric stretched as taut as possible, then add marbles and balls of various sizes and masses. The objects that pull the dips in the fabric would indicate the gravitational geometry of the object. Obviously, a marble that is smaller and more massive than a wiffle ball will create a much larger dip in the fabric. In this analogy, the Sun that the Earth revolves around would be the wiffle ball, something that is massive in its' own right, but when compared to the marble, which would represent a black hole, it has a relatively small dip.

In order to represent the geometries of any surface studied, a metric is used. For example, the metric for a flat plane, a very common representation in classical mechanics, is defined as:

$$ds^2 = dx^2 + dy^2 = (dx, dy) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} \tag{1}$$

In general relativity, there are also some unit changes that occur in general relativity. Specifically, the speed of light, c, is treated as having a value of 1, as opposed to the classical value of 299,800,000 meters per second squared.

These metrics vary in complexity depending on the application. The simplest metric used in this paper is the metric for flat space, the Minkowski metric. It is described as:

$$\eta = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

There are key things to note about metrics in general relativity. First, in classical mechanics, students are often taught to think in three spatial dimensions, x_1, x_2, and x_3. In general relativity, it is necessary to include a time, x_0, term.

For metrics, the terms are often defined as (x_0, x_1, x_2, x_3), where the terms represent (t, r, θ, ϕ), respectively.

More advanced metrics can be used to describe black holes in varying complexity. Each of these metrics should be able to reduce to the Minkowski metric. The Schwarzschild metric describes an uncharged, non-rotating, black hole and is the simplest expression for a black hole. It is defined as:

$$ds^2 = -(1 - \frac{2GM}{r})dt^2 + (1 - \frac{2GM}{r})^{-1}dr^2 + r^2 d\theta^2 + r^2 sin^2(\theta)d\phi^2 \tag{3}$$

with coordinates (t, r, θ, ϕ). Some key features are that as the mass approaches zero and as the radius approaches infinity, the Schwarzschild metric reduces to the Minkowski metric. There are other metrics that are used to describe black holes in varying complexity, but the main metric used in this paper is the Kerr metric.

The Kerr Metric

The Kerr metric is slightly more advanced than the Schwarzschild metric, describing uncharged rotating black holes. The initial goal of this project was to create a solution to the Kerr metric described as:

$$ds^2 = \Omega^2[-f(r)dt^2 + \frac{1}{f(r)}dr^2 + e^{-2\psi}d\theta^2 + e^{-2\psi}sin^2(\theta)(d\phi^2 - A^2(r)dt^2)] \tag{4}$$

with coordinates (t, r, θ, ϕ). In this equation, Ω is treated as being an arbirtary constant and ψ is the main solution that is attempting to be attained. A(r) is defined as:

$$A(r) = \frac{a}{r^2 + a^2} \tag{5}$$

and f(r) being:

$$f(r) = \frac{\Delta}{r^2 + a^2} \tag{6}$$

where:

$$\Delta = r^2 + a^2 - 2GMr \tag{7}$$

In order to attempt to create a solution for this metric, it is necessary to use the Einstein field equations. These are a set of sixteen coupled partial differential that are able to describe the gravitational effects on a given mass. These equations are defined as:

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} + \Lambda g_{\mu\nu} = 8\pi G T_{\mu\nu} \tag{8}$$

Where $R_{\mu\nu}$ is the Ricci tensor, R is the scalar curvature, $g_{\mu\nu}$ is the metric tensor, and $T_{\mu\nu}$ is the stress energy tensor. Also included in the equations is the cosmological constant, Λ, whose term is often canceled out or ignored.

There are multiple ways that this may be achieved, the specific options explored were to attempt to create a solution by hand or use computing software like Mathematica or Matlab. Solving by hand give the benefit of potentially no round-off error on any numerical solution, however, the computations necessary for the project are very complex and would likely lead to other errors in computation and deriving the solution would take an exorbitant amount of time. Matlab doe present the opportunity for time savings and has the ability to generate an accurate solution, but Mathematica was ultimately chosen. Mathematica is commonly used in numerical and general relativity research, it is easy to setup into a parallel setup, and it easily able to evaluate symbolic scripts.

