ESTIMATING BINARY BLACK-HOLE MERGER SIGNALS

by

Joseph Hill

A senior thesis submitted to the faculty of

Brigham Young University–Idaho

in partial fulfillment of the requirements for the degree of

Bachelor of Science

Department of Physics

Brigham Young University–Idaho

March 2013
DEPARTMENT APPROVAL

of a senior thesis submitted by

Joseph Hill

This thesis has been reviewed by the thesis advisor, thesis coordinator, and thesis committee and has been found to be satisfactory.

Date Richard Hatt, Thesis Advisor

Date Stephen McNeil, Thesis Coordinator

Date David Oliphant, Committee Member
ABSTRACT

ESTIMATING BINARY BLACK-HOLE MERGER SIGNALS

Joseph Hill

Department of Physics

Bachelor of Science

A Particle Swarm Optimization (PSO) based fitting method is used to estimate binary black hole merger signals within noisy data. This method optimizes knot spacing in a spline fitting function to find the best least-squares cubic spline fit to the data. PSO is a genetic algorithm optimization technique. A spline is a continuous, piecewise defined, polynomial function. It is shown that when analyzing noisy data, the number and spacing of knots is critical to achieving the best possible fit.
ACKNOWLEDGMENTS

First and foremost, I must give my deepest heartfelt thanks to my eternal companion Meagan who has helped me endlessly with not only this paper, but throughout my entire college education. She has been my part time secretary, editor, and cheering squad for the past several years. For all of her support, I cannot say thank you enough. There are many professors I would also like to thank. Dr. Evan Hansen who, as a freshman in a foundation of science class, helped me decide to study physics. Dr. Brian Tonks, who has, over the course of many semesters, shown me his love for astronomy and helped me to develop my own lifelong fascination for astrophysics, his enthusiasm for his field is inspiring and I thank him. Dr. David Stowell, who has been integral in my understanding of calculus and numerical analysis. Dr. Soumya Mohanty, my research advisor while I was at the University of Texas at Brownsville doing the research presented herein. To be sure, there are many professors that have influenced the past four years of my life, and I thank each and every one of you. Last, but not least, I owe a debt of gratitude to my parents who believed in me long before I did. Their support has seen me through many difficult days for which I am appreciative.
# Table of Contents

Department Approval ........................................................................................................III

Abstract ..............................................................................................................................V

Acknowledgments ...............................................................................................................VI

Table Of Contents .............................................................................................................VII

List Of Figures ...................................................................................................................IX

CHAPTER 1 .........................................................................................................................1
  Data Analysis Introduction ...............................................................................................1

CHAPTER 2 .........................................................................................................................4
  Spline Fitting ....................................................................................................................4

CHAPTER 3 .........................................................................................................................6
  Particle Swarm Optimization .............................................................................................6
  Initialization .......................................................................................................................7
  Particle Movement ...........................................................................................................8
  Movement Phases ............................................................................................................9
  Inertia ...............................................................................................................................9
  Testing .............................................................................................................................10
  Pros and Cons to Using Particle Swarm Optimization....................................................11

CHAPTER 4 .........................................................................................................................12
  Gravitational Waves ......................................................................................................12
  Types of Detectors .........................................................................................................13
  Laser Interferometer Gravitational-Wave Observatory....................................................15


List of Figures

Figure 1: Horn shaped antenna owned by Bell Labs .................................................. 1
Figure 2: Example of direct interpolation of noisy data ........................................... 4
Figure 3: Comparison of even knot spacing and optimized knot spacing ................. 5
Figure 4: Example of swarm behavior in a school of fish ...................................... 6
Figure 5: Surface and contour plot of the Ackley test function ............................... 11
Figure 6: Joseph Weber and his resonant mass detector ..................................... 13
Figure 7: Diagram of a laser interferometer .......................................................... 14
Figure 8: Aerial view of LIGO facility in Livingston, Louisiana ......................... 15
Figure 9: Comparison of gravitational-radiation waveforms ............................... 18
Figure 10: Comparison of optimized spacing with 10, 20, and 30 knots ............... 20
Figure 11: Comparison of Average fitness and number of knots used ..................... 21
Figure 12: Comparison of average runtime and fitness per number of particles ....... 22
Chapter 1

