COSC 6364 – Advanced Numerical Analysis
Final Project- DERIV2D

2D Function Gradient and Directional Derivative

By: Hai-Y Michael Tran Nguyen

Peoplesoft: 0925358

Date: 5/10/18

Introduction

The goal of this project is to extract derivatives and gradients of multidimensional functions and compare them. Specifically, we will numerically calculate the directional derivative of a Function A and compare it to another Function B (which is already the derivative). In this project, we will be doing this on 2D functions.

Technologies Used

This project is split up into two separate parts that will work together to achieve the desired results.

C# (MS Visual Studio 2018)

Part of this project was developed in C# using Microsoft Visual Studio 2018. This part mainly deals with calculating the derivative and generating and outputting the raw data (.csv) used for comparison.

R (R Studio)

Another part of this project was developed in R using R Studio. This part mainly deals with using the raw data generated from the previous part (C#) and running comparisons against them and generating the graphs and plots in this report.

Function Graphs

The following are the raw 2D graphs for Function A, derivative of function A, and B.

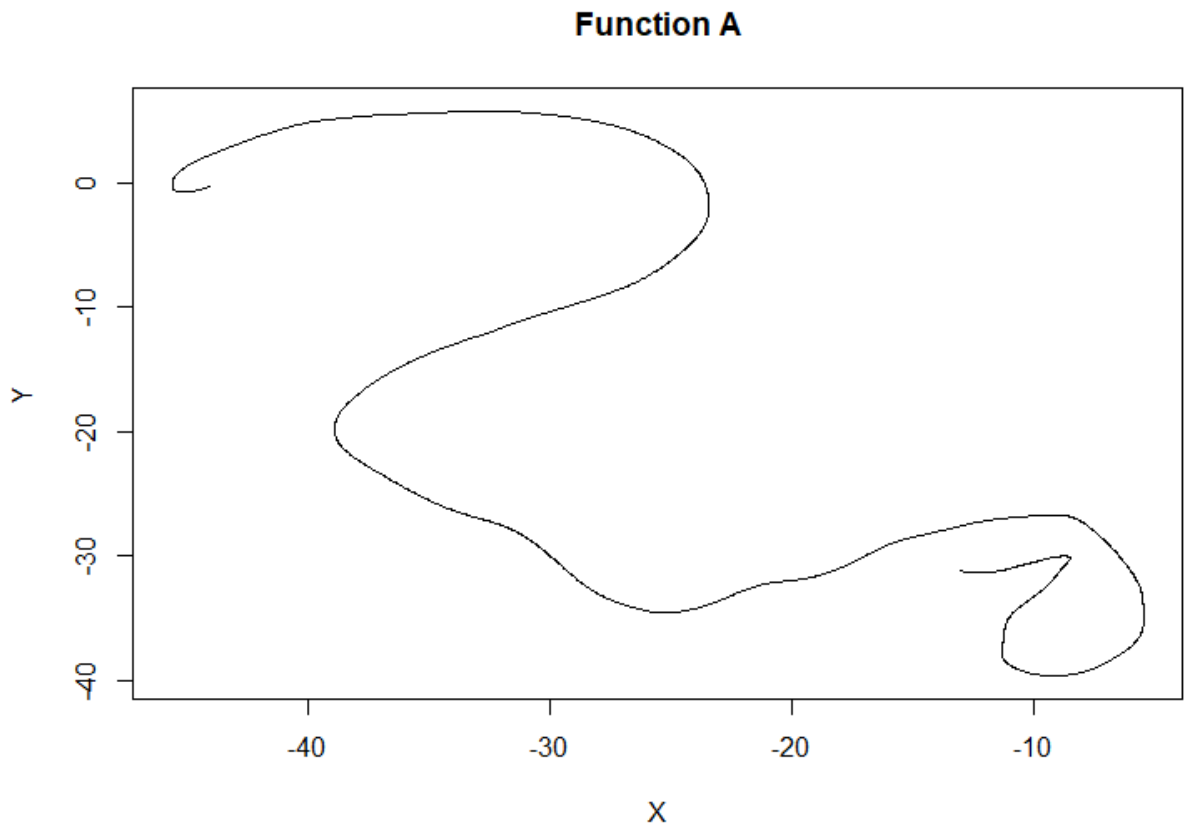

Figure 1 - Function A

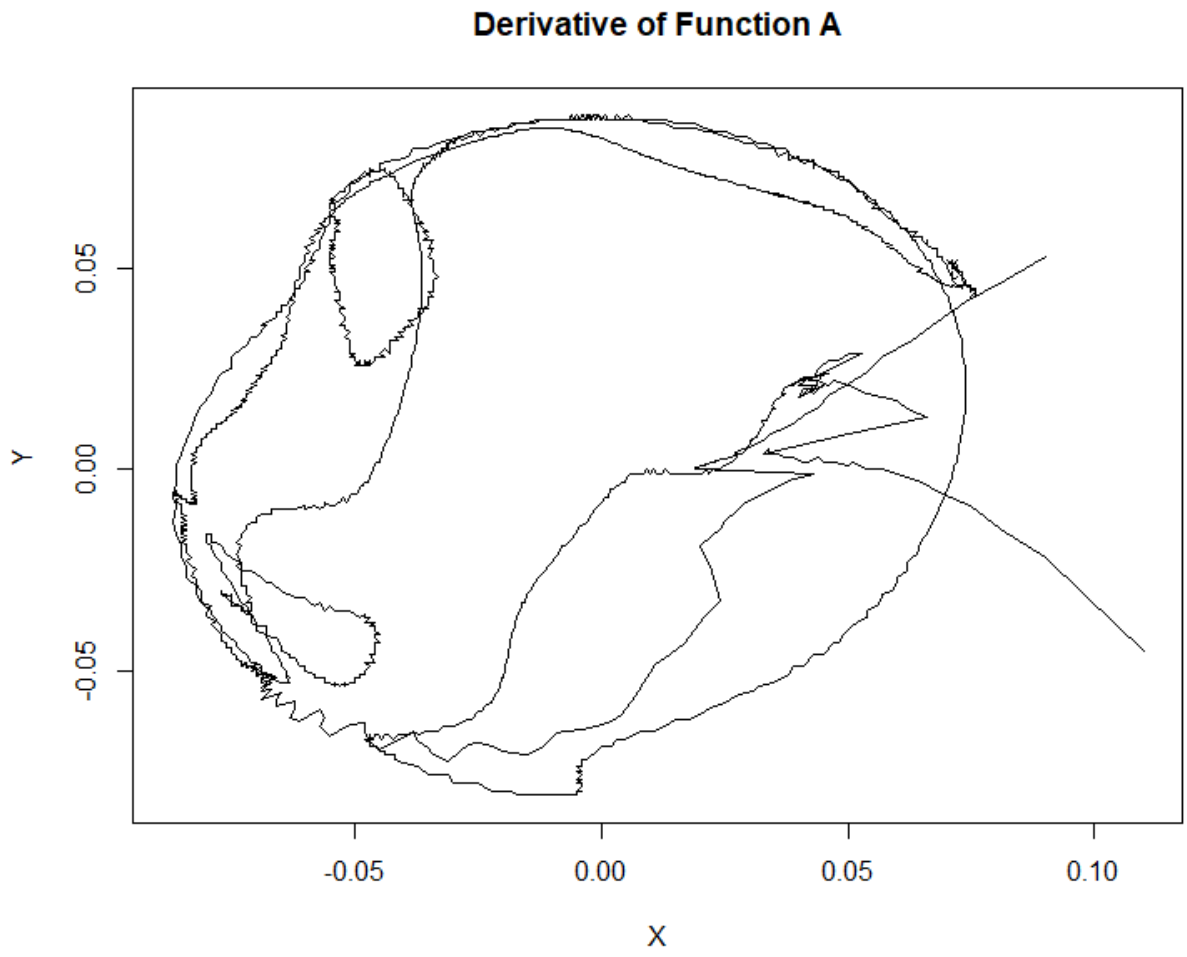

Figure 2 - Derivative of Function A

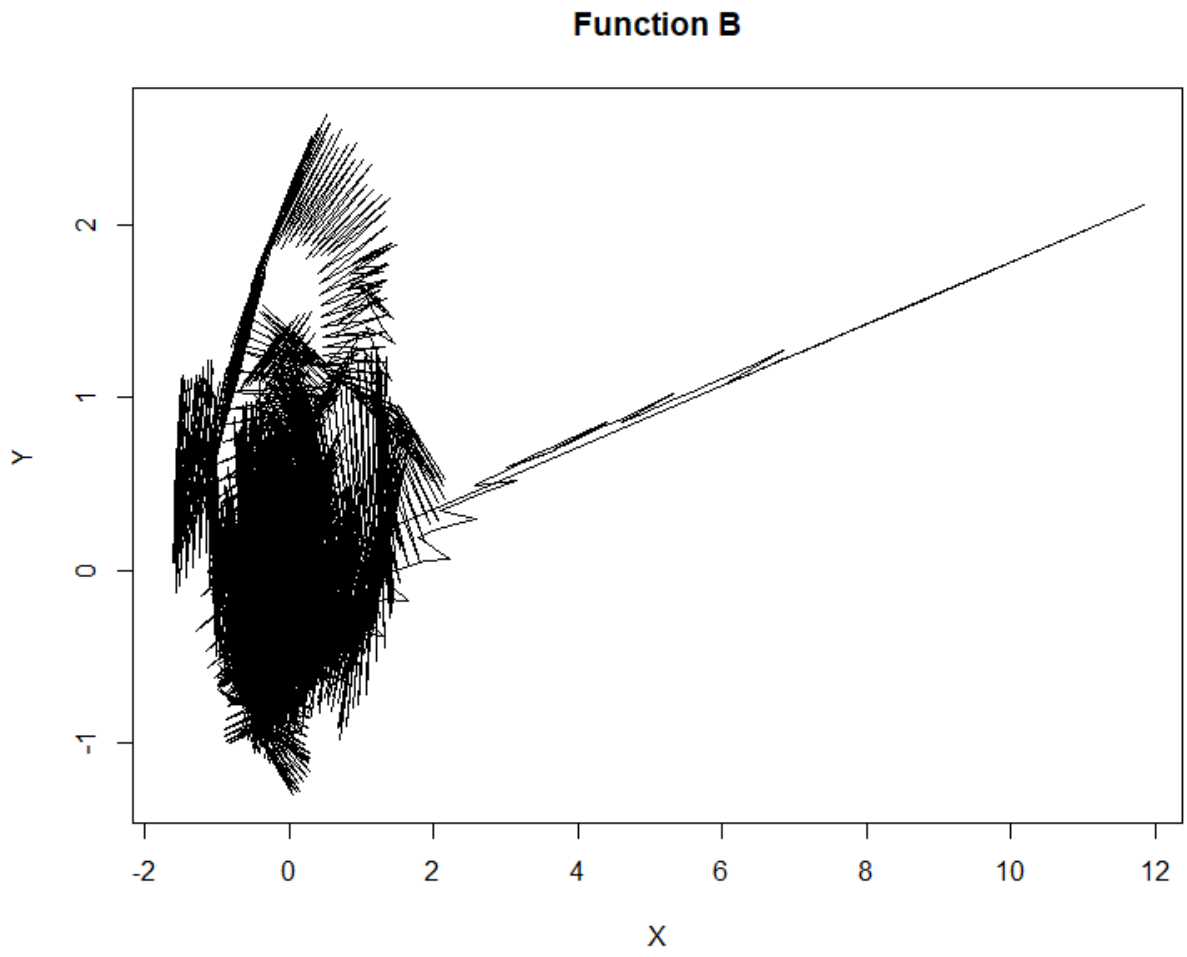

Figure 3 - Function B

As we can see from Figure 2 and Figure 3, they are radically difference from each other in not only the general shape, but also the scale of the X and Y values. From this, we can already predict that our derivative of Function A and Function B will potentially differ substantially. We will explore the potential reasons for this in later sections.