Specifically for this project a cluster computer of ten Raspberry Pi 3 Model B's, the specifics of the setup are described in Appendix A. The main goal of this project was to show the viability of applying cluster computing to numerical relativity. In order to demonstrate this, the scripts evaluated in this project were run on this cluster system.

In order to solve the Einstein field equations, the Christoffel symbols, a set of coefficients that descirbe a change in basis as an object moves from a point to a differential displacement along a curve where other coordinates are constant [1], must be defined. These Christoffel symbols are represented as:

$$\Gamma^a_{bd} = \frac{1}{2}g^{ae}[\partial_b g_{de} + \partial_d g_{eb} - \partial_e g_{bd}] \tag{9}$$

It is also necessary to define the Reimann tensor, which describes the curvature of a manifold, and should reduce to zero in flat space-time [1]. It is specifically defined as:

$$R^a_{bcd} = \partial_c \Gamma^a_{bd} - \partial d \Gamma^a_{bc} + \Gamma^a_{ce}\Gamma^e_{bd} - \Gamma^a_{de}\Gamma^e_{bc} \tag{10}$$

which reduces to the Ricci tensor as the first and third indices are retracted:

$$R_{bd} = R^a_{bcd} \tag{11}$$

The Ricci scalar becomes the scalar curvature when multiplied by the metric:

$$R = g^{bd} R_{bd} \tag{12}$$

When substituted for the Kerr values noted previously, the Ricci tensor is shown in figure 1, with a scalar curvature of:

$$\frac{2e^{-2\mathrm{Psi}}\left(a^2 b^4 \sin^2(\theta) + e^{2\mathrm{Psi}}\left(a^2+b^2\right)\left(a^6 e^{2\mathrm{Psi}} + 3a^4 b^2 e^{2\mathrm{Psi}} + 3a^2\left(b^4 e^{2\mathrm{Psi}} - \Delta\right) + b^2\left(b^4 e^{2\mathrm{Psi}} + \Delta\right)\right)\right)}{\mathrm{Omega}^2\left(a^2+b^2\right)^4} \tag{13}$$

Ultimately generating the stress-energy tensor depicted in figure 2, where r is replaced with a and a is replaced with b.

Gravitational Waves

This was not the only goal of this project, another objective was to attempt to show demonstrate the viability of using cluster computing to simulate gravitational waves, which has become a rather prolific field since the discoveries made by LIGO and Virgo.

There are some considerations to make for this portion of the project that were not previously specified. Specifically, these calculations rely on post-Newtonian theory, a general relativity theory that applies to weak gravitational fields and slow moving matter. This theory provides a foundation for calculating gravitational waves emitted from a compact binary system and can even show orbital evolution under radiative losses. As a matter of convention:

$$(\frac{v}{c})^n = \frac{n}{2} PN \tag{14}$$

where v is the object velocity, c is the speed of light, and PN represents some post-Newtonian order. The formulation for describing gravitational waves as they propagate through a detector rely on depend on some transverse and traceless perturbation from the flat space metric, defined as:

$$h_{\mu\nu}^{TT} = (g_{\mu\nu} - \eta_{\mu\nu})^{TT} \tag{15}$$

as before, η describes the flat space metric.

h is composed of two different states, h_+ and h_x. These are defined by:

$$h_+ = \frac{1}{2}(x_0 x_1 - y_0 y_1) * h_{\mu\nu}^{TT} \tag{16}$$

$$h_x = \frac{1}{2}(x_0 y_1 - x_1 y_0) * h_{\mu\nu}^{TT} \tag{17}$$

with x and y representing two vectors. Several sources have already created expansions for gravitational waves out to the order of 2PN [2], the goals of this research were to apply this formulation in Mathematica in order to create a simulation of gravitational waves similar to those made by Stein and Mohapatra [3, 4]. As is defined by Blanchet:

