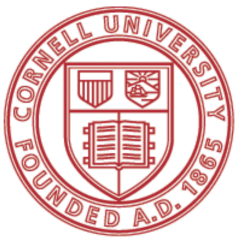
Masters of Engineering Project

# Using Fourier Series and Spherical Harmonics to Smooth a Beating Heart Model in Time and Space

Molly Drumm med275

May 18, 2024




Cornell University

# Abstract

This project smoothes models of beating embryonic hearts taken from ultrasound data over time and space in order to run simulations on the smooth models to study muscle strains and fluid flow inside the hearts as they develop. The ultrasound data has some noise as it beats and artifacts in space of the discrete nature in which the data was taken. The data is smoothed in time using a Fourier transform. This is done using the Matlab fft function. The high frequency modes are removed from the Fourier transform and the resulting movement vectors are smoothed over time. The data is smoothed in space using a spherical harmonic transform at a relatively low value of of order $l$, which approximates the model with less spherical basis functions to make a smoother file. However, the file contains the same number of nodes as the original which is good in order to map the evolution of the nodes over different days as the embryonic heart develops. It was found that a maximum $l$ value of 11 and a percent modes taken of 5 was sufficient to smooth the data in space and time. The code ran in a reasonable amount of time, about 25 minutes for the small sample data, and with an estimated 4 and a half hours for the type of data that will actually be used. The time smoothing worked exceptionally well and can be used in simulations. The space smoothing saw some distortion around the inlet and outlet which will need to be fixed in order to use that method to smooth models for simulation.

# Acknowledgements

Thanks to Gening Dong the PhD student I worked with for providing the data and letting me know the results, Jun Li another PhD student I worked with who found the uja_shfd code we used, and Professor Esmaily for recommending the use of Matlab fft and spherical harmonic transform and helping with the code, as well as being my project advisor.

# 1 Introduction

This project was completed in order to help Gening Dong with her PhD research, running muscle strain and fluid flow simulations to study the development of embryonic hearts. The motivation for this research is that heart defects in newborns develop in embryonic stages. If the development of the heart in these stages can be studied using CFD and muscle strains simulations, more can be learned about these defects. This can be done by creating a 3D model of the developing heart beating to run the simulations with. Chick hearts are used because they develop somewhat like human hearts and you can see into the eggs to image them easily. Stacked ultrasounds are used to make a 3D model, and they are made at different time points over the period which the heart beats. This data creates a model with a lot of high frequency noise in the movement over time, and some artifacts of the discrete data in space which look like stair shaped ridges across the surface. That is why using Fourier transform for smoothing the model in time and spherical harmonics for smoothing in space can help to get more accurate results from simulations.

The data is smoothed in space using a spherical harmonics representation of a lower maximum order $l$ to create a smoother surface. This was done using the function uja_shfd.m. The data is smoothed over time using a Fourier transform, which approximates a function using superposition of sine and cos waves. Then, the high noise frequencies are removed from the function, using Matlab Fast Fourier Transform (fft).

The given files to be smoothed are vtk files, which contain information about the coordinates of the vertices of the surface and the connectivity between them. Specifically, the data input to the code will be excel sheets of the coordinates from the vtk files with the connectivity information for these coordinates stored in the vtk files. The coordinates output from the code can then be converted back to vtk in order to view the animation of the smoothed beating of the heart. The sample data set being used had 56 time points, with a vtk file for each time point that, when played in sequence, was a 3D model of a heart beating. The vtk files have 2502 nodes, but the real data that Gening will be using has a number of nodes about an order of magnitude higher than this.

# 3 Literature Review

For the smoothing in time, Fourier transforms are a very generic way to filter out high frequency noise and smooth signals over time. The matlab fft function page itself had information on using Fourier transforms to do this.

What made this project different is that from the literature that was reviewed there was not a function that did this specifically for 3D files moving in time.

As for spherical harmonic smoothing in space, there was many articles with different methods for this, but eventually Jun found a great article (*Encoding Cortical Surface By Spherical Harmonics*) that already had a Matlab function that was able to complete a spherical harmonic transform on an obj file and return the smoothed coordinates in a freesurfer file. This source seemed like a good one to use because the researchers were performing the analysis on a biological model as well, this time of a brain, for their own research.