Methods

As stated in the previous section, this project is split up into two parts.

Part 1 (C#)

A brief summary of this part is that it will generate the derivative of function A and generate the raw comparison values.

Derivative Algorithm

The derivative of Function A will be calculated here.

It will loop through the total number of coordinates ($CountOfA - 1$) and compare the next point with the current point. ($NextPoint - CurrentPoint$)

After the loop is complete, we achieve the resultant derivative. Note that the total number of points decrease by 1 from Function A to derivative of Function A.

Normalization

Derivative of Function A and Function B normalization will be calculated here.

First, the absolute maximum of the derivative of Function A and Function B will be calculated. This is done by looping through each function and getting the absolute value of X and Y and then storing the highest value. After we have the absolute maximum value of both functions, we take the maximum of those two values to give us the overall absolute maximum.

We then normalize the values of X and Y in both functions by dividing each value by the overall absolute maximum value. This will convert and normalize the function data to a value between the interval [-1, 1].

Generating Comparison Data for derivative of Function A and Function B

In this project, I decided on comparing the data using 4 different techniques.

In general, all 4 methods follow the same process: number of steps in Function B per steps in Function A is determined. Then we loop through all points in Function A and we get the sum of however many points we take from Function B and then average that sum. ($\frac{SumOfPointsInB}{NumberOfPointsInB}$) This will result in the raw comparison data which we output to a .csv file for later use.

The underlying assumption for these methods is that all time steps in Function B and Function A are equal. Meaning that each coordinate and the next coordinate in either Function A or Function B take the same amount of time as any other coordinate in that same function with its next coordinate.

Static Steps Method 1 (SSM1)

For static steps method 1, we use the following formula to determine how many steps to take in Function B **per step** in Function A:

$$Floor(\frac{N_B}{N_A})$$

In this project, the number of steps in Function B is 5.

Note that this method does not use all points in Function B, and there will be some leftover unused points.

Static Steps Method 2 (SSM2)

For static steps method 2, we use the following formula to determine how many steps to take in Function B **per average of two steps** in Function A:

$$Floor(\frac{N_B}{N_A})$$

In this project, the number of steps in Function B is 5.

Note that this method does not use all points in Function B, and there will be some leftover unused points.

Dynamic Steps Method 1 (DSM1)

Dynamic step works slightly differently from static steps. It will dynamically choose a different number of steps based on how many steps have already occurred. This method will eliminate any leftover unused points and use all the points.

For example, in this project it will dynamically choose between 5 and 6 steps depending on how many steps have already occurred.

Dynamic step method 1 loops **per step** in Function A.

Dynamic Steps Method 2 (DSM2)

This method also uses the same method as DSM1.

Dynamic step method 2 loops **per average of two steps** in Function A.

Part 2 (R)

A brief summary of this part is that it will load and use the results of Part 1 and compare by-axis, and by magnitude and degree. It will also generate the resultant graphs for the "Results" section.

By-axis Differences and Comparisons

This part will generate comparison and differences graphs by the X and Y axis. The comparison graphs will plot both Function A and Function B per axis, by time. The differences graphs will plot the difference (e.g. $B_X - A_X$) between the two functions per axis, by time.

With these graphs, we will be able to see how the two functions compare and differ. The mean of differences per axis are also calculated.

By magnitude and degree

This part will generate magnitude comparison and difference graphs by time, and also degree comparison and difference graphs by time.

The magnitude for each function is calculated by using the following formula: $sqrt(X^2 + Y^2)$

The degree for each function is calculated by using the following formula: $\tan^{-1}(Y/X)$ and then converting from radians to degrees.

The mean of differences of magnitude and degree are also calculated.

Performance

The code performed quite fast. Part 1 and Part 2 took less than 1 second to perform. Part 1 code is quite efficient at how it handles the looping and calculations. Part 2 code shouldn't have any performance problems as there are no loops and mainly all it does is loads and generate the graphs.

Results

The following graphs are titled by the following format: {Type of Method} – {Type of Result}. Where Type of Method is one of the comparison techniques outlined in the previous section.

The raw outputs generated in Part 1 can be found the folder titled “R”. All Part 2 outputs are the figures.

SSM1

The final 637 coordinates in Function B were skipped in this comparison due to the method used.