$$h_{+,x} = \frac{2Gm\eta}{c_0^2 r} x[H^0_{+,x} + x^{1/2}H^{1/2}_{+,x} + xH^1_{+,x} + x^{3/2}H^{3/2}_{+,x} + x^2 H^2_{+,x}] \tag{18}$$

with H_+ and H_x components found to match those introduced in [2]. In these components,

$$x = (\frac{Gmw}{c_0^3})^{2/3} \tag{19}$$

and w is the orbital frequency of an assumed circular orbit. m is the total mass of the system, ∂m is the difference between the two masses, and η is $\frac{m1m2}{m^2}$. The variables c and s represent the cosine and sine of the angle of inclination the binary plane makes with the observer, respectively. The variable Ψ represents a basic phase variable defined as:

$$\psi = \phi(t) - \frac{Gmw}{c_0^3} ln(\frac{w(t)}{w_0}) \tag{20}$$

where:

$$\phi(t) = \phi_c - \frac{1}{\epsilon}[\Theta^{5/8} + (\frac{3715}{8064} + \frac{55}{96}\epsilon)\Theta^{3/8} - \frac{3\pi}{4}\Theta^{1/4} + (\frac{9275495}{14450688} + \frac{284875}{258048}\epsilon + \frac{1855}{2048}\epsilon^2)\Theta^{1/8}] \tag{21}$$

and

$$\omega(t) = \frac{1}{8Gm}[\Theta^{-3/8} + (\frac{743}{2688} + \frac{11}{32}\epsilon)\Theta^{-5/8} - \frac{3\pi}{10}\Theta^{-3/4} + (\frac{1855099}{14450688} + \frac{56975}{258048}\epsilon + \frac{371}{2048}\epsilon^2)\Theta^{-7/8}] \tag{22}$$

In these equations, a dimensionless time variable, Θ represents:

$$\Theta = \frac{c_0^3 \eta}{5Gm}(tc - t) \tag{23}$$

with tc being a time constant, ϕ_c represents a phase constant at time tc. These equations may be put into Mathematica to create a representation of a binary in-spiral system and its corresponding frequency chart that an observer may be able to record. An example of this is provided below in figures 3 and 4. Figure 3 represents a binary in-spiral system with two arbitrary identical masses until merging. Figure 4 demonstrates what may be observed from the system depicted in figure 3. This is a rather rudimentary model when compared to actual observations that have been made by LIGO and Virgo [5], however it is advanced enough to demonstrate the capabilities of the modeling, indicating that with more refinement, a much more accurate system could be achieved.

Conclusions

This paper sought to demonstrate the potential for cluster computing to aid in numerical general relativity calculations. To achieve this, a cluster computer composed of ten Raspberry Pi 3 Model B's was created to perform computations. At first, an attempt was made to redefine the Kerr metric in terms of two unknown variables, Ω and ψ. Following that, gravitational wave simulations were made to see if a cluster computing system could successfully model a rudimentary binary in-spiral system, which was successfully done. In the future, it would be ideal to use the Einstein Toolkit, an open source software designed for modeling gravitational waves, to model a gravitational wave system more accurately. Other implications of this project could be to attempt to create a solution for more advanced metrics, like Kerr-Newman black holes, or create more accurate gravitational wave calculations.

Appendix A

Cluster computing is connecting computers together so that they may act under a single system. The system itself is is generally composed of a single "master" node that interacts with the user and coordinates the other "slave" nodes. These slave nodes generally have their own operating systems, memory, and disk space. They are often coordinated by the master node with little, or in most cases, no direct interaction with the user.

There are many benefits to using cluster computers, several of which apply directly to the application presented in this paper. First, cluster computers provide a substantially lower cost solution than a single computer with similar specifications. Second, it is very easy to increase or decrease nodes, depending on size and computing requirements. Third, the modular nature of cluster computers makes them very reliable for both data storage and recovery in the event of failure.