# 4 Theory

## 4.1 Fourier Transform

The Fourier transform, performed in this code by the Matlab function fft, approximates a function using superposition of sine and cos waves. In this case the function was each node's displacement from starting position, processed separately in x, y , and z. Euler's formula in Equation 1 shows how a complex exponential function can be used to represent cos and sin waves with frequency ω and time t.

$$e^{i\omega t} = cos(\omega t) + i\, sin(\omega t) \tag{1}$$

This is then summed in the Fourier series, as shown in Equation 2, where the Fourier coefficient u describes the weighting of each frequency sine and cosine function added together to approximate the original function, in this case the displacement of the node.

$$x(t) = \sum_{\omega} a_{\omega} e^{i\omega t} \tag{2}$$

Once the Fourier transform of the displacement vector is completed using Matlab fft, the lower frequency modes have significantly larger coefficients (and therefore a larger part in the description of the movement of the node) than the high frequency modes which contain the noise. By taking the vector of Fourier coefficients and setting the high frequency modes to zero, and then putting this new coefficient vector back through the Matlab inverse Fourier transform (ifft) function, the displacement over time is smoothed to only include the movement based on the lower modes.

## 4.2 Spherical Harmonics

Spherical harmonics are similar to Fourier transform but instead of smoothing over time in a single dimension they can smooth over space in two dimensions. For a surface function f(θ,$\phi$), the spherical harmonic approximation is equal to

$$f(\theta,\phi) \simeq \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} a_l^m Y_l^m(\theta,\phi) \tag{3}$$

where θ and $\phi$ are the polar and azimuth angles in a spherical coordinate system. $l$ is the order of the transform and m is the degree, which describes the number of basis functions to be computed for each order. The order and degrees are described in Figure 1 which shows the spherical functions that are summed with weighting coefficients to describe the surface.

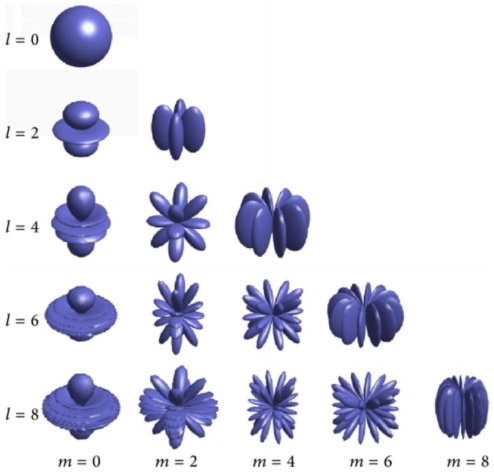



Figure 1: Spherical harmonic basis functions with increasing order l and different degrees m.

The spherical harmonic functions are defined by $Y_l^m$, with weighted coefficients $a_l^m$. The code uses an inverse spherical harmonic transform to get the weighting coefficients to describe the surface with a given number of orders $l$. As $l \to \infty$, the spherical harmonic representation becomes the exact surface. As we are trying to smooth out the surface, we use a lower $l$ to describe it and the surface becomes increasingly spherical and smooth.

The equations and images in this section were taken from the *Spherical Harmonics for Surface Parametrisation and Remeshing* article.

# 5 Design Criteria

## 5.1 Design Requirements

For the smoothing over time, the code needed to use Matlab fft and ifft to complete the Fourier transform. These functions can be difficult to use as they involve imaginary numbers in the resulting coefficients. The output of fft is a symmetric vector containing the real fourier coefficients on one side and their complex conjugates on the other. When smoothing the output of the fft function, in order to put the resulting smoothed coefficient vector back into the ifft function, it was essential the code output a symmetric coefficient vector.

The next requirement for the code is sorting the data effectively. There are thousands of nodes in each time step of the beating heart, and the model worked with had 56 time steps. This was a lot of data that needed to be handled to smooth the x, y, and z directions separately over time. To do this for loops were used which filled vectors with preallocated lengths in order to save time.

Finally, the Fourier transform code must take the amount of modes required for smooth movement without noise. The percent of modes taken is a user input that can be adjusted for the given set of mesh files.

As for the spherical harmonics code, the reason to use spherical harmonics rather than other smoothing methods is that this method retains the same number of nodes for the mesh. This is important so that when comparing beating heart models between different days of embryonic development, the same location can be easily compared over different days.

Another requirement is having a high enough order $l$ of the spherical harmonic approximation to capture the geometry well but low enough that the discrete artifacts do not show up and the surface is smooth. Too high of an $l$ value is not ideal because the computation time increases greatly with higher order as each order has a higher degree m, corresponding to a higher number of basis functions to be computed.

Overall, a requirement is that the code must be able to mesh well with the existing workflow that Gening is using. It must input and output the coordinates of the smoothed meshes at different time steps, with the connectivity matrix between these coordinates stored in the associated vtk files. Since Matlab fft was already being used, it was ideal for the entire project code to be able to run in a single Matlab script.

Finally, the code must run in a reasonable amount of time. Since the files are very large and the computation is involved, my hope is that for my sample data set, it can run in under 20 minutes, and for Gening's larger files it can run in under an hour.