Data Analysis Introduction

The Big Bang model for the formation of the universe predicts large amounts of high-energy radiation output. In 1965, Arno Penzias and Robert Wilson attempted to use a 20-foot tall horn shaped antenna, shown in figure 1, in connection with an early satellite system named Echo to analyze radio signals from the spaces between nearby galaxies. No matter where they pointed the antenna, however, they received constant interference. After extensive error checking, cleaning, and modifications, Penzias and Wilson realized that a group from Princeton had predicted this exact phenomenon. The constant background “noise” picked up by their antenna was the same radiation predicted by the Big Bang theory. Noise is considered any signal collected that is not directly desired. The radiation had red-shifted.
so much that it was found in the microwave part of the spectrum and was being detected by their antenna.

Background radiation is only one source of noise that can affect a received signal. Many Earth-based and cosmic sources can produce noise that may interfere with data collection. The ratio between the desired signal and additional noise is an important factor in data analysis. When receiving and analyzing a signal, signal-to-noise ratios must be considered and efforts should be taken to reduce or eliminate noise where possible.

Data analysis is a scientific field dedicated to the process of finding a significant subset of data from a larger set of data. There are many research fields and applications involving or requiring the use of data analysis. Much of this work is done via computer. Computer data analysis usually concerns itself with analyzing quantitative data in order to find repetition, patterns, or significant meaning to otherwise meaningless information. Most data analysis is done with the end goal of understanding the signal. Data mining is a discipline of the field that focuses on data analysis for predictive rather than descriptive understanding. The actual task of data mining is usually a fully automated task where computer programs are tasked with finding meaningful information. One area where data mining can be applied is in signal analysis.

When analyzing noisy data to look for a particular known signal, there are methods that can significantly reduce noise, even with low signal to noise ratios. A spline fitting technique in conjunction with an optimization method for knot spacing is used to accomplish this task. This method was the focus of my research during a summer Research Experience for Undergraduates program at the University of Texas at
Brownsville. Within this research, the development and understanding of a particle swarm optimization method was developed first. After understanding and implementing this method, it was tested using multiple benchmarking functions. The optimization technique was applied to finding optimal knot spacing within a spline-fitting program. Using these two methods in unison allows a computer to fit a smooth curve to noisy data. Finally, this technique was applied to simulated noisy signals that are similar to those expected from data collected at the Laser Interferometer Gravitational-Wave Observatory facilities.
Chapter 2

Spline Fitting

When analyzing noisy data, direct interpolation can result in unwanted oscillations of the produced curve as seen in figure 2. Data interpolation using a least squares method can lead to a smoother, but poorly fit curve, especially when the data has a low signal-to-noise ratio. Spline fitting is a data analysis technique used for estimating, via least squares, the parameters of a spline polynomial (1). A spline polynomial is a piecewise defined function that is created so that each piece meets in a way that the function is continuous and differentiable.

Spline interpolation first splits data up into sections. The positions where the data is split are referred to as knots. Each section of data is fit to a user defined $N^{th}$ order
polynomial and each of these polynomials are fit together in a way that the fit curve is continuous at each knot. Data can be broken up into evenly sized or evenly spaced sections of data. User-defined knot spacing can also be applied. As shown in figure 3, evenly spaced knot placement can result in a very poorly fit spline (2). Other factors can affect the resulting curve such as number of knots and the order of the fit polynomial.