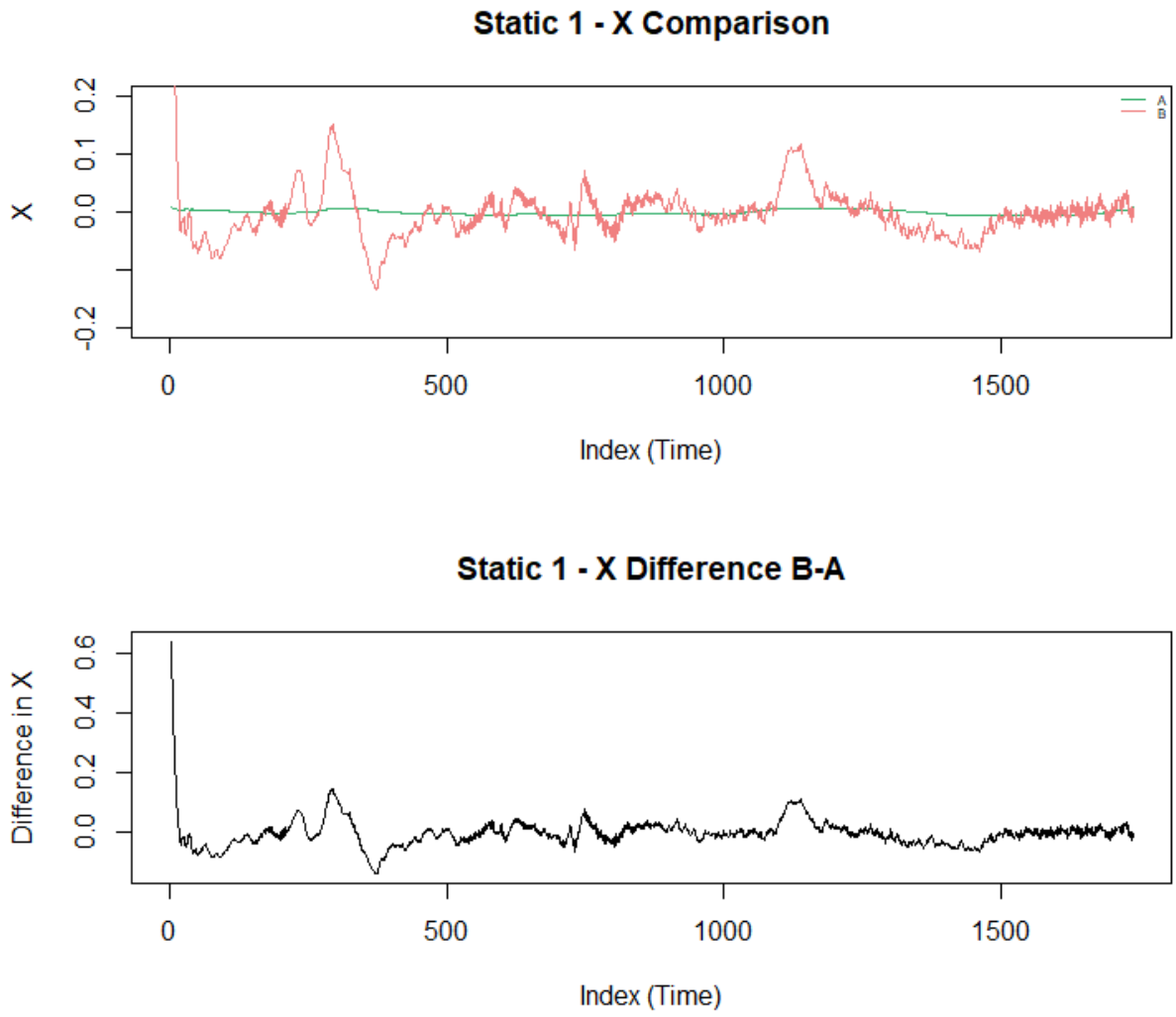

Figure 4 - Static 1 - X

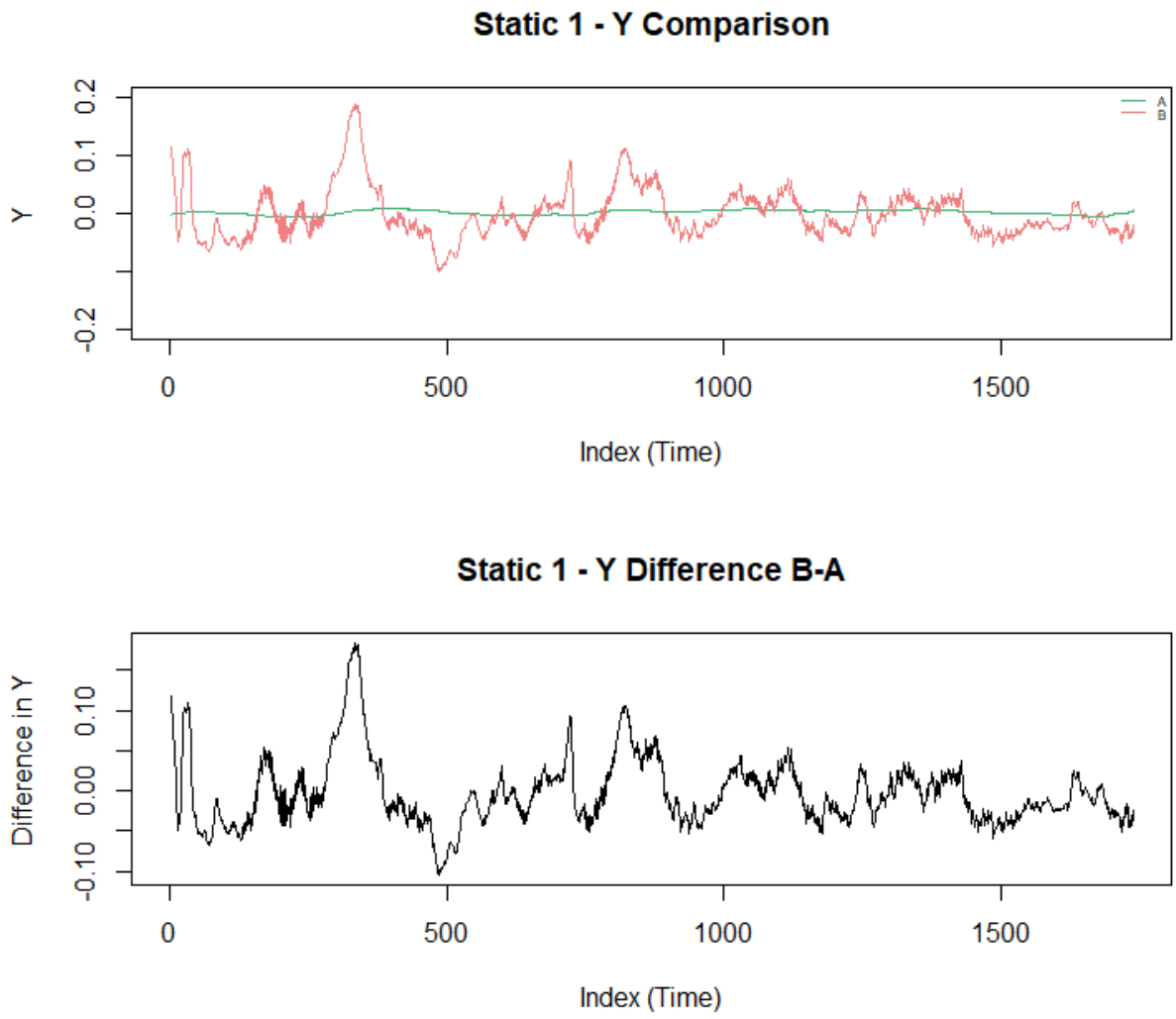

Figure 5 - Static 1 – Y

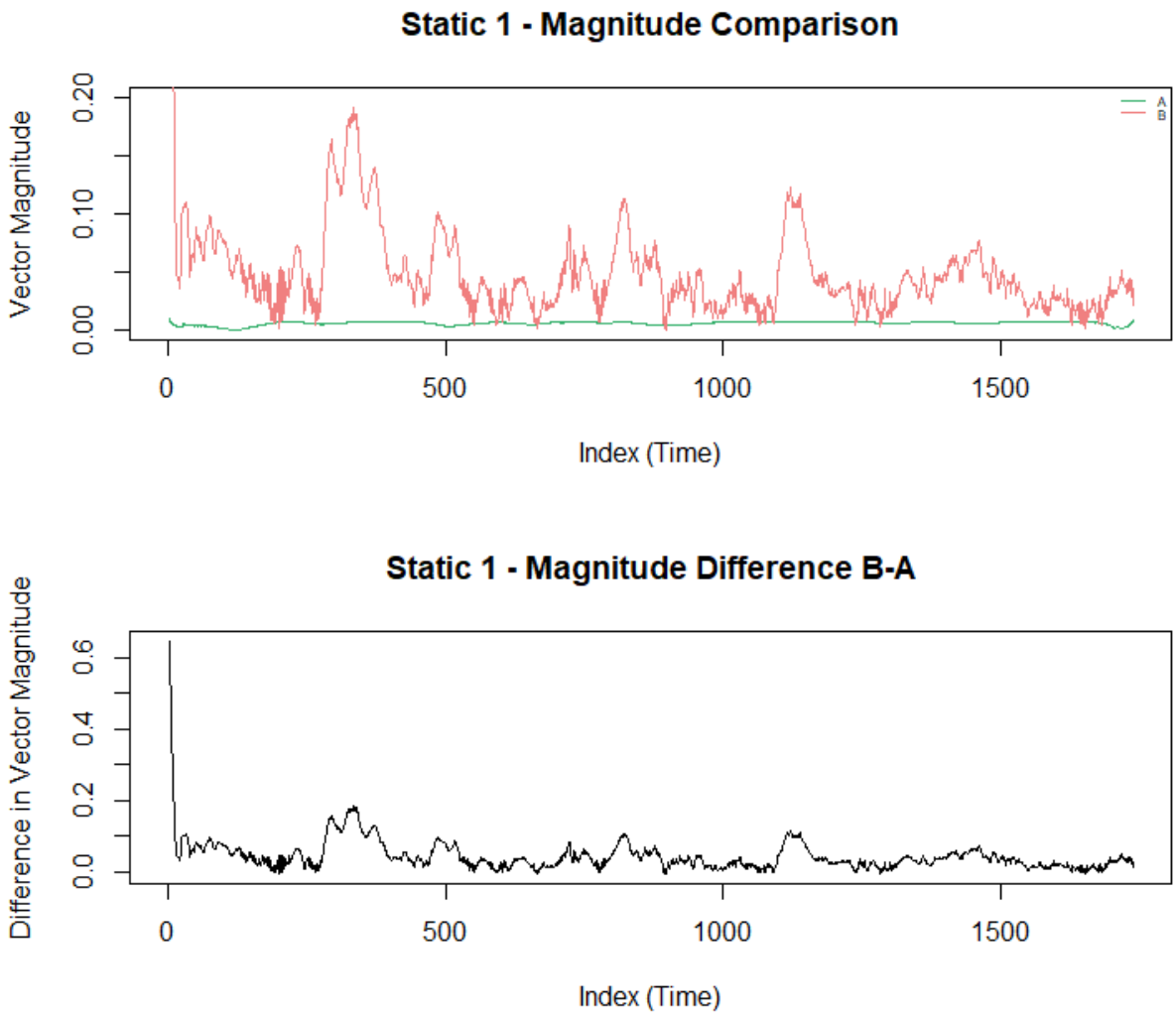