## 5.2 Design Choices

As for choices in the design, the meshes were first looped over and smoothed in space, and then the resulting coordinates were looped over to smooth in time. For the time smoothing, nested for loops were used to loop through mesh data. The number of time points had to be input

as a user parameter as that could change for the given data. This lead to the first matrix of the coordinates created to have to change size each loop as each new time points data was added, because without knowing the number of time points its size could not be preallocated. This is not ideal for computing time but there was no other way found to do this.

The smoothing was also performed separately in x, y, and z, as if all three components are smoothed in time then their sum is also smoothed. This proved to be the most efficient way to correctly smooth the data.

Finally, for the test case, with some meddling and testing it was found that the ideal order $l$ was 11 for the model to be sufficiently smoothed in space, and the ideal percent of Fourier modes to take was 5 to be sufficiently smoothed in time.

# 6 Software

## 6.1 Matlab

Matlab was used to take inputs and run the spherical harmonic code. It then obtained the resulting coordinates from the space smoothing and sorted the data to run fft and ifft to smooth in time. Finally, it output excel files with the coordinate data.

## 6.2 Fortran

Fortran was used for converting the resulting coordinate file from matlab to vtk using a script Professor Esmaily wrote for me. The script took one sample unsmoothed vtk file, which has the same connectivity matrix as the smooth files should have, and put it, with the smoothed coordinates, into a new vtk file.

## 6.3 Paraview

Paraview was used to watch animations of the series of vtk file meshes moving over time. This could be used to judge qualitatively if the model was sufficiently smoothed in space and time, and also provided a great visual of the results.

# 7 Procedure

At the beginning of the code, the user picks the number of time points for the given data set, as well as the desired max order *l* for the spherical harmonics and percent of modes to take in the Fourier transform. The loops over the input files for the beginning of both the space and time smoothing sections need to be updated for the given data set, as they are currently set to accommodate the sample data set.

In the final code folder, there is a program provided to let the user run the time smoothing without the space smoothing. This is considerably faster and lets the user tweak the percent of modes taken setting to their liking.

The space smoothing can also be run for a single time step, and using the provided GUI with the code, the spherical harmonic representations can be visualized for increasing *l* values to adjust to the *l* value that works best. It is recommended to use a starting *l* value of 25 or less, as the default value of 40 has a much longer computing time. For the sample data set a *l* value of 11 was found to be sufficiently smoothed.

Once these parameters are set the user can run the file while in the folder with the uja_shfd.m code, which performs the spherical harmonic smoothing. The files must be converted to obj format in order to enter the uja_shfd code. The Matlab script loops through each of the time steps and smooths each file in space. It then goes into the resulting freesurfer files and get the coordinates of each smoothed time step to input into the Fourier transform.

The coordinates are put into a matrix set up along the row with the x, y, and z coordinates in order of time for each row representing each node. Then the rows of the matrix were looped through to get the x, y, and z vector over time for each node. These vectors were smoothed with the fft function by taking out the higher frequencies of the Fourier series. The center frequencies of the coordinate vector, the 95% highest frequencies from each side of the vector, were set to zero in order to keep the vector symmetrical. Only 5% of the coordinates were kept for the smooth low frequency modes used to represent the x, y, and z movement over time. The data was then converted into coordinates in a excel sheet format.

# 8 Results and Discussion

## 8.1 Smoothing in Space

At an *l* value of 11, the sample data was sufficiently smoothed and the general shape was retained. Figure 2 shows the spherical harmonic representations of the shape at increasing *l* values. Figure 3 shows a side by side of the original mesh and the final smoothed mesh set to an *l* value of 11.