When using spline fitting to estimate a signal from noisy data, the size of each spline has a large effect on the fit of the curve. In order to achieve an optimal fit between the data and fit curve, knots are placed such that the $\ell^2$-norm between the data and the fit curve is minimized, thus this becomes a problem of optimization.
Chapter 3

Particle Swarm Optimization

Particle swarm optimization is a population-based evolutionary algorithm inspired by the swarming behaviors of flocking birds or schooling fish. As an example of how the swarm functions, imaging a school of fish, such as in figure 4, searching for a food source. Each fish swims from one place to another without the need to consult its neighbors. Each fish independently searches for a food source, remembering the best food source that it has found so far as well as communicating the best food source found by any member of the school. Over time, the fish all converge on the best source of food available. Within the optimization program, particles act like the fish, each searching for a location that minimizes the test function. Each location discovered by a particle is a potential solution to the optimization problem.
Within a swarm optimization technique, a group of particles is generated randomly throughout the N-dimensional search space. The particles move independently from each other, moving from one location to another. As a new location is discovered, its position is tested against a fitness function to see if it is a more desirable location than the previous. Particles in the swarm adjust their motion based on the best personal and global positions within the swarm. During the initial search phase, particles have high inertia, which tends to keep them spread out over the search space. Over time, as no new favorable locations are discovered, inertia decreases, and the swarm enters a convergence phase where particles converge toward the best global position (3).

**Initialization**

Particle swarm optimization can be used to minimize functions with any number of parameters. This requires that the swarm is capable of searching through any N-dimensional space using any number of particles. As the swarm is initialized, particles are generated randomly within an N-dimensional hyper-cube search space. Each particle is also generated with velocities in random directions having magnitudes between zero and a maximum velocity. Maximum velocity prevents particles from spending too much time outside of the search area and is calculated as one-half the width of the hyper-cube along any single axis.

$$\text{velocity}_{\max} = \frac{\text{Range}_{\max} - \text{Range}_{\min}}{2}$$

Because the optimization method is not an actual simulation of movement, time step size is unimportant. For purposes of unit analysis, each iteration is considered one time step.
During initialization, particles receive an inertia value. All particles in the swarm have the same value for inertia. The position of each particle in the swarm is then used to test a fitness function. Values for each particle are stored as \( pBest \), or the best known value for each particle. One location is considered better than another if it results in a lower value within the test function. As particles move within the search space, they will update their best-known positions as new and better positions are discovered. The lowest value for \( pBest \) is also stored as \( gBest \), the best-known position globally.

**Particle Movement**

At each iteration, particles move according to their current velocity.

\[
\text{position}_{n+1} = \text{position}_n + \text{velocity}_n
\]

After moving, the particle locations are again used to evaluate the fitness function. New values are used to update \( pBest \) and \( gBest \) if necessary. Particle velocities are updated using pseudorandom accelerations toward \( pBest \) and \( gBest \).

\[
v_{n+1} = \text{inertia} \cdot v_n + M_1 \cdot (pBest - \text{position}_n) + M_2 \cdot (gBest - \text{position}_n)
\]

When calculating velocity, \( M_1 \) and \( M_2 \) are randomly generated diagonal matrices with values between zero and a user-defined acceleration parameter. Accelerations are larger depending on the distance that the particle is from each of the best-known positions. Using the \( M_i \) matrices allows for random variations in the movement of the particles while still keeping them within the search space. In the case where a particle does attempt to leave the defined search space they are allowed to move freely, but the fitness function is not calculated for their position until they are back within the search space. This is
done so that the particles are allowed to maintain their natural movement while not wasting computational time evaluating functions that lie outside the search space of interest.

**Movement Phases**

Particles within the swarm go through two movement phases. Phases are defined by the type of movement the particles display. During the first phase, particles search throughout the entire search space for possible solutions. During the second phase, particles converge on the best solution and a focused search is done. It is possible for the swarm to go back and forth between the phases several times before a final solution is returned.