Figure 6 - Static 1 – Magnitude

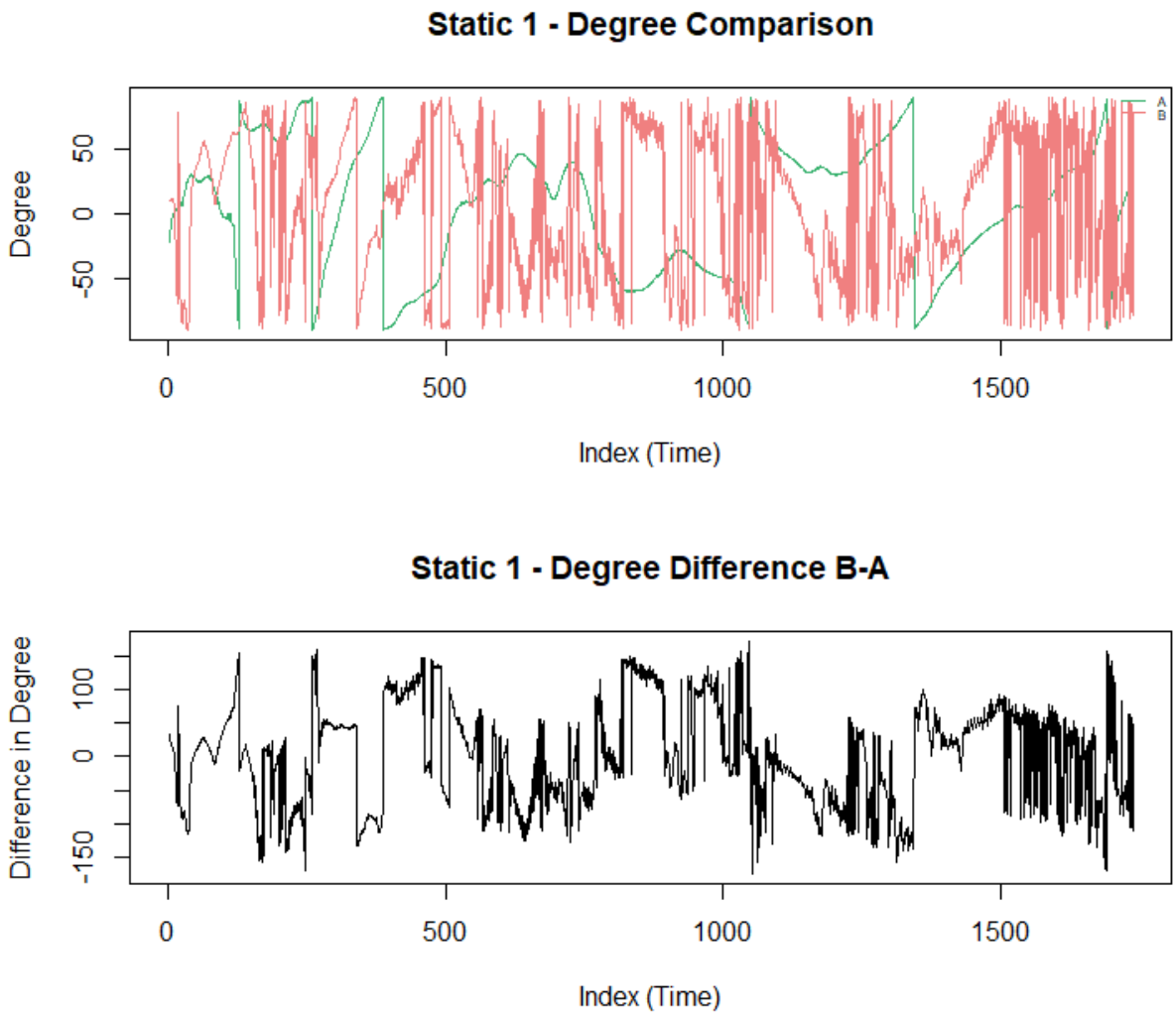

Figure 7 - Static 1 - Degree

SSM2

The final 642 coordinates in Function B were skipped in this comparison due to the method used.