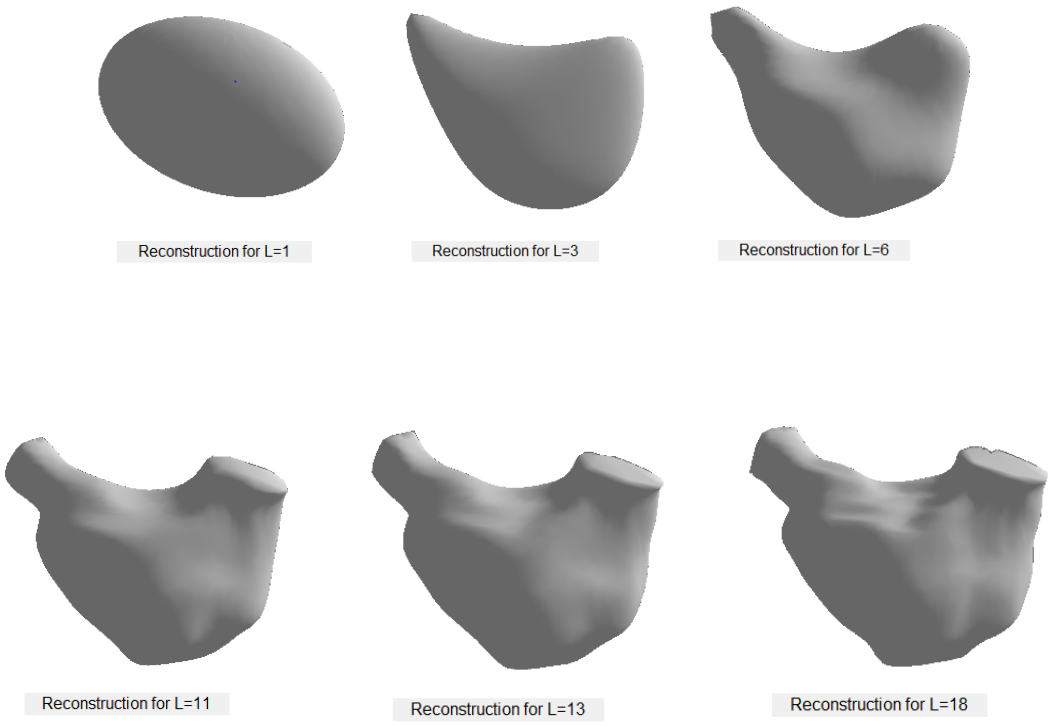
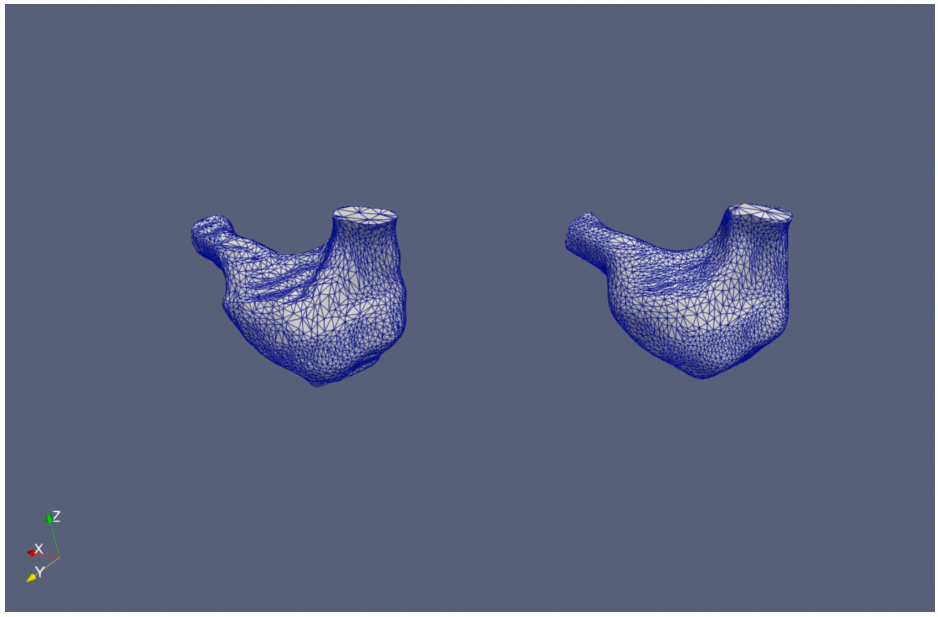



Figure 2: Spherical harmonic representation of model for increasing *l* value. 11 was chosen as the ideal smoothing while keeping the general shape for the model. By $l = 12$ and up the discrete data artifact ridges start to appear at the top of the model and by $l = 18$ they are very clear.

a.



b.

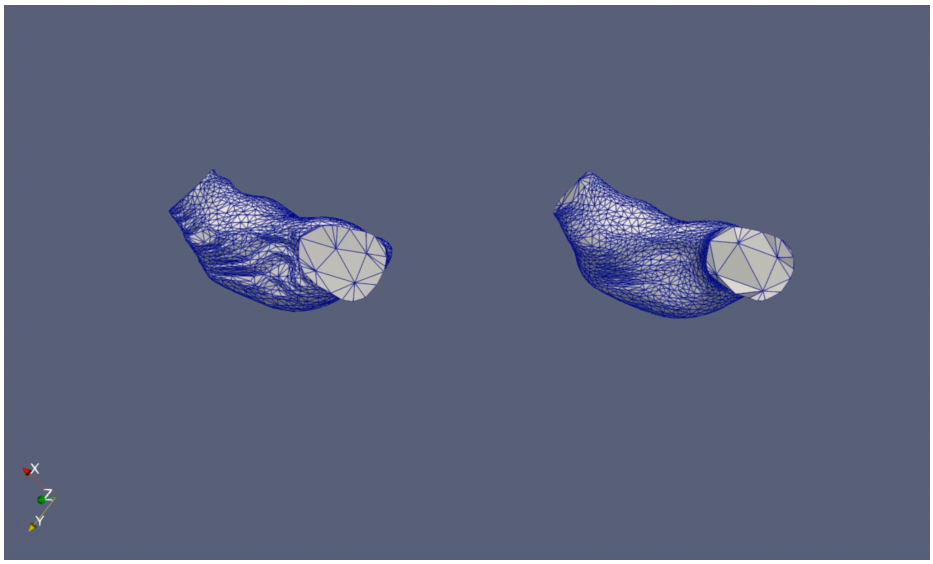


Figure 3: Original (left) and smoothed (right) meshes in space for a front (a) and top view (b).