**Inertia**

During the initialization phase, a hyper-sphere is created with its center at the initial $gBest$. As $gBest$ is updated, its location is compared to the radius of the sphere to see if the new $gBest$ has moved outside of the sphere. Particles are initialized with a high value for inertia. During each step, if $gBest$ has not moved outside of the sphere, a counter is kept and used to calculate a new inertia for the particles:

$$\text{inertia} = \text{inertia}_0 - \frac{.4 \times \text{count}}{\text{count}_{\text{max}}}$$

If the position of $gBest$ does move outside of the sphere radius, $\text{count}$ is reset to zero and the sphere is re-centered at $gBest$. 
When count is a low value, inertia is high. This high value for inertia allows the particle velocity to dominate its motion through the search space. Initially, when the particles are discovering several new positions for gBest, the sphere will tend to move rapidly and count stays quite low. This is the search phase for the particles. They search throughout the entire search space looking for the most likely candidate for the global minima. As fewer new positions for gBest are found, the sphere stops moving and count increases toward count\textsubscript{max}. As count reaches large values, inertia becomes small and the acceleration toward gBest and pBest has larger impact on the motion of the particles. As inertia decreases, the particles enter the second phase, a convergence phase. During this phase, the particles will tend to congregate and swarm near the area of gBest. The swarm conducts a more thorough search around the area. During this phase, it is still possible to find a new gBest outside of the sphere, causing count to reset and the particles to resume a search phase. During the convergence phase, if no new gBest is found outside of the sphere, the search will end when count reaches count\textsubscript{max}.

**Testing**

The purpose of any optimization technique is to find the minima (or maxima) of some test function. Many variations of particle swarm optimization exist, using varying topologies. In order to compare one to another, several standard test functions have been developed (4). The developed particle swarm optimization method was tested using several of these functions. Each function is designed to have several local minima and one or more known global minima. If the optimization method fails to find global minima and instead returns local minima, the method needs to be revised, or a different method
used. Figure 5 shows one such function, the Ackley test function, in two dimensions. The z-axis corresponds to the fitness as a function of x and y. The Ackley function is dependent on the number of dimensions \((N)\) of interest and is defined by:

\[
f(x) = 20 + e - 20e^{-0.2 \sqrt{\sum_{i=1}^{N} x_i^2}} - e^{\sum_{i=1}^{N} \cos(2\pi x_i)}
\]

Running multiple benchmarking tests allows for comparison in both accuracy and computational time of the developed algorithms.

**Pros and Cons to Using Particle Swarm Optimization**

Particle swarm optimization has many advantages when being applied to data analysis. It is easy to implement, runs quickly, and does not rely on the gradient of the function being optimized in order to find a minima. Not relying on a gradient search allows swarm optimization to be applied to inconsistent and non-differentiable functions.

Particle swarm optimization is easily adaptable to any number of dimensions. When applied to spline fitting, the number of dimensions used in the optimization translates directly to the number of knots used in the spline.
Chapter 4

Gravitational Waves

Albert Einstein predicted gravitational waves as a natural consequence of his theory of general relativity. Gravitational waves are ripples in space-time, propagating out from their source at the speed of light (5). Observations of binary pulsar stars have been used to confirm several predictions made by general relativity. Indirect evidence of gravitational wave existence has been discovered as well. As gravitational waves propagate through space, they cause small distortions in the shape of space. As a gravitational wave passes a ring of particles, space is distorted asymmetrically and the distance between particles changes. As a slow moving (relatively speaking) observer within the distortion, it is impossible to directly detect the expansion and compression of the space affected by the passing gravitational wave. However, there are ways to detect these changes indirectly. Indirect detection of gravitational waves can be done by taking advantage of these distortions.

Violent, large-scale astronomical events offer to be promising sources of gravitational waves. Such events include supernovae, spinning-down pulsars, and binary systems of neutron star, black-hole, or white dwarf mergers. The Big-Bang theory of the
origin of the universe predicts that gravitational wave background noise should also be detectable today. Gravitational wave background noise consists of the superposition of many gravitational waves of random phases. The gravitational wave background (6) is similar in nature to the cosmic background radiation discovered in the 1960’s by Penzias and Wilson.

