Computational Materials Science

{Final Project}

David Witman
Department of Scientific Computing

1 Introduction

This project is intended to explore and examine the ideas presented in a recent paper in the field of computational material science. Specifically we hope to analyze and attempt to implement the methods presented in the paper entitled: Time Series Analysis of Molecular Dynamics Simulation using Wavelets by Mikito Toda. The object of this paper is to use a combination of wavelet compression techniques coupled with the singular value decomposition to generate a reduced order approximation of a molecular dynamics simulation. The first two sections will present a method for designing a molecular dynamics simulation along with the algorithms used as well as some additional caveats. The third section of this work will discuss the methods presented in the paper as well as the reasoning including comments on the presented work. The final section will present the results of our molecular dynamics simulation and attempt to implement the concepts presented in the paper we wish to analyze.

2 Lennard Jones

In any particle simulation, one must define how the particles interact with one another in terms of the forces that they exert on one another. For large body simulations, Newton's gravitational equations are typically used where the force from each particle interaction is dependent on the inverse of the squared distance from one another and a gravitational constant. Unfortunately this model is not very good at approximating small neutral particles and if we wish to consider quantum effects. In order to circumvent these shortcomings, and provide a more robust method for modeling small particles Lennard-Jones Potential is commonly used.

The Lennard-Jone potential is well known and often used mechanism for determining the forces of particles within a system on one another. Additionally due to the simplicity of the method, it is also used in many computational simulations. There is a plethora of existing software dedicated to modeling particle interactions within a system yet for the purposes of this project we will attempt to implement our own Lennard-Jones simulation. The most common representation of the Lennard-Jones potential is in its potential energy form given in equation 1

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \tag{1}$$

where ϵ is the depth of the potential well that the particle would be sitting in, σ is the diameter of the set of particles themselves and r_{ij} is the radial distance between each particle respectively. One thing that we notice from this equation is that the r_{ij} term accounts for a set of two particles and not just a single particle. This means that the pairwise interaction between the two particles

is going to have the same amount of energy applied to each one. Thus we must be careful to implement an efficient algorithm such that the forces that act on each particle are not computed twice.

Using the definition of the potential energy given by the Lennard-Jones potential in equation 1 we can define the force acting on a set of particles as:

$$F(r_{ij}) = \frac{24\epsilon}{\sigma^2} r_{ij} \left[2 \left(\frac{\sigma}{r_{ij}} \right)^{14} - \left(\frac{\sigma}{r_{ij}} \right)^{8} \right] \tag{2}$$

from this equation we can see the dilemma in computing the force on every pair of particles. The first thought would be to include a double loop over all the particles which would require $\mathcal{O}(N^2)$ computational cost. But if we inspect the equations a little closer we can see that this method would induce unnecessary computation. To avoid this problem we can modify the inner loop of our force calculation to only compute the force over particles that have yet to be computed, this incurs only a $\mathcal{O}(\frac{N(N-1)}{2})$ computational cost. It is important to note that this new implementation is still $\mathcal{O}(N^2)$ except there is now a sizeable pre-factor in front of of the N^2. We can visualize this implementation using the pseudo-code provided below:

```
for iF=1:N
    for jF=iF+1:N
        Keep running sum of the force on the particle
```

With this algorithm we can more efficiently calculate the force acting on each of the particles within our system.

3 Velcity Verlet

Now that we understand how to compute the force on each of the particles within our system,the question becomes: how do we determine the positions and velocities of each of our particles within the system. One of the most commonly used methods in the area of computational material science is known as the Verlet algorithm. The Verlet algorithm seeks to compute the velocities and positions of each of the particles within a system. To compute the positions, one can use the form:

$$r(t + \Delta t) = 2r(t) - r(t - \Delta t) + \frac{F(t)}{m} \Delta t^2 + \mathcal{O}(\Delta t^4) \tag{3}$$

where r and v represent the position and velocity information respectively and m defines the mass of the particle. The velocity component can be calculated with the equation:

$$v(t) = \frac{r(t + \Delta t) - r(t - \Delta t)}{2\Delta t} + \mathcal{O}(\Delta t^2) \tag{4}$$

These equations for the positions and velocity components of our particles have a number of benefits but some major drawbacks as well. The benefits include numerical accuracy and stability, which is paramount in a computational simulation. Also the energy and momentum within the system is relatively conserved which is good for the practicality of this simulation. Unfortunately though, this method requires a large amount of data from each time step to be stored. For the case of the velocity calculation, we see that positional information from the past time step and the next

time step are required which means this information must be stored in memory. To get around this problem we can introduce the Velocity Verlet method.