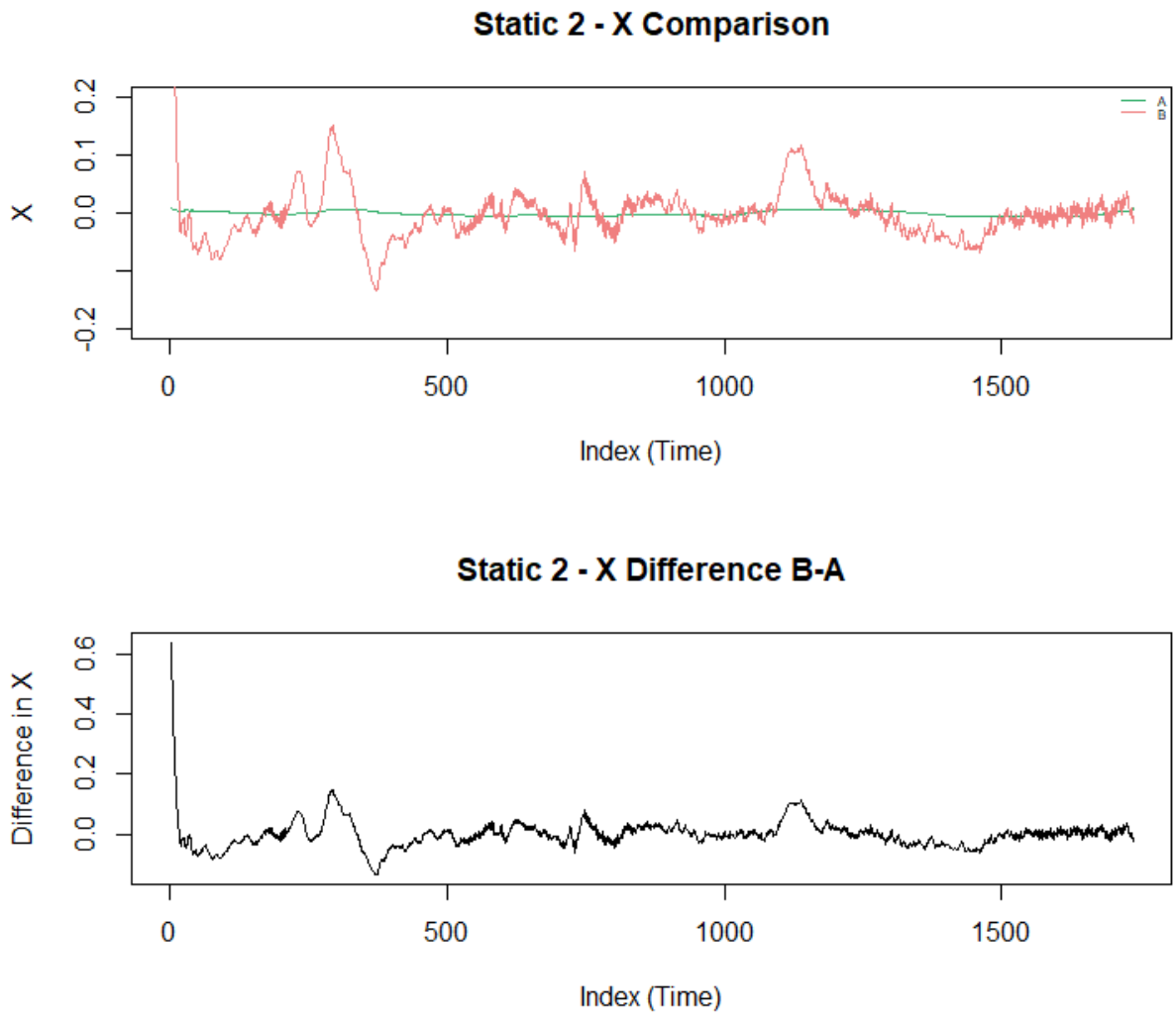

Figure 8 - Static 2 – X

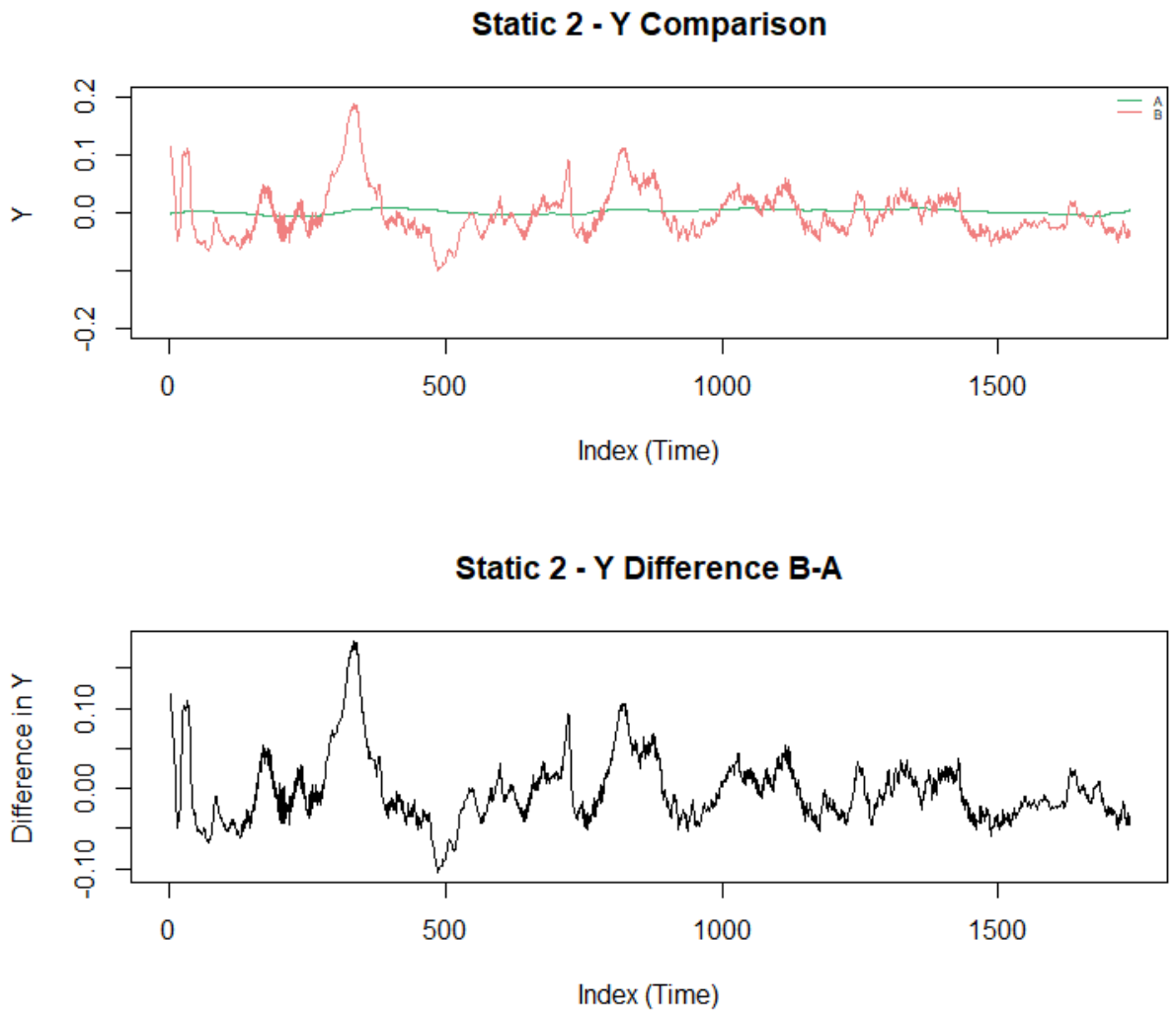

Figure 9 - Static 2 - Y Difference B-A

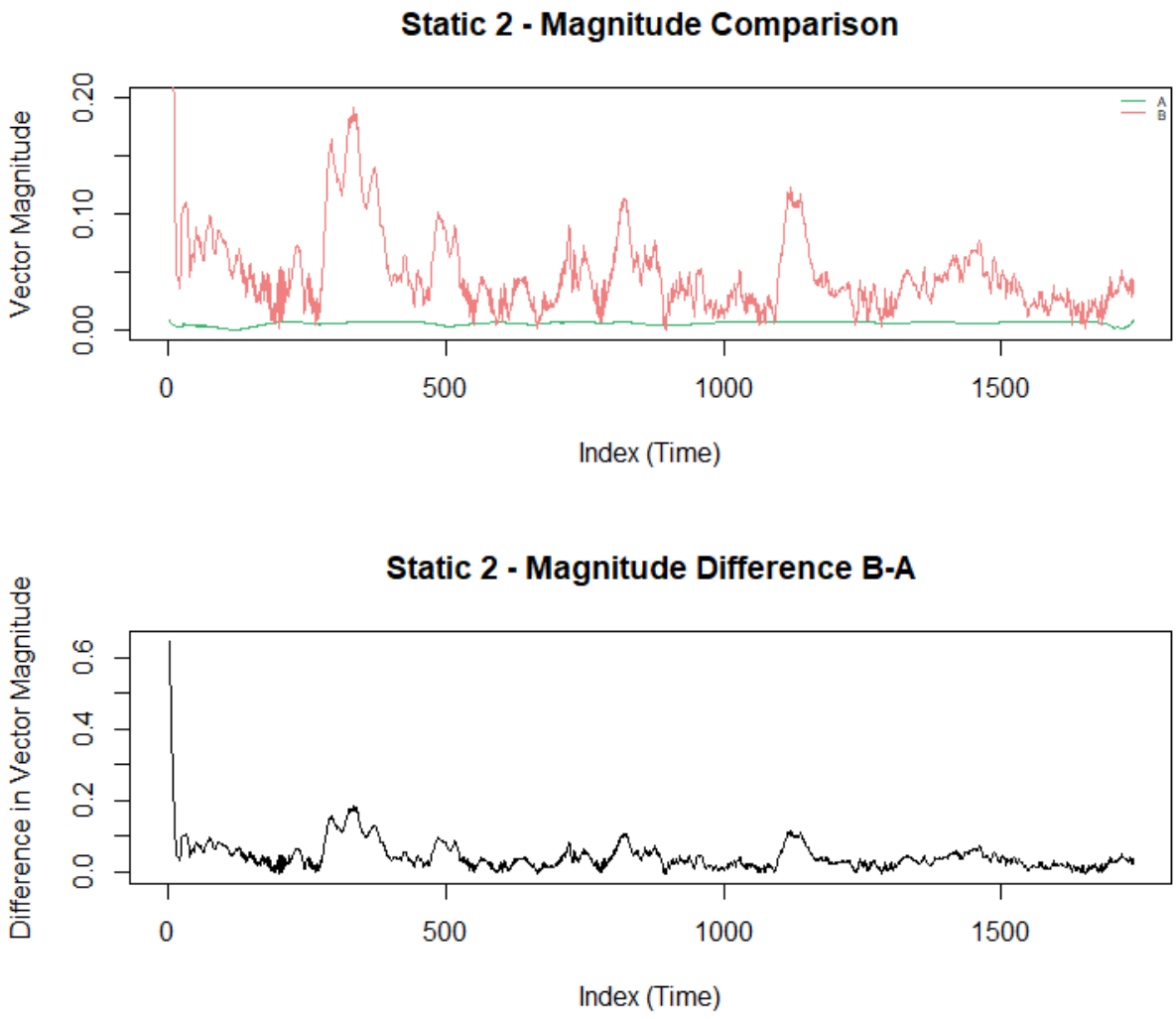

Figure 10 - Static 2 – Magnitude

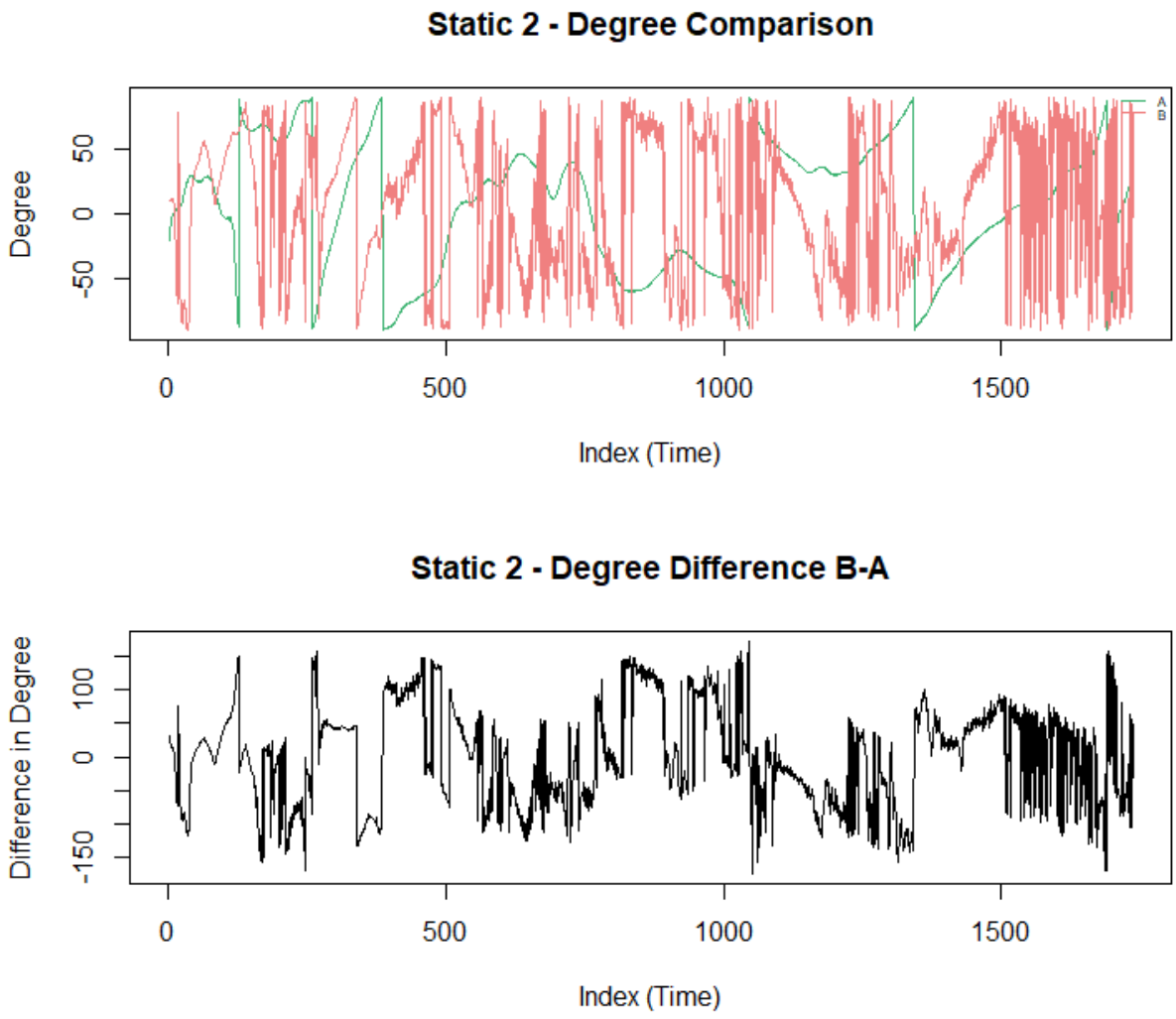

Figure 11 - Static 2 - Degree

DSM1

No coordinates were skipped in this comparison method.