However, there are also potential drawbacks to using cluster computing. These drawbacks often manifest in improper use of the capabilities that cluster computers offer. In order to interact with the different nodes, the master must divide up the work that must be completed, then send and receive information from each of the nodes. As this explanation implies, for processes that are rather quick, cluster computing will take much longer to complete the operation than a single node would take. An example of this would be calculating the number pi for fifteen digits after the decimal place. This is a simple computation that can easily be done on a single node, a computation that when moved to multiple nodes, ends up taking more time. The timings are shown in the table below from multiple Raspberry Pi Model 3B's:

Number of Raspberry Pis	Time to Compute Pi
1	0.001327
3	0.003389
5	0.005092
10	0.007940

For the purpose of this project, cluster computing is an optimal option. This is because the information that will be processed in this project will require rather substantial computing power and it is possible to be analyzed in a parallel format.

The specifics on the cluster computer built for this project are as follows: 10 Raspberry Pi Model 3B's, 9 16 Gigabyte SD cards, 1 64 Gigabyte SD cards, 10 Ethernet cables, a 10 port USB hub, and a 10 port desktop switch capable of supporting 10 static IP's. The Pis are all housed in a single 3-D printed case designed to neatly minimize the space required to store all of the Pi's.

Aside from physical setup, an electronic setup is also required. In order to properly setup the system there are multiple guides and tutorials to help setup clustering capabilities on various operating systems [6]. For this project Raspbian was chosen as the default operating system for multiple reasons. First, it is a stable operating system that has a good development community behind it so any issues that may occur should be easy to remedy. Second, many cluster computing device are built with Raspbian as an operating system so finding literature or guides to aid in developing a cluster computer should be easy. Third, Raspbian comes with Mathematica for free, saving costs on the project.

An important note here is that the description will be done assuming all SD cards are the same size. In reality there are some slight changes that would need to be made if the SD cards are different sizes, the significance of which will be explained later. First, Raspbian must be installed on one of the SD cards. Once it is successfully installed, the password must be changed using the command:

```
passwd
```

Following that, in the preferences tab open "Raspberry Pi Configuration", choose the interfaces option and enable SSH. Then, in terminal make sure that the device is updated and Fortran is downloaded onto the Pi with the commands:

```
sudo apt-get update

sudo apt-get install gfortran
```

when prompted to continue, press 'Y'.

After that, make a directory to install MPICH onto, and install version 3.0.4 into that directory. MPICH is a messaging passing interface that is designed for use in parallel processing. It has direct use in many supercomputers, including some of the fastest super computers in the world. It is installed with the commands:

```
mkdir /home/pi/mpich3

cd mpich3

wget http://www.mpich.org/static/downloads/3.0.4/mpich-3.0.4.tar.gz

tar xfz mpich-3.0.4.tar.gz

sudo mkdir /home/rpimpi

sudo mkdir /home/rpimpi/mpich3-install

sudo mkdir /home/pi/mpich_build

cd /home/pi/mpich_build

sudo /home/pi/mpich3/mpich-3.0.4/configure -prefix=/home/rpimpi/mpich3-install

sudo make

sudo make install
```

Once MPICH is installed, it is possible to test the computer. To do so, create a file file called "Machinefile" and input the IP address of the current Pi and add commands to test the setup. This is done with the following commands:

```
sudo nano /home/pi/.profile

[Scroll down to the bottom and type the following:
# Adding MPI to the PATH

PATH="$PATH:/home/rpimpi/mpich3-install/bin"
Then, save and exit]

cd /home/rpimpi/mpich3-install/bin

cd ~

which mpiexec

which mpicc

mkdir mpi_testing

cd mpi_testing

sudo nano machinefile

[Input the IP of the Pi in the first line, then save and exit.]

mpiexec -f machinefile -n 1 hostname
```

If "which mpiexec" does not return anything, it is necessary to restart the computer that is being used before continuing. The following set of commands should return the name of the Raspberry Pi that is currently online. If the number is changed to something like two or three, it will display the hostname of all of the nodes. In order to compute a more meaningful calculation, like the value of Pi, it is necessary to do the following:

```
cd ~/mpi_testing

mpiexec -f machinefile -n 1 ~/mpich_build/examples/cpi
```

This should return the value pi, with both the error in calculation and the time it took to calculate. Similar to before, as the number of nodes increase, the timings and process numbers should change.