The Velocity Verlet method seeks to decrease the amount of memory that must be stored in the computation of the positions and velocities, while maintaining the accuracy of the standard Verlet method. The positional equation for the Velocity Verlet method can be given by the form:

$$r(t + \Delta t) = r(t) + v(t)\Delta t + \frac{1}{2m}F(\Delta t)\Delta t^2 \tag{5}$$

and the velocity equation:

$$v(t + \Delta t) = v(t) + \frac{F(t) + F(t + \Delta t)}{2m}\Delta t \tag{6}$$

At first the velocity equation looks as if we are going to need to store the force information for multiple time steps. But with some closer inspection, we can see that the force term used to calculate the velocity can be split apart and calculated in different steps of the algorithm. The idea being that we calculate all the forces acting on a specify particle, use them to calculate the new position of said particle and the first half of the new velocity. Then we compute the forces at the new positions so we can apply this final piece to determine the velocities at the new positions. Using this algorithm we do not need to store as much information and we can maintain the accuracy and stability of the Verlet method. It can also be shown through some algebraic manipulation that this Velocity Verlet method is in fact the same as the standard Verlet method.

4 Paper Analysis

The paper we wish to analyze, Time Series Analysis of Molecular Dynamics Simulation using Wavelets, presents the concepts for applying a wavelet compression technique coupled with the singular value decomposition to create a reduced order approximation of a molecular dynamics simulation. The idea behind coupling these two methods is to account for the temporal oscillatory nature of the particles using wavelet compression and then use the singular value decomposition to reduce the degrees of freedom in the system. The object of this paper is to apply this method to protein atom movement within a system. The proteins being analyzed in this paper are Adenylate Kinase (ADK) from Escherichia coli and Thermomyces lanuginosa lipase (TLL). ADK is used as an enzyme that catalyzes the interconversion of adenine nucleotides and is an protein for cellular energy homeostasis. Homeostasis is a mechanism that allows a property to be regulated in order to preserve homogeneity. Unfortunately this authors knowledge of proteins is very limited so there will not really be any more discussion on the properties and uses of the proteins in this study. Instead the focus will be on the computational methods of the work.

4.1 Morlet Wavelet

After giving a brief explanation on the motivation behind the study along with some further explanation of the proteins being used, the author then moves into the explanation of using the wavelet transform to decompose the positional data into a set of wavelet coefficients. The paper explains how the wavelet transform can be thought of as a windowed Fourier transform because wavelets are not infinitely defined unlike the sine and cosine terms in a Fourier transform. Also the paper notes that there is a large number of discrete and continuous wavelets that have been proposed and

analyzed in a wide area of research. This paper will specifically focus on the Morlet wavelet which is a continuous wavelet that can be a real or a complex valued function. The nice thing about the Morlet wavelet is that it is not defined by a recurrence relation like the Daubechies family of wavelets, meaning a specific value can be mapped directly to the Morlet wavelet. An example Morlet wavelet is provided in Figure 3 and has support over the domain $[-4, 4]$.

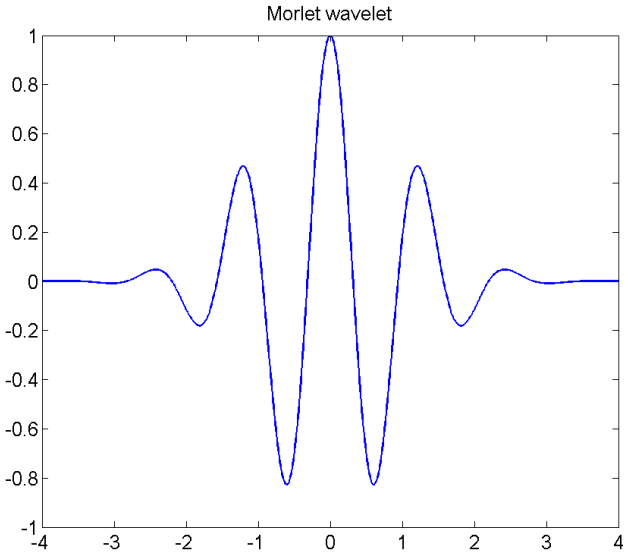

Figure 1: Morlet Wavelet

The paper then goes on to define the Morlet transform in terms of a function $f(t)$ which can be sampled at $f(s)$. Using the definition of the function along with the time t, frequency ω and the window $\frac{2\pi\sigma}{\omega}$, the Morlet wavelet transform $\hat{f}(t, \omega)$ can be defined:

$$\hat{f}(t,\omega) = \left(\frac{2\omega^2}{\sigma^2\pi^2}\right)^{\frac{1}{4}} \int_{-\infty}^{\infty} ds f(s) \exp\left(-i\omega(s-t) - \frac{\omega^2}{\sigma^2\pi^2}(s-t)^2\right) \tag{7}$$

4.2 Singular Value Decomposition

Once the wavelet transform has been defined, the next step in the paper we are analyzing is the singular value decomposition (SVD). The idea of the singular value decomposition is to reduce the degrees of freedom of the system in the hopes that the system could be operated on and then reconstructed with minimal effort. From a computational scientists viewpoint, the singular value decomposition is an invaluable tool to have at your disposal. The SVD can be defined by the relation:

$$A = USV^T \tag{8}$$

where A is the matrix that one would like to decompose, U and V are unitary matrices that represent the column and row space of A respectively and S is a diagonal matrix containing the singular values. The singular values can be thought of as the importance of each column/row in the U and V matrices. When one wishes to form a reduced order model, the hope is that the majority of the information contained within A can be represented in only a few singular values. This way minimal information is needed to construct a reduced order solution.

After explaining these two concepts, the paper then turns its attention to using these methods to generate a reduced order approximation to a molecular dynamics simulation. The first step is to decompose the time series approximations into wavelet coefficients using the Morlet wavelet decomposition. The paper defines a time series as $q_n(t)$ where n is the number of degrees of freedom. So in the case of our molecular dynamics simulation that we would like to construct, the degrees of freedom is going to be dependent on the number of particles we have in the system. Then we can form the decomposed time series $\hat{q_n}(t_i, \omega_l)$ at the discrete time intervals t_i. We also wish to perform the decomposition with a number of different frequencies in order to capture the oscillatory nature of the particle so we can define a number of different frequencies ω_l to test for. Using $\hat{q_n}(t_i, \omega_l)$ we now have all we need to construct the A matrix to be used in our singular value decomposition. The paper instructs us to assemble our A matrix at a discrete time step $A(t_i)$ as $\{A_{n,l}(t_i)\}_{n,l}$. This means that the number of particles in the system determine the number of rows of A while the test frequencies define the columns. With this definition of A, the paper believes that the oscillatory nature of the particles in a system can be adequately represented using only a few degrees of freedom.

4.3 Comments

Apart from being at times difficult to read, this paper is relatively concise in explaining its methodology to one that is already familiar with the concepts of wavelet's and the singular value decomposition. For one not familiar with either of these topics, this could be a very difficult paper to understand due to its brevity in explaining difficult computational science topics. Additionally this paper could benefit dramatically by including an example problem with more plots and figures intended to illustrate its point. But in the end this is really just an exploratory theoretical paper intended to propose a new methodology that could be used in future research applications.

5 Implementation and Results

Now that we have read the paper: Time Series Analysis of Molecular Dynamics Simulation using Wavelets by Mikito Toda and understood the process of creating a reduced order approximation to a molecular dynamics simulation, we hope to be able to implement this new proposed method. Due to time restrictions we will not be able perform much post processing on the reduced order model, like reconstructing a solution. There is also very little contained within the paper regarding how one would actually reconstruct the reduced order simulation. So instead we will focus on constructing our molecular dynamics simulation and then building the reduced order model using the techniques defined in the paper. To do this we must first we must construct the Molecular Dynamics simulation with the Lennard-Jones Potential embedded in a Velocity Verlet Algorithm. Then using the results from this simulation we will build an A matrix at every time step with the wavelet transform. Finally using the singular value decomposition we will reduce the degrees of freedom that the system depends on.