Types of Detectors

There are currently two different types of instruments developed specifically for gravitational wave detection (7). In the 1960’s, resonant-mass detectors, such as the one in figure 6, were developed. These detectors use large metal cylinders or spheres as test masses. The slight changes in the geometry of the test mass induce a small electrical signal. Amplifiers enhance the induced signal, which transducers are able to detect. The test masses in these detectors are aluminum with masses of $\sim 2.3 \times 10^3$ kg operating at cryogenic temperatures. The limitation on these bar-detectors comes from their sensitivity, which is $\sim \pm 20$ Hz from the resonant frequency of the detectors at $\sim 900$ Hz. Resonant-mass detectors are only sensitive enough to detect very powerful gravitational waves. Due to their limited sensitivity, these detectors have not yet produced any scientifically significant data, and only a small handful of this type of detector are known to still be in operation.
Laser interferometry is a much more sensitive method used to detect gravitational waves. The basic process for laser interferometry is shown in figure 7. Mirrored test masses at each end of a laser interferometer arm are used to contain a laser beam. Laser light is split at the center of the observatory and sent down each arm of the interferometer. Suspended mirrors trap the light within the interferometer arms and a high-energy beam is contained. As light is recombined, it is passed through a photodetector. By precise placement, destructive interference between the recombined beams is maximal and the photodetector detects no light. As a gravitational wave passes over the interferometer, the arms contract and expand independently depending on the orientation of the interferometer to the gravitational wave source. This variation in arm length causes a shift in orientation of light within each light storage arm. As the shifted light recombines and passes through the photodetector, interference is no longer completely destructive and a signal is collected. The largest of these interferometers have arms up to 4km long and are sensitive to changes from ~43km to ~10,000km.
or ~30Hz to ~7000Hz. Plans for a space based laser interferometer are being developed that will host arm lengths up to $5 \times 10^6$km (8).

**Laser Interferometer Gravitational-Wave Observatory**

The Laser Interferometer Gravitational-Wave Observatory (LIGO) operates two interferometers, each with 4km long arms. They are located near Richland, Washington and Livingston, Louisiana. Figure 8 shows the Livingston facility. Over 3000 km separate the two detectors, which allows for triangulation between the two detectors to discover the source direction of a detected wave. A third observatory is in production in Europe, which will allow LIGO scientists to more accurately determine source locations. LIGO has gone through multiple stages since its initial run in 2002 without any gravitational wave detection. Enhanced detectors were implemented and Enhanced LIGO completed a run in 2010 with no gravitational wave signals detected. Advanced detection features are currently being implemented and Advanced LIGO is expected to begin operations in July 2013.

![Figure 8: Aerial view of LIGO facility in Livingston, Louisiana](image courtesy of science.psu.edu)
Even with the most advanced detection devices possible, there are inherent limits to the detection ability of an Earth-bound laser interferometer. Seismic activity, electromagnetic interference, mirror or lens imperfections, thermal expansion, and residual gas in the detector arms can all cause noise in the detector. Because of these sources of error, laser interferometers have a maximum sensitivity range that is inherent in the design of the system.
Chapter 5

Binary Black-Hole Merger Waveform

Due to the inherent noise in the LIGO data, a low signal-to-noise ratio can be expected. It is therefore advantageous to have accurate predictions of different types of waveforms. By comparing received signals to these predicted waveforms, false positives can be eliminated quickly. There are a number of gravitational waveforms available from NASA’s Astrophysical Gravitational-Wave Sources Archive (9). Significant time and research has gone into independent development of several different waveforms.

Converging binaries are an excellent potential source of gravitational waves that Advanced LIGO should be able to detect. Binary black-hole merger waveforms are one of the waveforms calculated by multiple independent research groups. The three groups who worked on the project were from the NASA Goddard Space Flight Center (GSFC), The Center for Gravitational Wave Astronomy at the University of Texas at Brownsville (UTB), and a group lead by Frans Pretorius (Pretorius)
Each group simulated the merger phase for binary black-holes using various configurations and calculated the resulting waveform for an interferometer type detector. The waveforms, figure 9, show significant agreement between the groups who used different methods to solve the Einstein field equations for coalescing black-hole binaries (10). The final waveforms were processed over several days using multiple supercomputers. Data from these waveforms can be modified, using a computer script, to simulate an expected signal from different sources based on red-shifted mass, luminosity distance, inclination, and polarization.