Next, the SD card is ready to be cloned. This is where the differences occur if multiple SD cards sizes are used. If this was done on multiple different sized SD cards, the process should be repeated for each different size card. Following that, a backup should be made using a program like Win32DiskImager. This clone may be saved and then re-written onto every other SD card.

In theory, the computer can run in parallel at this point, however it would be necessary to ssh into each node every-time something is run. To avoid this, do the following:

```
ssh-keygen -t rsa -C "raspberrypi@raspberrypi"

[Press Enter
You can enter a passphrase, however it is not advised]

cat ~/.ssh/id_rsa.pub | ssh pi@IPAddressofOtherPi "mkdir .ssh;cat >> .ssh/authorized_keys"

[Yes
Enter Password]
```

Now, if there are just two Pis in the cluster it would be possible to test. If there are more, the previous section would be repeated for each different node. Following that:

```
cd mpi_testing

sudo nano machinefile

[Edit the machine file, adding each IP to its' own line.

Then, save and exit.]

mpiexec -f machinefile -n NumberofNodes ~/mpich_build/examples/cpi
```

As was the case before, the number of nodes may be changed to allocate for the number of nodes in the cluster. Not every node is necessary for this calculation, using more nodes is a viable option and will cause no issues. If at this point, or any point prior, the commands do not run properly, it is likely that either the IP address of on of the Pis changed, or one of the commands was improperly implemented. Assuming, however, that no issues occur the cluster computer is successfully setup. At this point, for clarity, it is helpful to change the names of all of the slave nodes to something like PiXX, where each "X" denotes a number from zero to nine. In order to set the computer to run in parallel in Mathematica, see the instructions below.

It was previously mentioned how cluster computing provides potential advantages and disadvantages. Aside from that, it is important to assess the potential benefits from using a certain number of nodes. A real world assessment of this was constructed using 10 Raspberry Pis running a test script through Mathematica on a varying number of nodes. The results follow the trend illustrated below:

*** in practice, the more nodes used the faster the faster the script was evaluated. However, the amount of time saved per node decreases substantially. This indicates that for very complex computations, or if multiple operations were to be run at the same time, using tens or hundreds of nodes becomes a viable and potentially even preferred option.

With that in mind, the intended use of the cluster has a large effect on the number of nodes included. As was shown before, the number of nodes used increases the time it takes the system to pass and retrieve all of the information. However, this point is likely to be reached long after the point that the system is financially viable.

Cluster Computing in Mathematica

To properly setup cluster computing capabilities in Mathematica, it is necessary to access the parallel kernel configuration. This is done by evaluating the command:

```
SystemInformation[]
```

Then accessing the front end interface and selecting the parallel configuration option. Following that, select remote kernels and add a new host for each node other than the master. For the hostname, use the IP address of the Pi. Then, it is necessary to use a custom launch command, and replace

```
-| '3' '1'
```

with

```
hostname@IP
```

where 'IP' represents the IP address of the Pi. Following that, it is necessary to enable and initialize the kernels. Since different clusters may be used at any point, it is recommended that the kernels be disabled at the end of each session and re-enabled at the start of any other session.

References

[1] Moore, T. A. (2013). *A General Relativity Workbook*. Mill Valley, California: University Science Books.

[2] Blanchet, L., Iyer, B. R., Will, C. m., and Wiseman, A. G. (1966). Gravitational waveforms from inspiralling compact binaries to second-post-Newtonian order.