The computing time for this operation is about 2 minutes when using the GUI for a single time step file. When looping over all the 56 files of sample data without using the GUI, it took about 25 minutes to run, which is ideal. For running a single time step of the larger files with over 25 thousand nodes, it took about 5 minutes to run. So if the code were to loop over about 50 time steps of a file that size, it would take about 4 hours which is a bit long but still within reason.

One failure of the code was the fact that the nodes near the inlet and outlet of the mesh were distorted poorly. The flat inlet and outlet surfaces are not actually part of the heart, they just represent where the flow goes in and out. Since they do not adhere to the natural spherical shape

of the heart, there is distortion at the nodes around the sharp edges along the inlet and outlet borders. This could possibly cause unphysical values in simulations run with these models.

## 8.2 Smoothing in Time

After running the time smoothing code, the model is sufficiently smoothed in time with 5 percent of the modes taken. Figure 4 shows the unfiltered vs. smooth movement of a single node in the z direction. Figure 5 shows the complex magnitude of the Fourier transform, showing the symmetrical relative magnitude of the Fourier coefficients for the different frequency modes.

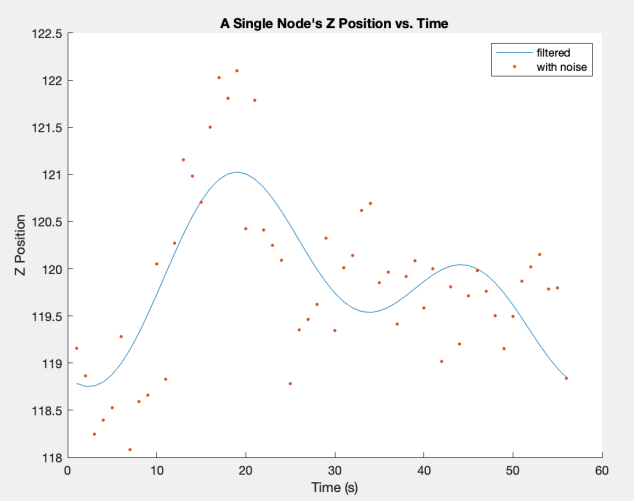



Figure 4: The unfiltered vs. smooth movement of a single node in the z direction. Though only the z direction is shown, the graphs for the x and y components look similar and summed together create smooth movement of the node.

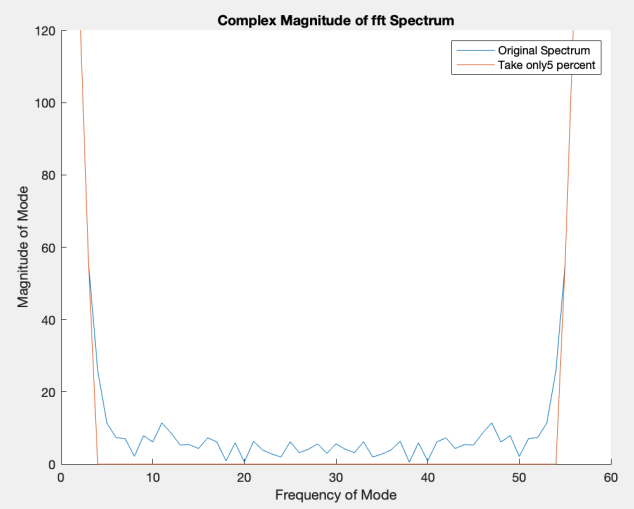



Figure 5: The complex magnitude of the Fourier transform, showing the symmetrical relative magnitude of the Fourier coefficients for the different frequency modes. The red line shows 5 percent of the modes taken with smoothing, with the rest of the coefficients being set to zero. It appears that two modes have been taken to describe the movement of the nodes.

The code takes only less than a minute to run for the sample data set. When Gening ran it for the much larger data, it took about 20 minutes, which is a great result.

Visualizing the time smoothing results can be difficult, so there is a video posted on Youtube at this link: https://youtu.be/l4rvc1FyW7g in order to watch an animation of the beating heart.