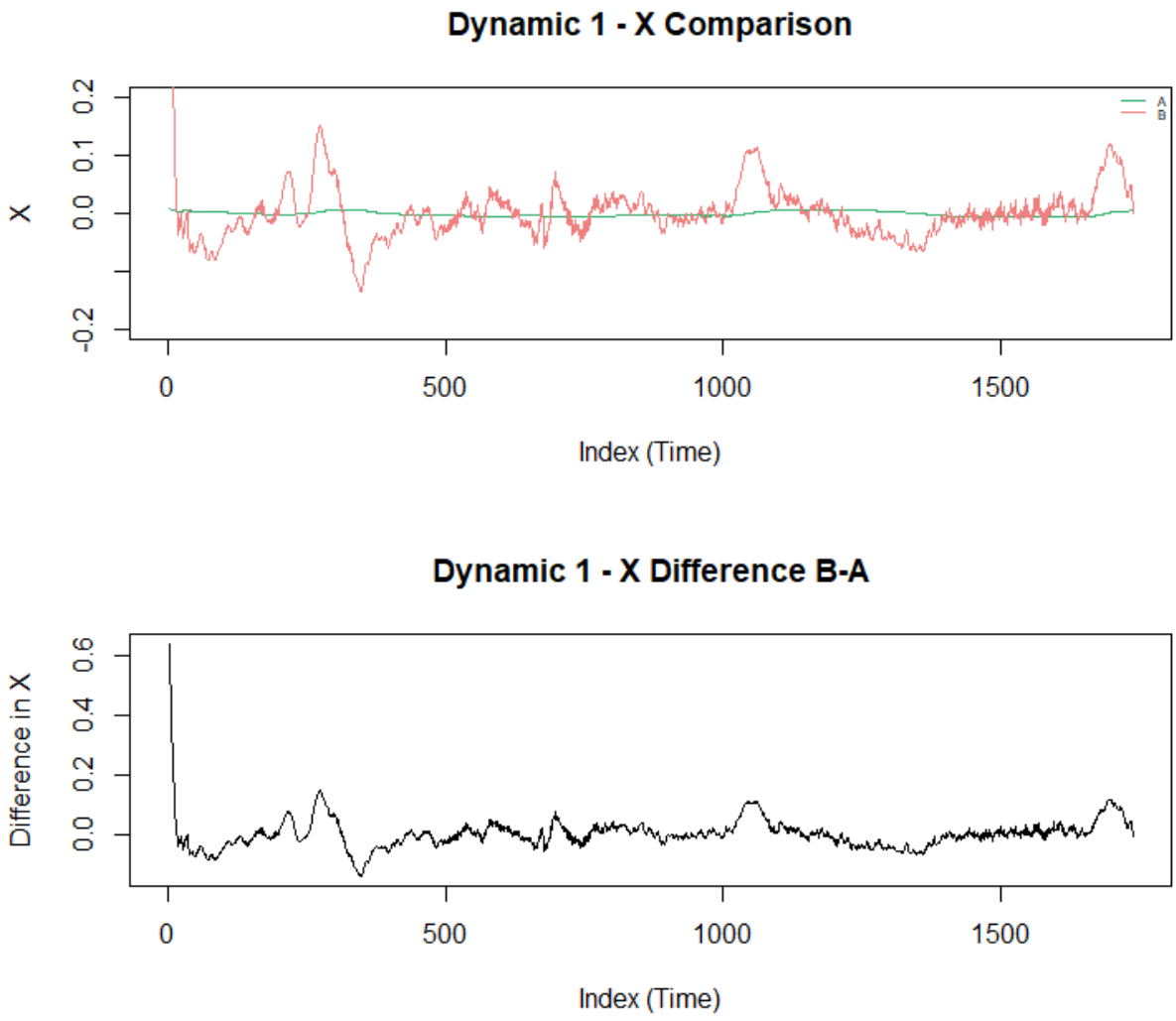

Figure 12 - Dynamic 1 – X

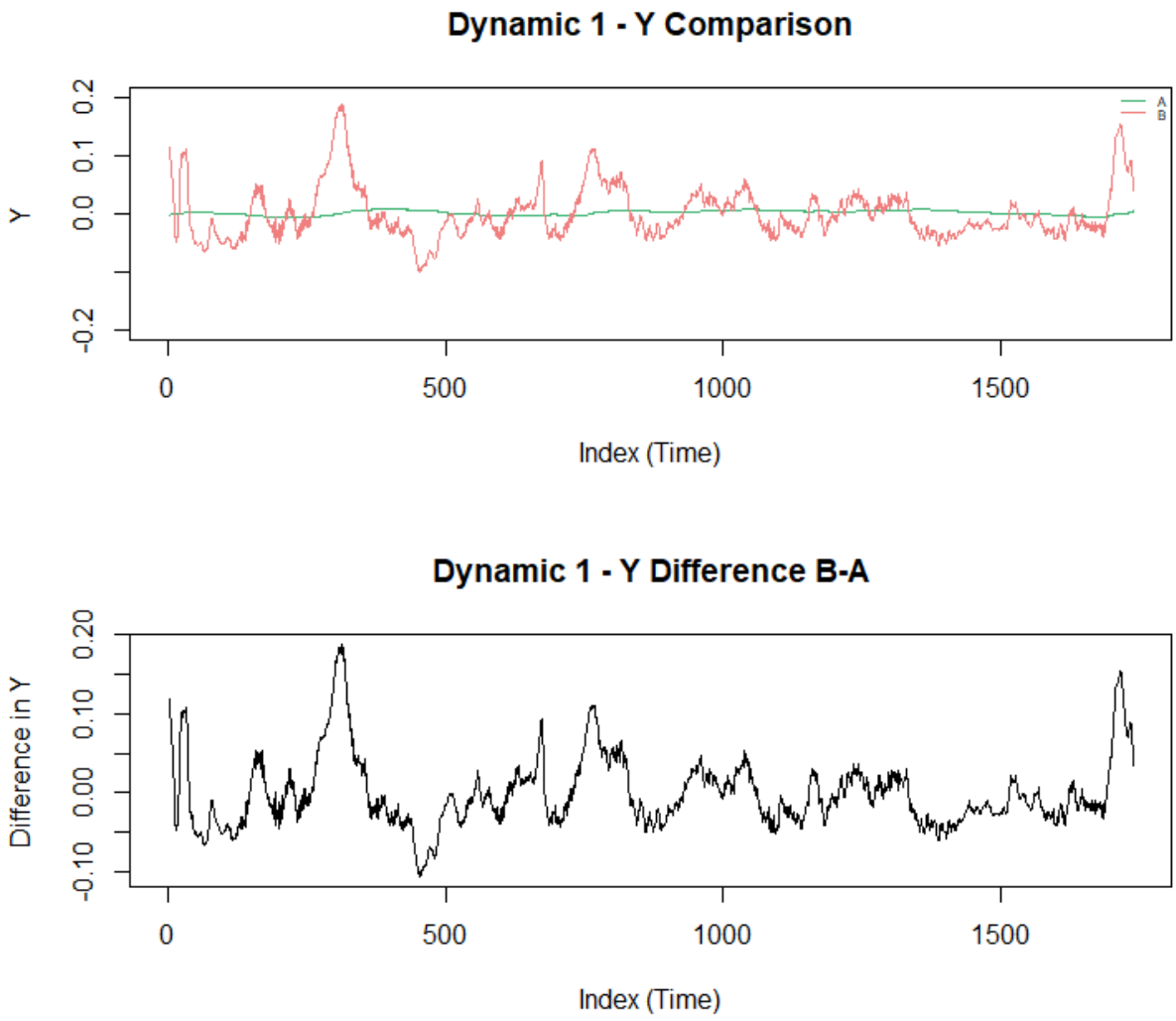

Figure 13 - Dynamic 1 – Y

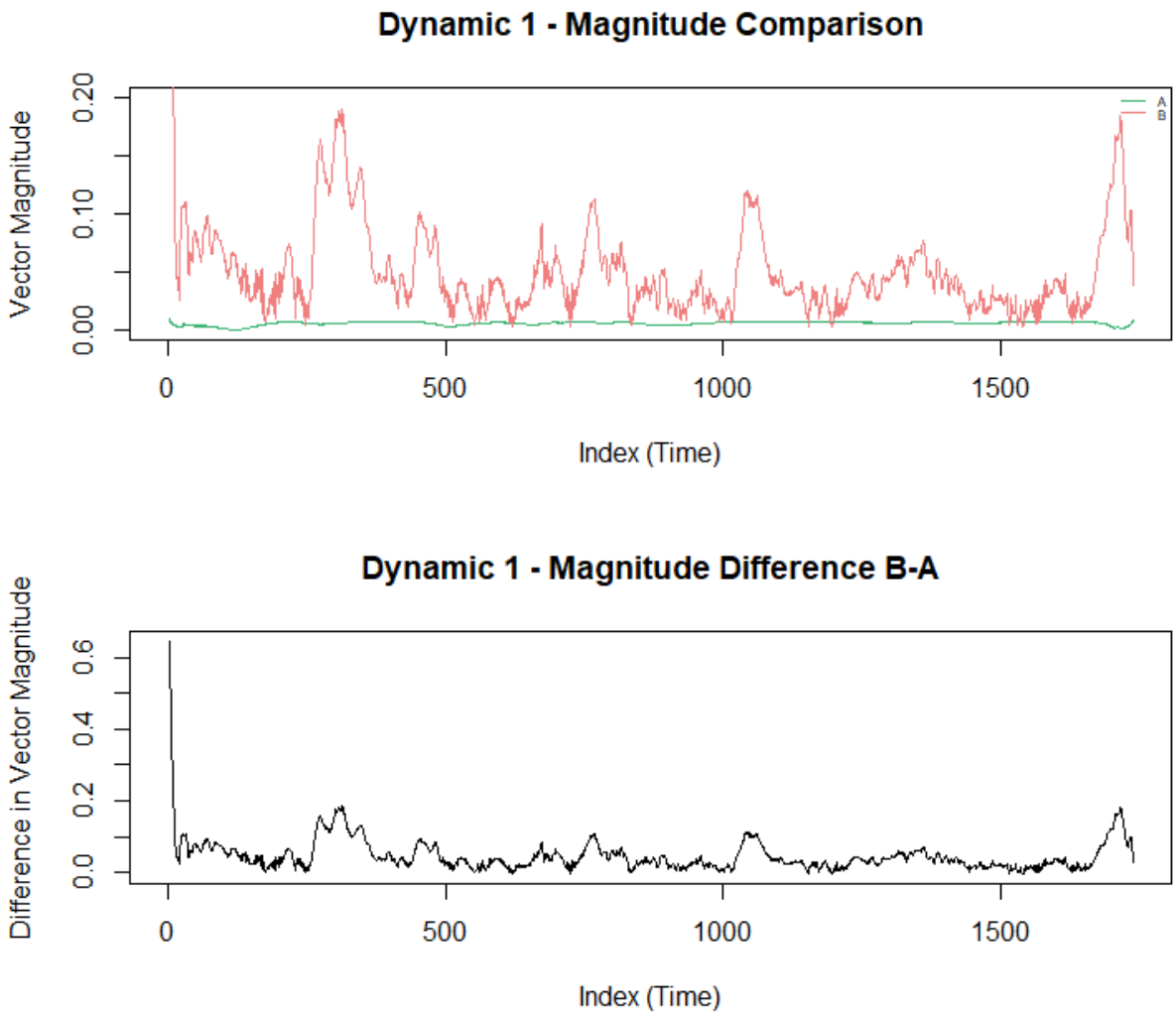

Figure 14 - Dynamic 1 – Magnitude

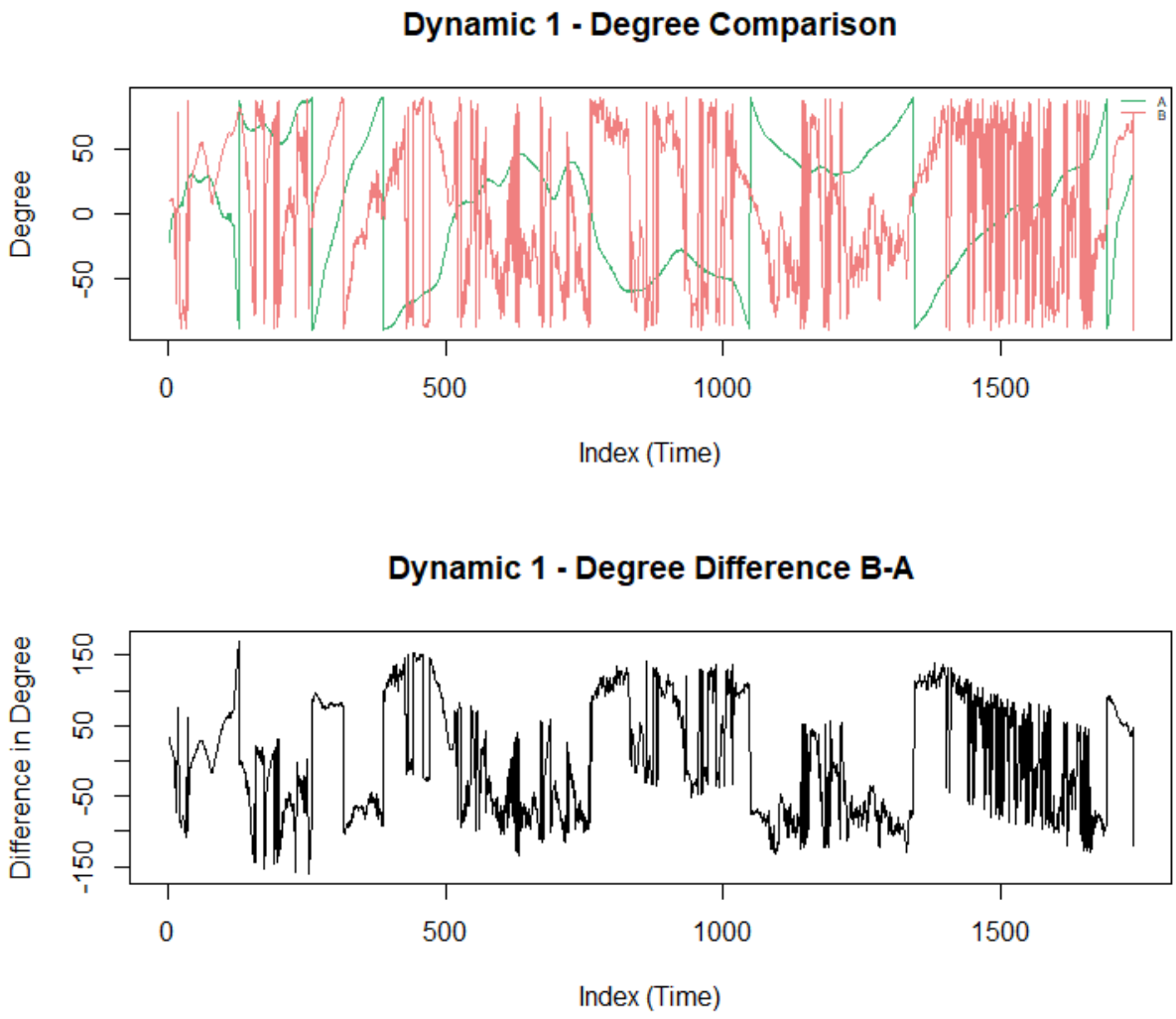

Figure 15 - Dynamic 1 - Degree

DSM2

No coordinates were skipped in this comparison method.