Modifying these waveforms can allow data analysts to compare a detected signal against the waveform. These modified waveforms may then be used in simulation of various signals, which can be used to test analysis methods of LIGO data. Noise inherent in the collection method used by LIGO is simulated into the data by first normalizing the waveform using the desired signal-to-noise ratio as a scalar, and then adding Gaussian white noise to simulate noisy data of the desired waveform. Testing analysis methods on
a simulated signal is helpful in finding and eliminating sources of error during analysis of actual data.
Chapter 6

Bringing it all together

Using a binary black-hole merger waveform modified to simulate a merger of two black-holes with mass equal to one hundred times the mass of the Sun at a distance of ten mega-parsecs with inclination and polarity of zero, noisy data was generated to simulate realistic noisy data with a signal-to-noise ratio of ~25, a ratio representative of expected advanced-LIGO data. When applying a spline fit to the data, too many or too few knots used will result in a poorly fit curve as shown in figure 10. In order to determine the

![Figure 10: Comparison of optimized spacing with 10, 20, and 30 knots](image)
ideal number of knots for spline fitting, a particle swarm optimization method was used to determine knot spacing, which minimized the $\ell^2$-norm between the fit curve and the original waveform.

There are few variables in the particle swarm method that can be modified to affect efficiency. The number of particles used can be changed easily. Adding more particles allows for more locations in the search area to be tested per iteration, but is also computationally more expensive. Using a varying number of particles in the swarm, average runtimes were compared. Average fitness is compared at the same time to see if more particles translate to a better fit. As shown in figure 11, fitness does not improve with more particles in the swarm, but computation times significantly increase with each additional particle.

When analyzing noisy signal data and attempting to use a spline fitting technique to fit a curve, all knots must lie between the start and endpoints of the signal. The analysis method developed uses a hyper-cube search area allowing each dimension to have a search range between the start and end of the signal. The number of dimensions used in the swarm optimization method translates directly into the number of knots used.
for the spline fitting method. The position of a single particle in each dimension translates to the location of a single knot within the data. This does allow for the possibility that multiple knots will be placed near or directly on top of one another. In such a case, the fitness of the fit curve is generally very poor. If the best fit is found with closely spaced knots, it is indicative that too many knots are being used to fit the data, see figure 10c.

Tests using between one and one hundred knots averaged over thirty individual runs, shown in figure 12, show that between fifteen and twenty-one knots are ideal when analyzing noisy data to look for black-hole merger signals of the source tested. Future work may show if this range is ideal for other test sources of black-hole merger signals.

Fitness is determined by taking the $\ell^2$-norm, also known as the vector norm, of the difference between the noisy data and a discretization of the fit curve.

$$\text{fitness} = |\text{data} - \text{spline}|$$

Using particle position as knot location, swarm optimization can quickly calculate the optimal knot spacing for spline fitting.
Conclusion

Because numerical computation of black-hole merger waveforms is very computationally expensive, it is unlikely that enough waveforms will be on hand in the near future to implement a direct-matched filtering-based search across the entire astrophysically relevant parameter space. The method presented in this paper is a less computationally expensive alternative since it does not require prior signal waveforms but estimates them reliably at signal-to-noise ratios that are relevant for future gravitational-wave detectors.

This method allows a computer to scan LIGO data in real-time and return a fitness comparison between estimated signals and calculated waveforms. The fitness may then be used to rate the reliability of a positive match for gravitational-wave discovery.

When analyzing gravitational-wave data to search for binary black-hole merger signals, it has been shown that the spacing and number of knots used to fit a spline to the data is critical in the fit of the resultant curve. An optimization method such as particle swarm optimization can quickly locate optimal knot positioning. Using data from multiple black-hole merger waveforms, the best results are found when using fifteen to twenty-one knots. Adding particles to the swarm increases computation time without improving the fitness of the spline. To improve computational performance, an upper limit of ten particles is recommended.
Bibliography


[7] Nishizawa, A. “Search for cosmological gravitational-wave background at high frequencies”. Kyoto University, pp. 4-8.