# 9 Conclusion

Overall, the model was smoothed in both space and time. The code worked within a reasonable amount of time. The smoothing in time could be used immediately to smooth models for simulation, it works very well. The smoothing in space had some distortion issues that need to be fixed before models smoothed with this method can be used in simulations.

However, there is still much future work that could be done. The code did not end up being implemented so that it could take simply the excel files of coordinates. It needs to take obj files of the meshes. If it would be possible to have Matlab convert the excel files to obj in the future, this would greatly improve the workflow.

The distortion around the inlet and outlet was also an issue. Future work could include trying to run the spherical harmonic transform on the wall only to see if the inlet and outlet surfaces will not affect the resulting mesh. However, it is unsure if you can run a spherical harmonic transform on a mesh that isn't a closed surface.

Finally, to track the evolution of the embryonic heart, the nodes could be mapped between days of testing for different models taken on different days.

# 10 Appendix

## 10.1 Read Me File

The smoothing code (project_final.m) uses a function called uja_shfd.m that is inside a folder called uja_shfd inside a bigger folder called uja_shfd_software that contains examples and directions for running the function. This must be run on a Windows computer.

To run the code, you need to first have your data as obj files in a folder called data_obj that must be inside the uja_shfd folder. The data_obj file currently inside the folder holds the example data set.

There are 3 user inputs in the code. p should be set to the number of time points in the data.

L should be set to the maximum spherical harmonic order to approximate the surface with. Lower values create smoother and smoother surfaces. To find the ideal L value, when in the uja_shfd folder you can type "uja_shfd" in the Matlab command line and a GUI will run where you can pick an obj file for a single time point and run over a certain amount of L values (default is 40, I recommend using less than 25 for less computing time). In the output, pressing the "SH reconstructions" button shows you the model at different L values so you can pick the ideal one.

b should be set to the percent of modes you wish to take for smoothing over time. The script fourier.m in my folder can be run on a set of csv files containig the coordinates at different time steps to smooth in just time and not space which is much faster. This can be used to tweak the amount of nodes taken. After using this, the output file coordinates.txt can be input into Professor Esmaily's code to convert the smoothed coordinates into vtk files. These files can then be viewed in Paraview to see if the time smoothing is sufficient.

The loops over the input files for the beginning of both the space and time smoothing sections need to be updated for the file names of the given data set, as they are currently set to accommodate the sample data set.

Once these parameters are set the user can run the file project_final.m while in the folder with the uja_shfd.m code. The Matlab script loops through each of the time steps and smooths each file in space. It then goes into the resulting

freesurfer files and get the coordinates of each smoothed time step to input into the Fourier transform.