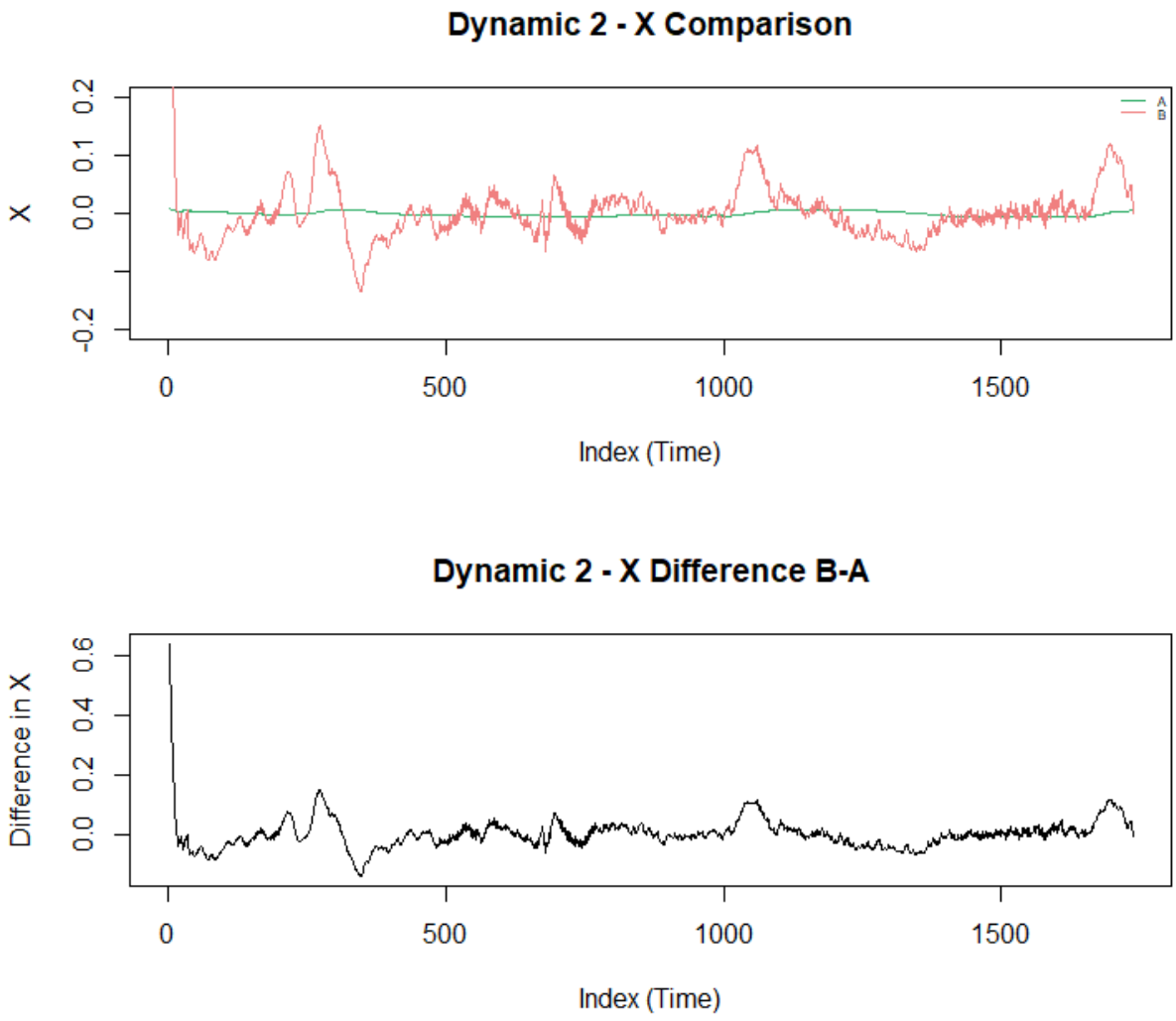

Figure 16 - Dynamic 2 – X

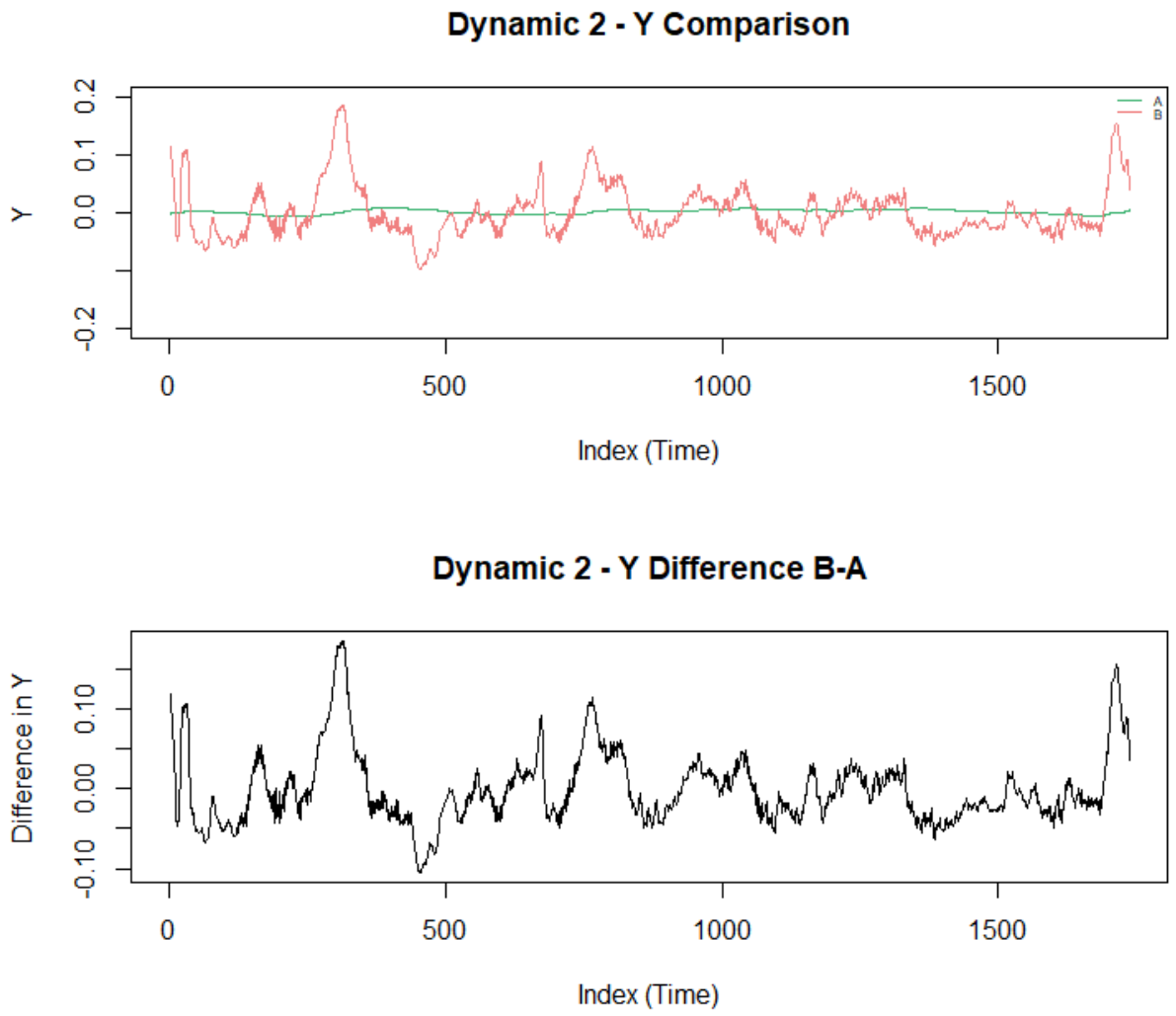

Figure 17 - Dynamic 2 - Y

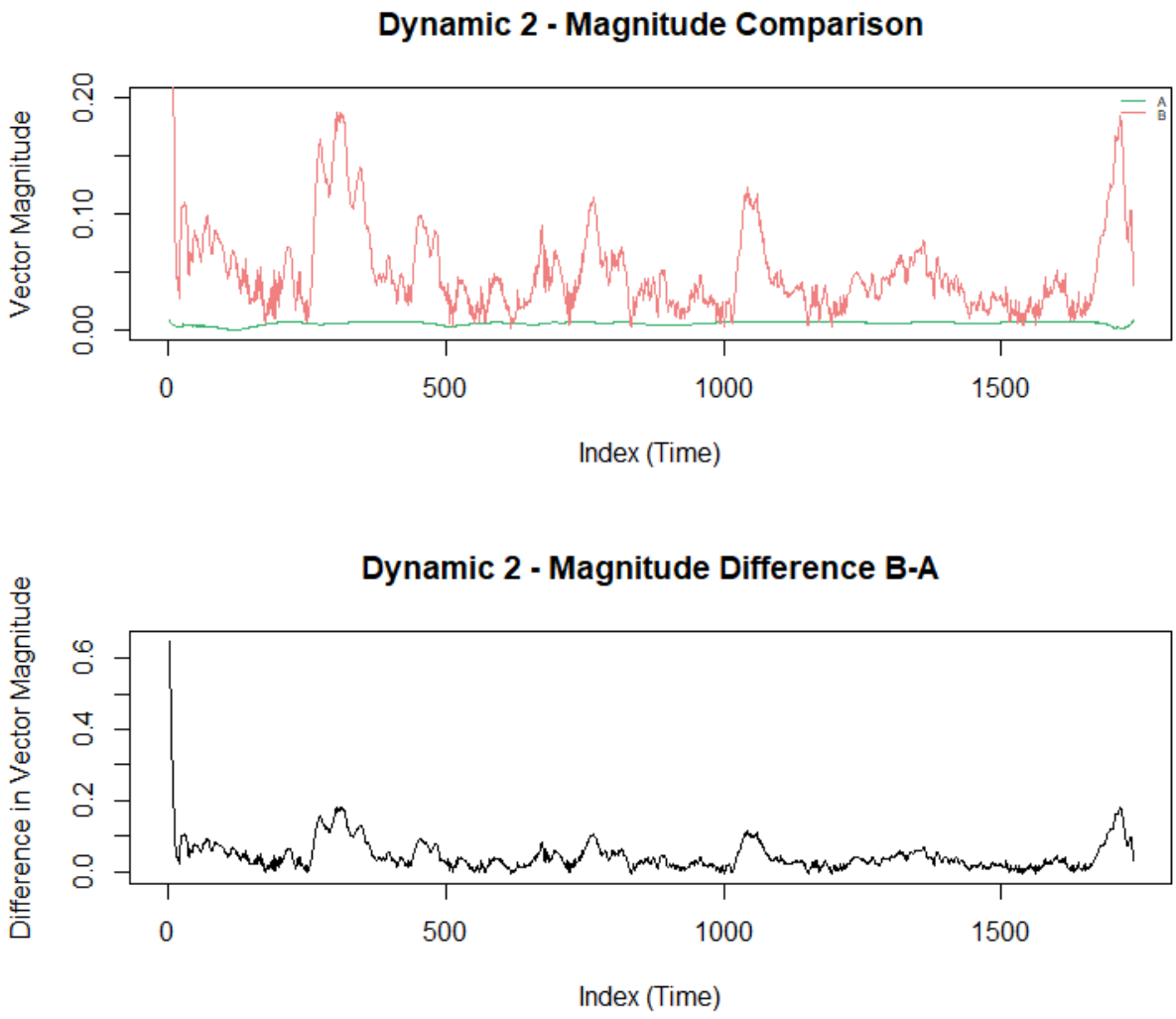

Figure 18 - Dynamic 2 – Magnitude

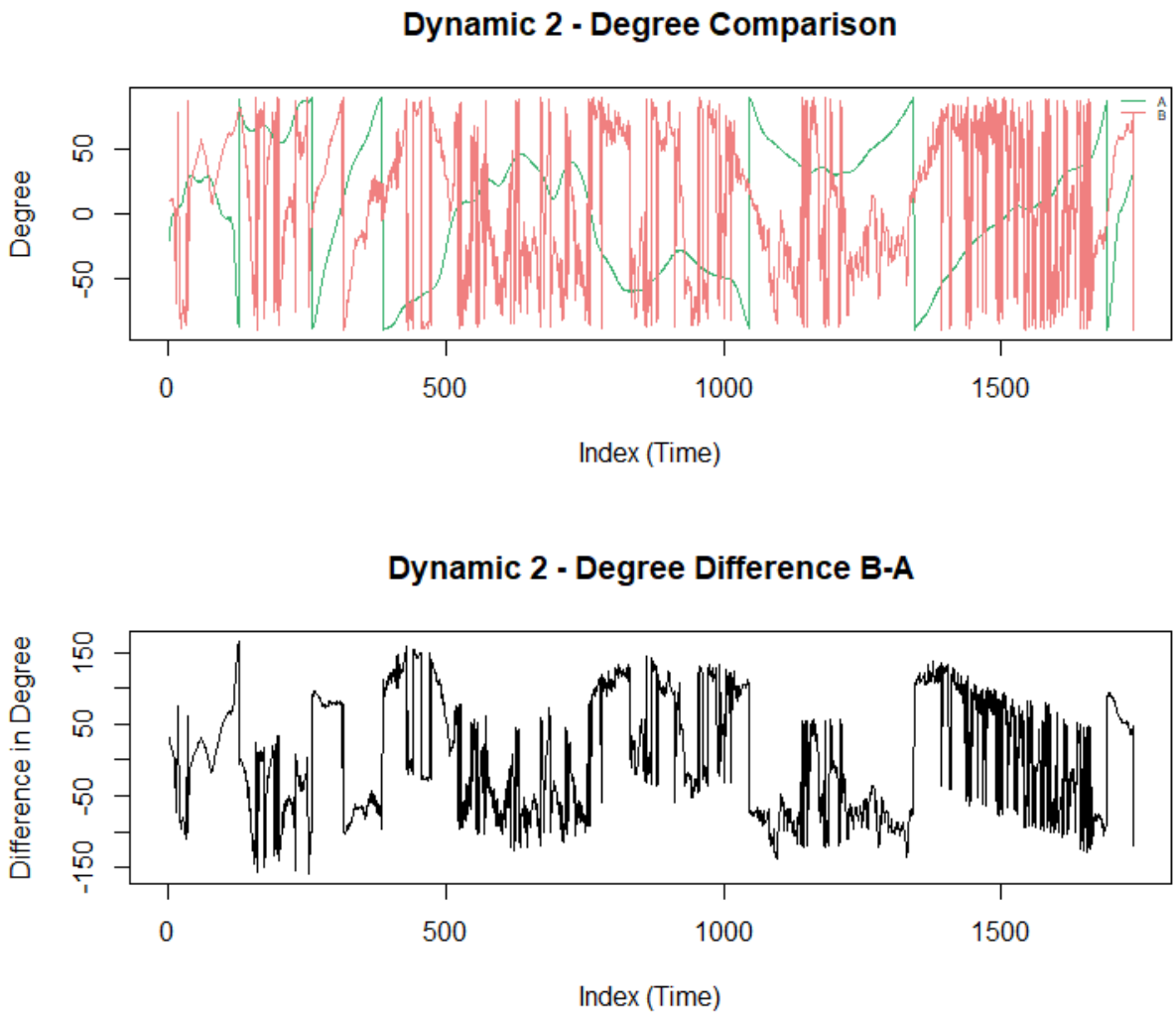

Figure 19 - Dynamic 2 – Degree

Overall Mean

Method	Mean of Difference of X	Mean of Difference of Y	Mean of Difference of Magnitude	Mean of Difference of Degree
SSM1	-0.000519901	-0.002764169	0.04402508	1.611708
SSM2	-0.0005176885	-0.002753617	0.04404393	1.661593
DSM1	0.002816749	-0.00092656	0.04543144	2.710786
DSM2	0.002818639	-0.0009473579	0.04543719	2.971562

Table 1 - Overall Mean

Conclusions

Comparison Methods Reducing the Noise

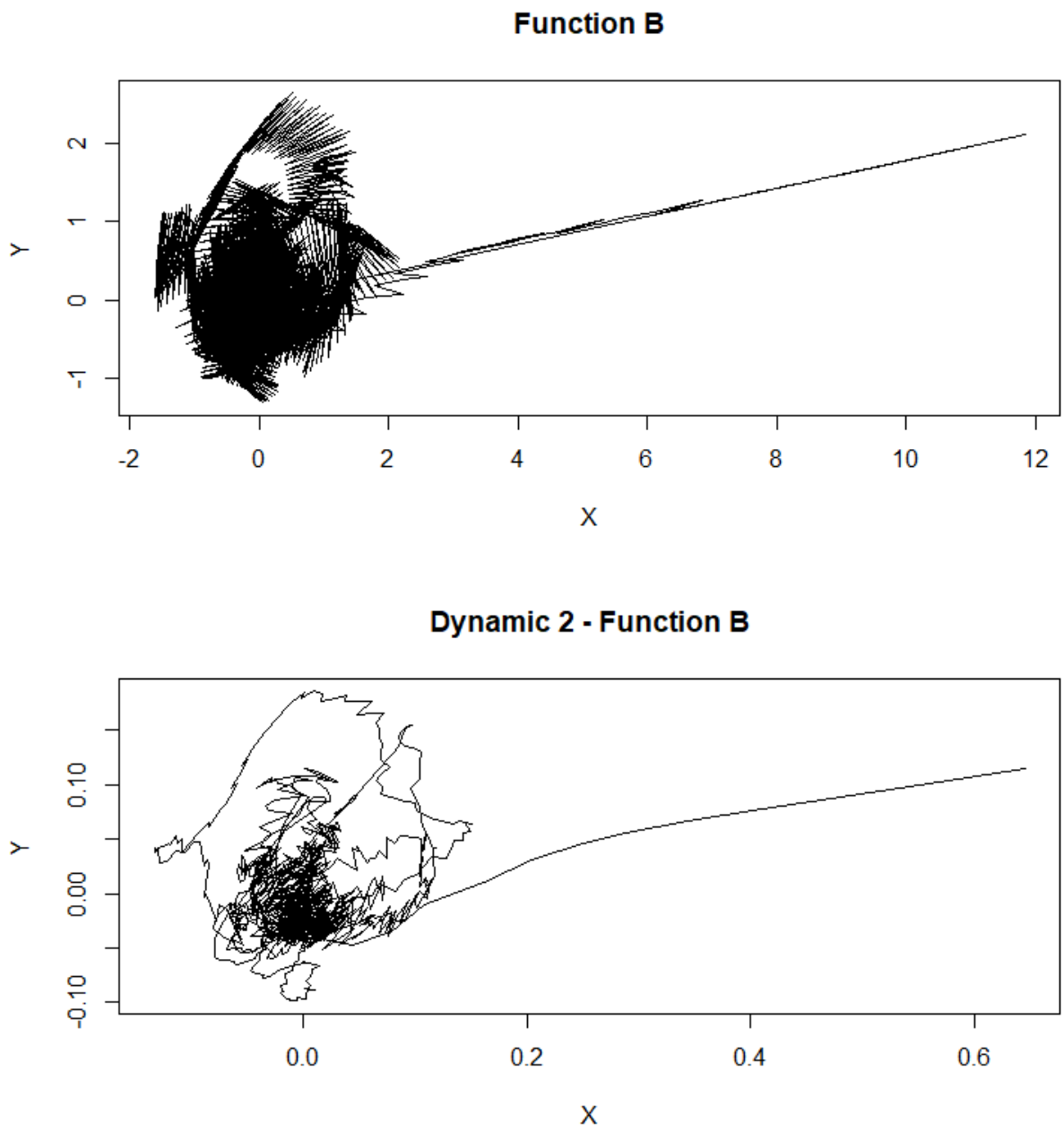

Figure 20 - Function B vs DSM2 Function B

As we can see from the above figure, when applying DSM2 (or any of the methods), not only do we generate the comparison data for Function B to Function A, we reduce a lot of the noise that the original function initially had. It is much smoother and retains the same shape that the initial function had.

SSM1 vs SSM2

Overall, the graphs for both static methods remain relatively the same as the difference in the graphs are relatively small. Their means were also relatively close to each other in values in all respects. Therefore, the differences between SSM1 and SSM2 are relatively small.

DSM1 vs DSM2

Likewise, the graphs for both dynamic static methods remain relatively the same as the difference in the graphs are relatively small. Their means were also relatively close to each other in values in all respects. Therefore, the differences between DSM1 and DSM2 are relatively small.

Method 1 vs Method 2