5.1 About the Molecular Dynamics Algorithm

As mentioned, we will be using a Lennard-Jones Potential embedded in a Velocity Verlet algorithm to construct our molecular dynamics simulation. Following the general outline of the Velocity Verlet algorithm was straight-forward but there were a few difficulties. One of the biggest difficulties was the implementation of periodic boundary conditions which required us to consider particles near the boundaries within 2.5σ. Additionally because we need to store positional information for each of the particles at each time step for use in the wavelet transform, we had to use a three-dimensional matrix to define each of our (x, y) coordinates at each time step t_i. One last thing that is worth mentioning is that the initial velocities and positions of the particles were generated randomly within a given domain. This means that in all likelihood the system will not reach an equilibrium state as the particles will continue upon their trajectories until influenced by other particles.

5.2 About the Code

The code is written in MATLAB and arranged into three parts:

1. `Verlet.m`: contains all the molecular dynamics simulation with the velocity Verlet algorithm and Lennard-Jones Potential
2. `pbc.m`: takes as input two points and decides whether they are close enough to be considered in the force calculation. The periodic boundary conditions are also handled here
3. `Decompose.m`: Runs through the wavelet transform and singular value decomposition as defined in the paper.

5.3 Results

The majority of the time spent on this project was put into getting the molecular dynamics simulation working with more than two particles. But after some struggles this author believes that the code is now an acceptable implementation of the velocity Verlet algorithm using the Lennard-Jones potential. To verify whether the simulation was working a system of two particles was observed under the following conditions:

- $\Delta t = 0.01$
- $T = 4$
- $\sigma = 0.05$
- Particles of mass $m = 1$ placed randomly in a square domain of size 1
- Initial velocities randomly selected between $[0, 0.005]$

With this set of parameters, we can track the particles over time shown in Figure 2

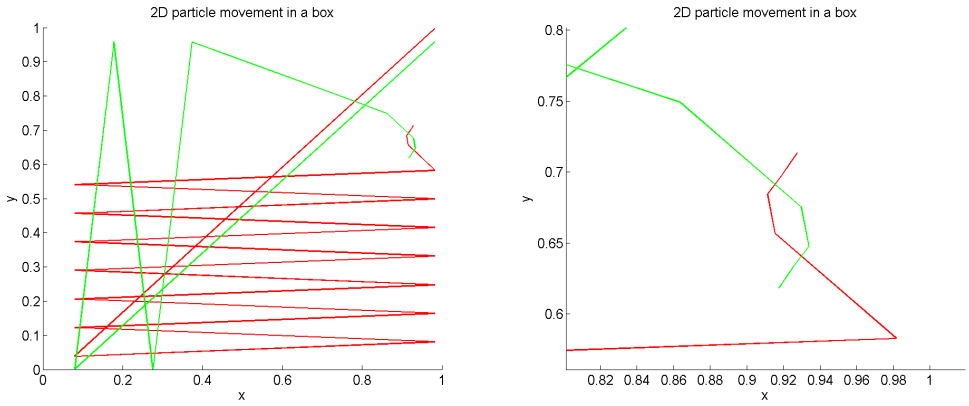

Figure 2: Two particle system in a box with periodic boundary conditions

Figure 2 also contains a zoomed in figure where we see the two particles clearly acting on each other at the end of the simulation. It appears that the particles may be close enough initially but due to the random positions and trajectories they must be far enough away to not affect one another. Using this simulation as a guide we now have some confidence in our molecular dynamics simulation.

The next step is to consider a system with more particles occupying the same domain and under the same conditions. So if we increase our system size to 20 particles, our plot becomes a little more cluttered:

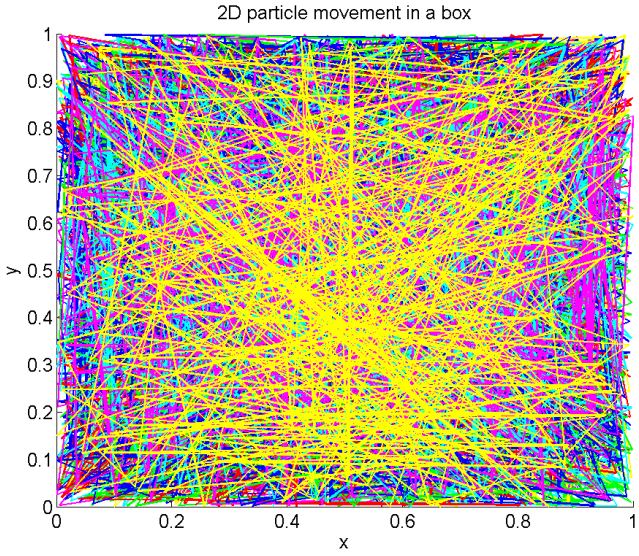

Figure 3: 20 particle system

From this plot we see a jumbled mess of information which is to be expected. Since we now have ten times the number of particles in our box, the paths that each of them follow is difficult to keep track of although we should note a few things from this plot. First we see that there are a number of particles that appear to just jump back and forth across the box. This is due to the periodic boundary conditions that we have imposed because if a particle leaves our box it must enter the box on the other side with corresponding velocity components. Also we notice that the particles appear to zoom quickly across the domain, this is most likely due to the large mass compared to the size of the particles and size of our system. More massive particles mean greater forces and increased velocity after being influenced by a new particle. Because this is a study of the computational methods and not the application we note that some of the parameter values may be unrealistic.

Now that the difficult part of building the molecular dynamics simulation is done, all that is left to do is the wavelet transform and singular value decomposition. Using the methods discussed in this study based of the ones proposed in the paper we analyzed we can perform the Morlet wavelet transform on our particle positions. The built in MATLAB function, `cwt` is used to generate the continuous wavelet coefficients using a logarithmically generated vector of frequency components from 10^{-2} to 10^{1}. Once we have generated these wavelet coefficients we can plot them with respect to the particle that they are trying to represent:

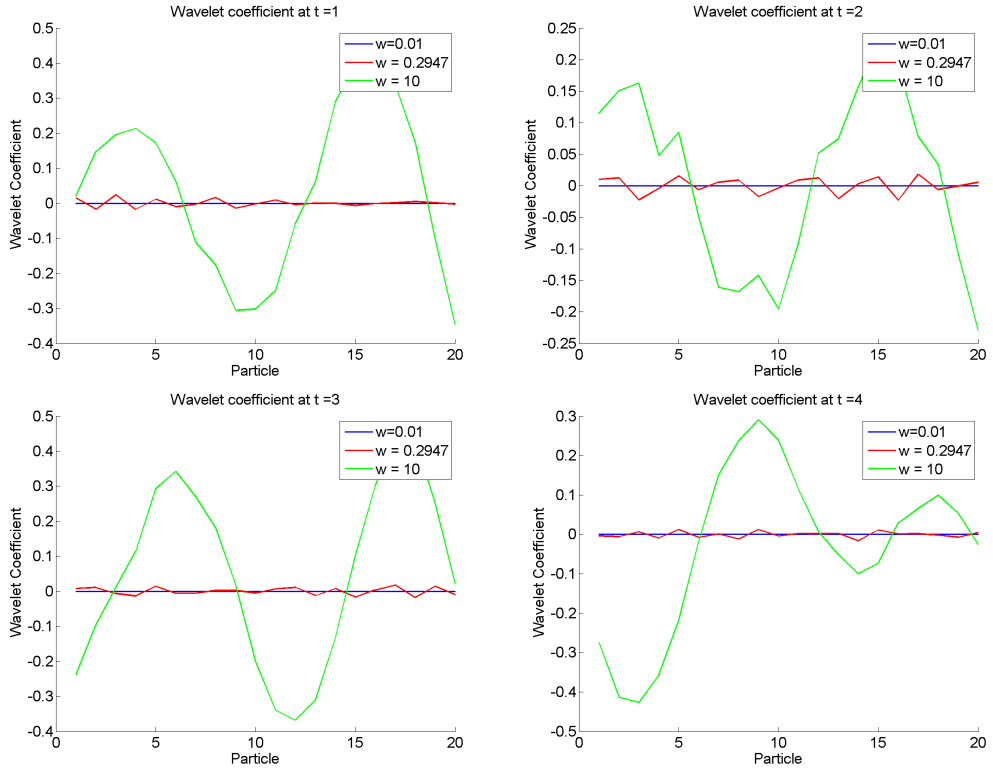

Figure 4: Wavelet Coefficients

Once the position data has been transformed into the wavelet coefficients we can perform the singular value decomposition on the data to extract the singular values and the two unitary matrices. Figure 5 displays the singular values for the four time instances we are analyzing. Figure 6 shows the reduced wavelet coefficients with respect to the particles and Figure 7 shows the reduced wavelet coefficients with respect to the frequency wavelet they are analyzed with.