[3] "Binary Inspiral Gravitational Waves from a Post-Newtonian Expansion" from the Wolfram Demonstrations Project http://demonstrations.wolfram.com/BinaryInspiralGravitationalWavesFrom APostNewtonianExpansion/ Contributed by: Leo C. Stein

[4] Satya Mohapatra "Scalogram of Gravitational Wave from a Binary Black Hole Inspiral" http://demonstrations.wolfram.com/ScalogramOfGravitationalWaveFromA BinaryBlackHoleInspiral/ Wolfram Demonstrations Project Published: September 13, 2011

[5] LIGO Laboratory (2017). Gravitational wave signatures detected by each observatory. *California Institute of Technology*. Retrieved from https://www.ligo.caltech.edu/image/ligo20170927a

[6] Kinsley, H. (2014) *Build Your Own Supercomputer*. Retrieved from https://www.youtube.com/watch?v=13x90STvKnQ

1 Figures

Figure 1

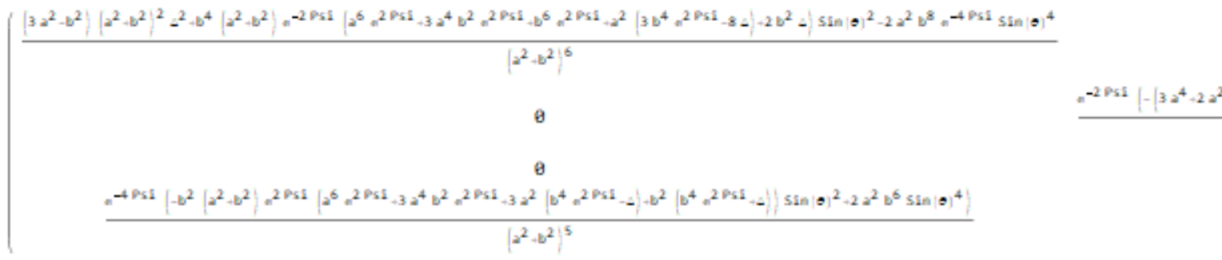


Figure 2

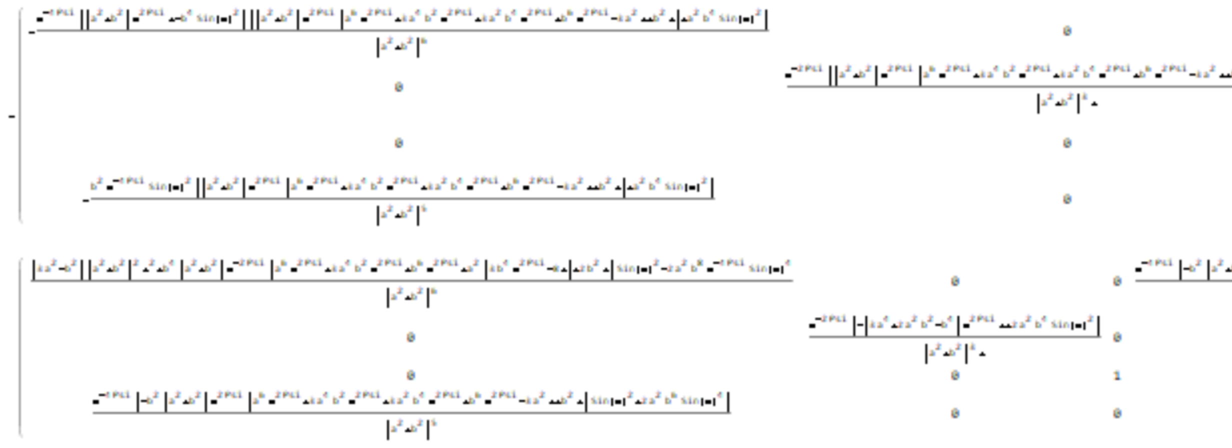


Figure 3

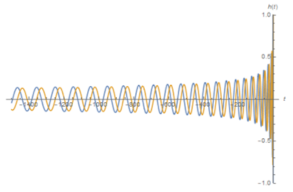
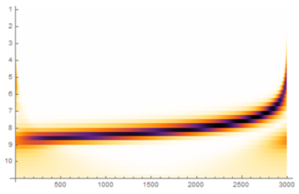


Figure 4


1
2
3
4
5
6
7
8
9
10
500
1000
1500
2000
2500
3000