APPENDIX A:

MatLab Code for Particle Swarm Optimization

```matlab
function [ vBest, gBest, numIterations ] = PSO( func, numParticles, particleRange, numVariables, maxIterations, inertia, Cp )

% Particle Swarm Optimization:
% Creates particles that move throughout a designated hypercube search
% space. On each iterated movement, particle position is compared against a
% given fitness function. Particles adjust movement based on individual as
% well as global best known locations. Best location is considered to be
% one that minimizes the fitness function.
%
% User Input
% func = name of function to minimize, input as a string
% numParticles = number of particles in the swarm
% particleRange = [minRange maxRange]
% numVariables = number of variables in the function
% maxIterations = maximum number of iterations before stopping
% inertia = movability coefficient of the swarm
% Cp = user defined parameters for acceleration

%% Initialization
% Convert function name to function handle
Func = str2func(func);

% pBest = position of the lowest known value of the particle
pBest = inf(numVariables,numParticles);

% gBest = position of the lowest known global value
gBest = inf(numVariables,1);

% vBest = lowest known global value
vBest = inf;

% bestValue = value of the particle at pBest
bestValue = inf(1,numParticles);

% particleValue = current value of the particle
particleValue = inf(1,numParticles);

% vMax = velocity limit in any direction
vMax = (particleRange(2) - particleRange(1)) / 2;

% velocity = speed of each particle, initially zero
```
velocity = 2*vMax*rand(numVariables,numParticles)-vMax;

% Create initial position of swarm randomly throughout the max and min range
position = (particleRange(2) - particleRange(1)) * ... 
  rand(numVariables,numParticles) + particleRange(1);

% Store initial inertia
inertia0 = inertia;

% Create a hyper-sphere around gbest, while gBest remains within the
% sphere, inertia decreases with each iteration. If gBest leaves this area,
% inertia resets to its initial value and boxCenter resets to gBest
count = 0;
maxCount = 100;
boxCenter = gBest;
boxRadius = 0.001 * (particleRange(2) - particleRange(1));

%% Used to animate and output video of particle movement
%% writerObj = VideoWriter('PSO.avi');
%% writerObj.FrameRate = 10;
%% open(writerObj);

%% Main Loop
for numIterations = 1:maxIterations

  %% Calculate the values for each particle and update best values
  for i = 1:numParticles

    % Only calculate fitness if the particle is within the search area
    if max(position(:,i)) > particleRange(2) || ... 
      min(position(:,i)) < particleRange(1)
      %particleValue = current value of each particle
      particleValue(i) = inf;
    else
      particleValue(i) = Func(position(:,i))';
      % Compare particleValue with pBest, update pBest value
      if particleValue(i) < bestValue(i);
        bestValue(i) = particleValue(i);
        pBest(:,i) = position(:,i);
      end %if
    end %else
  end %i

  %% Compare and update gBest

  % determine best global position by comparing current gBest with lowest
  % current gBest value
  [currentMin, Index] = min(particleValue);
  if currentMin < vBest
    vBest = currentMin;
    gBest = position(:,Index);
    % Check to see if updated gBest is still within the box radius
    if (norm(gBest - boxCenter) > boxRadius)
      % Center box on new gBest and reset inertia counter
      boxCenter = gBest;
      count = 0;
    end %if
  end %if

  %% Particle Motion
%create random diagonal matrices for acceleration
M1 = diag(rand(1,numVariables)*Cp);
M2 = diag(rand(1,numVariables)*Cp);

% Calculate new particle velocity using acceleration towards pBest and gBest
for j = 1:numParticles
    velocity(:,j) = inertia .* velocity(:,j) + ...
        M1 * (pBest(:,j) - position(:,j)) + ...
        M2 * (gBest - position(:,j));

    % Used to animate and output video of particle movement
    % scatter(gbest(1),gbest(2),'s', 'k')
    % hold on
    % scatter(position(1,j), position(2,j))
    % axis([particleRange(1) particleRange(2) ...
    %     particleRange(1) particleRange(2)])
end %j

% Used to animate and output video of particle movement
% frame = getframe;
% writeVideo(writerObj,frame);
% hold off
% pause(.00001)