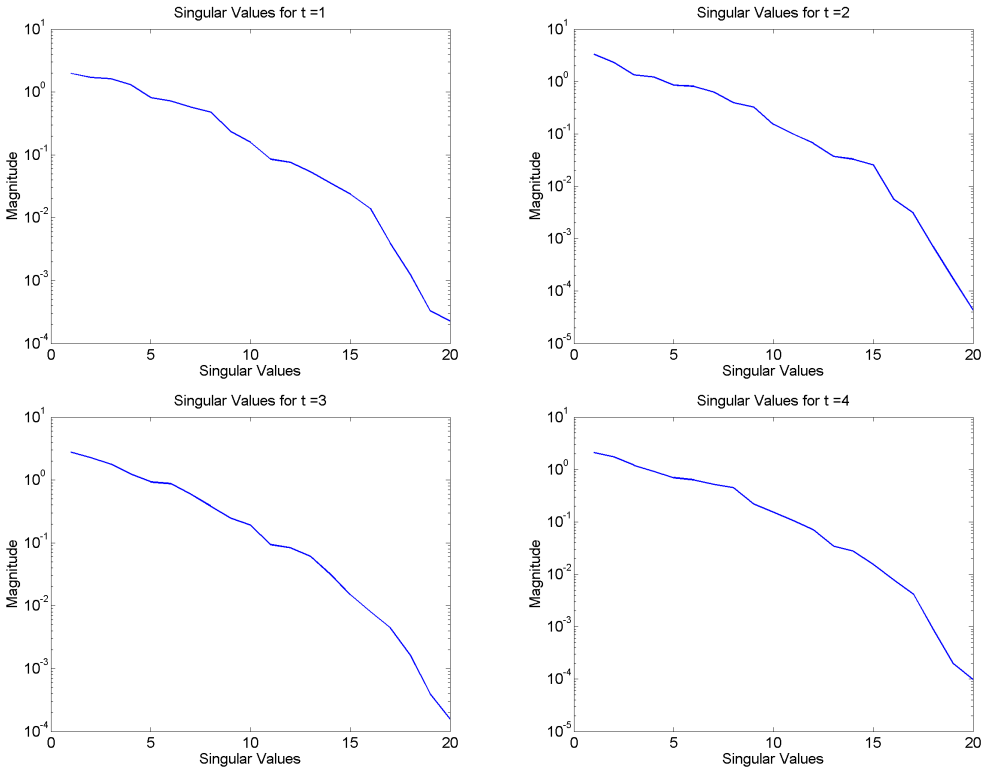

Figure 5: Singular Values

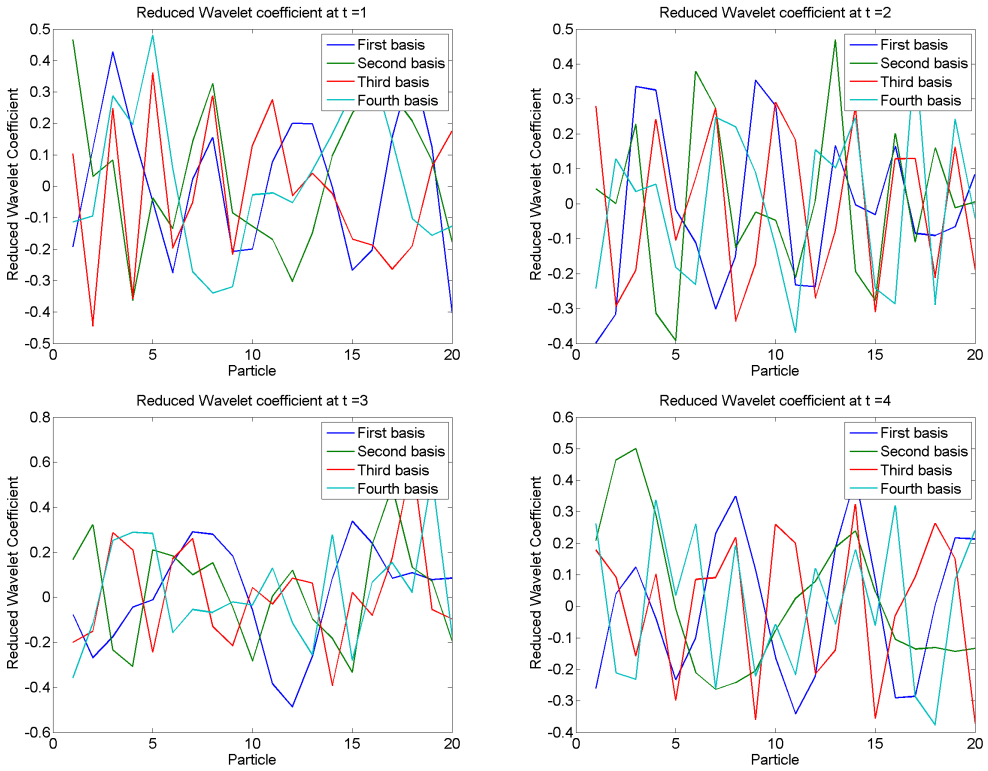

Figure 6: Reduced Basis with respect to the particle

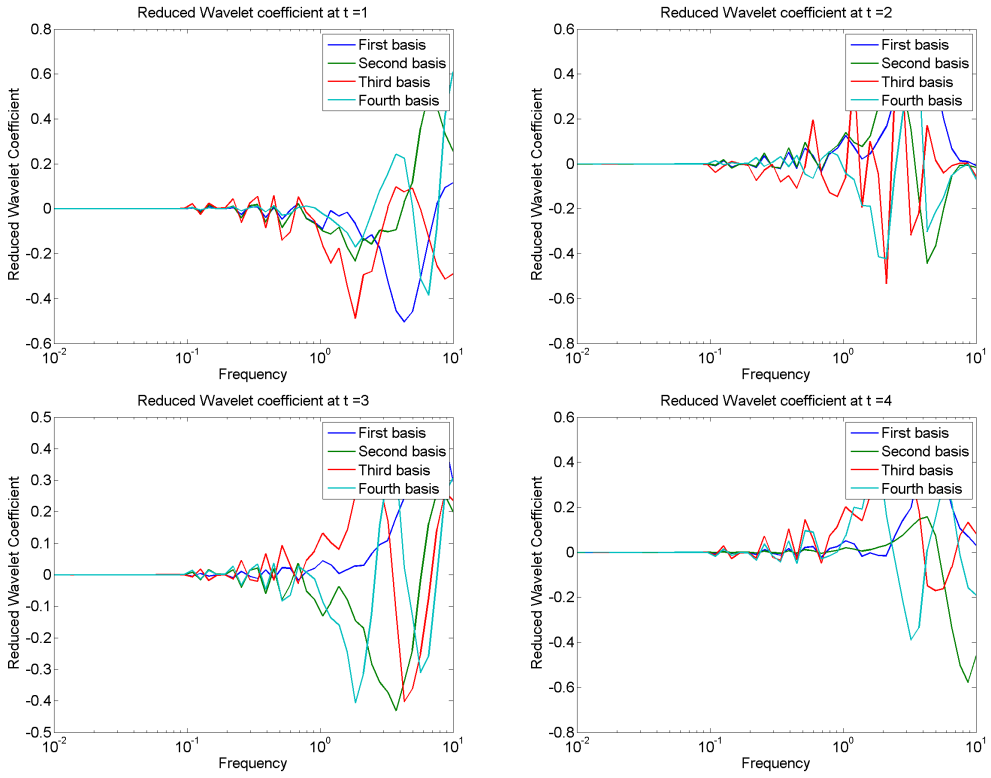

Figure 7: Reduced Basis with respect to the frequency

From this set of plots we can see that the singular values appear to decrease logarithmically as expected meaning that only the first few basis functions contain the majority of the information about the wavelet coefficients. In Figure 6 we see that the coefficients appear to be zero mean oscillatory with respect to the particles in the system. Finally in Figure 7 we see that the frequency appears to play a large effect on whether the wavelet coefficients are captured, which is understandable.

5.4 Final Comments

Work of this kind is very exploratory in nature and has really yet to be exposed to real practical problems. But until these methods are subjected to rigorous practical testing the best thing we, as computational scientists, can do is to continue to experiment with the tools available to us. There is a lot of future work that would need to be done with this sort of study but there certainly is some potential.