## 10.2 Fourier Transform Matlab Code

```
clear all
%%%%INPUTS%%%%%%%%%%%%%%%%%%
%number of time points
p = 56;
%percent of frequncies in Fourier transform to keep
b = 5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%openning file and reading into a matrix
coords = [];
for j = 1:p
   str = string(j-1);
   if strlength(str) == 1
       nam = '00' + str;
   else
       nam = '0' + str; %loop over all the file names, taking into account some
start w 00 and some with 0
   end

   name = 'B6E8_D5_TEST_'+nam+'.csv';
   %read file
   m = readmatrix(name) ;
   m(:,1:2) = -1.*m(:,1:2); %make x and y not negative
   coords = [coords m];
end
%coords is set up like [(x1(t1),y1(t1),z1(t1)), (x1(t2),y1(t2),z1(t2))...;
%                       (x2(t1),y2(t1),z2(t1)), (x2(t2),y2(t2),z2(t2))...;...]
%every row is a different point, every three columns is x, y, and z at each
l = length(coords(1,:)); %how many time points * 3
q = length(coords(:,1));
s = ones(size(coords)); %to save smoothed coords in
for n = 1:q %loop over the nodes (rows)
   x = ones(1, p);
   y = ones(1, p);
   z = ones(1, p);
   for xs = 1:3:(l-2)
       x(((xs-1)/3)+1) = coords(n, xs); %make vector of all the x values at
each time point, do the same for y and z
   end
   for ys = 2:3:(l-1)
      y(((ys-2)/3)+1) = coords(n, ys);
   end
   for zs = 3:3:l
       z(zs/3) = coords(n, zs);
   end
```

```
   x_til = fft(x); %forier transform on each dimension
   y_til = fft(y);
   z_til = fft(z);
   mat = [x_til; y_til; z_til;]; %matrix of the fourier transforms on each
dimension
   matn = ones(size(mat)); %matrix of smoothed x y and z

   for pp = 1:3 %loop over x y and z
       fy = ones(1, p);
       c = ceil((p-1)*b/100);
       for w = 1:p
           if w > c && w <= ((p - c) + 1)
               fy(w) = 0;
           else
               fy(w) = mat(pp, w);
           end
       end
       matn(pp, :) = ifft(fy);
   end



   for a = 1:3:(l-2)
       s(n, a) = matn(1, (((a-1)/3)+1));
       s(n,a+1) = matn(2, (((a-1)/3)+1));
       s(n,a+2) = matn(3, (((a-1)/3)+1));
   end

end
%below is to graph the last nodes smoothed vs original coordinates, right
%now is set to z but you can also run on x or y
t = 1:p;
figure
hold on
plot(t,matn(1,:))
plot(t,x, '.')
legend('filtered', 'with noise')
title('A Single Node''s Z Position vs. Time')
xlabel('Time (s)')
ylabel('Z Position')
hold off
shg
%to graph complex magnitude of fourier spectrum, see how many modes you
%took
figure
hold on
plot(abs(z_til))
plot(abs(fy))
ylim([0 120])
```

```
title("Complex Magnitude of fft Spectrum")
xlabel("Frequency of Mode")
ylabel("Magnitude of Mode")
legend('Original Spectrum', strcat('Take only ', num2str(b), ' percent'))
%this plots displacement over time for a node r
% ds = []; %figure out how to preallocate
% for i = 1:length(s(:,1))
%     x0 = s(i, 1);
%     y0 = s(i, 2);
%     z0 = s(i, 3);
%     disps = []; %figure out how to preallocate
%     for k = 4:3:166 %this is bc I know there's 56 time points (so 56*3 is the
amount of coords, then -2 for the last x coord), in the future need to create
variable for how many time points there are
%         x = s(i,k);
%         y = s(i,k+1);
%         z = s(i,k+2);
%         disp = sqrt(sum([(x-x0)^2,(y-y0)^2,(z-z0)^2]));
%         disps = [disps disp];
%     end
%     ds = [ds; disps];
% end
%
% r = 1;
%
% g = ds(r,:);
% t = 1:(p-1); %one less than the number of time points bc there's number of
time points - 1 displacements between points
%
% figure
% plot(t, g)
% shg
%now save to txt file that can be read by fortran code and converted to a
%vtk file
%fortran doesnt like the long rows like I have it set up so the text file
%is set up like (if first number is node and second is time)
% x(1, 1) y(1,1) z(1,1)
% x(2, 1) y(2,1) z(2,1)
% ...all the nodes at time 1
% x(1, 2) y(1,2) z(1,2)
csvs = [q, p, 0]; %so that the first like is the number of nodes, number of
time points, and 0 for the dimension to be right (for fortran to read)
for ii = 1:3:(l - 2)
   csv = ones(q, 3); %for time point 1
   for jj = 1:q
       csv(jj,:) = s(jj,ii:(ii+2));
   end
   csvs = [csvs; csv];
end
```

```
writematrix(csvs, 'coordinates.txt')
```