% Apply speed limits on particle velocity
velocity = (velocity >= 0) .* (velocity <= vMax) .* velocity + ...
    (velocity <= 0) .* (velocity >= -vMax) .* velocity + ...
    vMax * (velocity > vMax) - vMax * (velocity < -vMax);

position = position+velocity;

% Update inertia
inertia = inertia0 - (.4 * count / maxCount);
if count == maxCount
    break
else
    count = count + 1;
end %else

end %numIterations
end %PSO
APPENDIX B:
Perl Script

Available from NASA to convert waveforms.
http://astrogravs.gsfc.nasa.gov/docs/waveforms/NRmergers/

Run using the syntax:

getDetectorStrain.pl <input data> <total redshifted mass (M_Sun)> <luminosity distance (Mpc)> <inclination (degs)> <polarization (degs)>

#!/usr/bin/perl -w

# Simple script to convert wave analysis "strain" output to usable
# hplus, hcross for data analysis purposes
# Adapted from c++ code of Sean McWilliams
# This code asusmes that only the (2,2) and (2,-2) modes contribute
# to the detected strain, and moreover, that they contribute equally.

use FileHandle;
use Math::Trig;

if(@ARGV != 5)
{
  print "Usage: $0 <input data> <total redshifted mass (M_Sun)> <luminosity distance (Mpc)> <inclination (degs)> <polarization (degs)> \n";
  exit;
}

$MADM = 1.0;
$PI = 2.0*acos(0.0);
#$C = 2.9979e8; #speed of light [m/s]
#$G = 6.6732e-11; #Newton's gravitational constant [m^3/kg/s^2]
$t_conv = 4.9169e-6; #convert t = [M] to t = [sec]
$h_conv = 4.7804e-20; #convert h = [M]/[r] to h dimensionless
$infile = $ARGV[0];
$M = $ARGV[1];
$D = $ARGV[2];
$iota = $PI*$ARGV[3]/180.0;
$pol = $PI*$ARGV[4]/180.0;


@raw_time = ();
@hRe = ();
@hIm = ();

$fileroot = substr($infile, 0, -4);
$outparams = $fileroot.".param";
$outfile = $fileroot."_at_detector.dat";

open(IN_FILE, "<$infile") || die "Unable to find file "$infile"."

$mode2pol = sqrt(5.0/(4.0*$PI))*$h_conv*$M/$D/$MADM;

$ii = 0;
while (<IN_FILE>) {
    $line = $_;
    if($line eq "\n") { # If line is empty line
        # dump
    } elsif(/\#/) { # If line has comments
        # dump also
    } else {
        chop($line);
        @data = split(/\s/,$line);
        $raw_time[$ii] = $data[0];
        $hRe[$ii] = $data[1];
        $hIm[$ii] = $data[2];
        $ii++;
    }
}

close(IN_FILE);

$dt = $raw_time[1] - $raw_time[0];
$dt = $t_conv*$M*$dt/$MADM;

@time_val = ();
@hplus_val = ();
@hcross_val = ();

open(OUT_FILE, ">$outfile") || die "Unable to create file "$outfile"."

print(OUT_FILE "#time\th plus\th cross\n";

for $jj (0 .. ($ii-1)) {
    $tnow = $jj*$dt;
    $time_val[$jj] = $t_conv*$M*$raw_time[$jj]/$MADM;
    $hplus_val[$jj] = $mode2pol*(cos(2*$pol)*$hRe[$jj]+sin(2*$pol)*$hIm[$jj])*(1 + (cos($iota))*(cos($iota)))/2;
$hcross_val[$jj] = mode2pol*(-
\sin(2*$$\text{pol}$$)*$hRe[$jj]+cos(2*$$\text{pol}$$)*$hIm[$jj])\times\cos($$\iota$$);

# print(OUT_FILE $tnow," ",hplus_val[$jj],"
","hcross_val[$jj],"
";
printf(OUT_FILE "%.8e\t%.8e\t%.8e\n",$tnow,hplus_val[$jj],hcross_val[$jj]);
}
close(OUT_FILE);