## 10.3 Full Space and Time Smoothing Matlab Code

```
clear all
%%%%INPUTS%%%%%%%%%%%%%%%%%
%number of time points
p = 56;
%max L value to use for spherical harmonic smoothing
L = 11;
%percent of frequncies in Fourier transform to keep
b = 5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%spherical harmonic smoothing
for j = 1:p
   str = string(j-1);
   if strlength(str) == 1
       nam = '00' + str;
   else
       nam = '0' + str; %loop over all the file names, taking into account some
start w 00 and some with 0
   end

   org = pwd;
   name =  'data_obj/B6E8_D5_TEST_'+nam+'.obj';
   uja_shfd(name, 'max_L', num2str(L), 'min_L_reg_local', num2str(1),
'max_L_reg_local', num2str(L), 'min_L_reg_global', num2str(1),
'max_L_reg_global', num2str(L))
   cd(org);

end
%openning file and reading into a matrix
coords = []; %figure out how to preallocate size of this later?
for j = 1:p
   str = string(j-1);
   if strlength(str) == 1
       nam = '00' + str;
   else
       nam = '0' + str; %loop over all the file names, taking into account some
start w 00 and some with 0
   end

   name =  'B6E8_D5_TEST_'+nam+'.obj_temp';
   folder = 'data_obj/'+name+'/';
   %read file
   fil = 'template_B6E8_D5_TEST_'+nam+'_OL_2O_'+num2str(L)+'_0_des_orig.surf';
%change the 11 here if you use a higher L value
   path = fullfile(folder, fil);
   file = fopen(path, 'r');
   spec = '%f %f %f';
```

```
   data = textscan(file, spec, 'HeaderLines', 26);
   fclose(file);
   m = cell2mat(data);
   coords = [coords m];
end
%coords is set up like [(x1(t1),y1(t1),z1(t1)), (x1(t2),y1(t2),z1(t2))...;
%                       (x2(t1),y2(t1),z2(t1)), (x2(t2),y2(t2),z2(t2))...;...]
%every row is a different point, every three columns is x, y, and z at each
l = length(coords(1,:)); %how many time points * 3
q = length(coords(:,1));
s = ones(size(coords)); %to save smoothed coords in
for n = 1:q %loop over the nodes (rows)
   x = ones(1, p);
   y = ones(1, p);
   z = ones(1, p);
   for xs = 1:3:(l-2)
       x(((xs-1)/3)+1) = coords(n, xs); %make vector of all the x values at
each time point, do the same for y and z
   end
   for ys = 2:3:(l-1)
      y(((ys-2)/3)+1) = coords(n, ys);
   end
   for zs = 3:3:l
       z(zs/3) = coords(n, zs);
   end
   x_til = fft(x); %forier transform on each dimension
   y_til = fft(y);
   z_til = fft(z);
   mat = [x_til; y_til; z_til;]; %matrix of the fourier transforms on each
dimension
   matn = ones(size(mat)); %matrix of smoothed x y and z

   for pp = 1:3 %loop over x y and z
       fy = ones(1, p);
       c = ceil((p-1)*b/100);
       for w = 1:p
           if w > c && w <= ((p - c) + 1)
               fy(w) = 0;
           else
               fy(w) = mat(pp, w);
           end
       end
       matn(pp, :) = ifft(fy);
   end



   for a = 1:3:(l-2)
       s(n, a) = matn(1, (((a-1)/3)+1));
```

```
        s(n,a+1) = matn(2, (((a-1)/3)+1));
        s(n,a+2) = matn(3, (((a-1)/3)+1));
    end

end
%below is to graph the last nodes smoothed vs original coordinates, right
%now is set to z but you can also run on x or y
t = 1:p;
figure
hold on
plot(t,matn(1,:))
plot(t,x, '.')
legend('filtered', 'with noise')
hold off
shg
%to graph complex magnitude of fourier spectrum, see how many modes you
%took
figure
hold on
plot(abs(z_til))
plot(abs(fy))
ylim([0 120])
title("Complex Magnitude of fft Spectrum")
xlabel("Frequency of Mode")
ylabel("Magnitude of Mode")
legend('Original Spectrum', strcat('Take only ', num2str(b), ' percent'))
%this plots displacement over time for a node r
% ds = []; %figure out how to preallocate
% for i = 1:length(s(:,1))
%     x0 = s(i, 1);
%     y0 = s(i, 2);
%     z0 = s(i, 3);
%     disps = []; %figure out how to preallocate
%     for k = 4:3:166 %this is bc I know there's 56 time points (so 56*3 is the
amount of coords, then -2 for the last x coord), in the future need to create
variable for how many time points there are
%         x = s(i,k);
%         y = s(i,k+1);
%         z = s(i,k+2);
%         disp = sqrt(sum([(x-x0)^2,(y-y0)^2,(z-z0)^2]));
%         disps = [disps disp];
%     end
%     ds = [ds; disps];
% end
%
% r = 1;
%
% g = ds(r,:);
```

```
% t = 1:(p-1); %one less than the number of time points bc there's number of
time points - 1 displacements between points
%
% figure
% plot(t, g)
% shg


%now save to txt file that can be read by fortran code and converted to a
%vtk file
%fortran doesnt like the long rows like I have it set up so the text file
%is set up like (if first number is node and second is time)
% x(1, 1) y(1,1) z(1,1)
% x(2, 1) y(2,1) z(2,1)
% ...all the nodes at time 1
% x(1, 2) y(1,2) z(1,2)
csvs = [q, p, 0]; %so that the first like is the number of nodes, number of
time points, and 0 for the dimension to be right (for fortran to read)
for ii = 1:3:(l - 2)
   csv = ones(q, 3); %for time point 1
   for jj = 1:q
       csv(jj,:) = s(jj,ii:(ii+2));
   end
   csvs = [csvs; csv];
end
writematrix(csvs, 'coordinates.txt')
```

# 11 References

Matlab fft documentation https://www.mathworks.com/help/matlab/ref/fft.html#buuutyt-6

Dong, Gening, *Longitudinal Live Imaging Derived 5D Hemodynamics And Dynamic Tissue Strains Across Outflow Tract Morphogenesis*

Chung, M., Hartley, R., Dalton, K., and Davidson, R., *Encoding Cortical Surface By Spherical Harmonics* https://www3.stat.sinica.edu.tw/sstest/oldpdf/A18n43.pdf

Nortje, C., Ward, W., Neuman, B., and Bai, L. *Spherical Harmonics for Surface Parametrisation and Remeshing* https://www.hindawi.com/journals/mpe/2015/582870/