From the conclusions above, we can see that Method 1 and Method 2 do not differ all that much in the grand scheme for these two functions.

However, since we used the average of two points in Function A in method 2, we essentially gave double the weight to every point except the first and last point. This has a potential to cause problems because we are not giving equal weight to every point. Another problem is that we're essentially "guessing" what the actual data is between the two points using the average. The actual data could have potentially been any number.

Axis Differences, Magnitude Differences, Degree Differences

From looking at the figures in the results section for all methods, we can see that the axis differences and magnitude for Function A were much smaller when compared to Function B. Function B looked like it had lots of noise, and was changing at a much more rapid pace when compared to Function A.

For the degrees on the other hand, Function A did change as much as Function B. However, it is still very obvious that Function B had a lot more noise and was changing at a much more rapid pace when compared to Function A.

Static Method

The static methods SSM1 and SSM2 resulted in 637 and 642 coordinates ignored respectively. This was due to the fact that I rounded down using the floor function to determine how many coordinates in Function B were to be mapped to every coordinate in Function A. Due to this, we essentially shifted and squished Function B to fit Function A. The result of this is what is known as error propagation. Every-time we map a number of points from Function B to function A, we introduce a certain amount of error in the calculations. And as a result, we ended up losing 637-642 coordinates of data.

Dynamic Method

The dynamic method DSM1 and DSM2 aimed to help fix the error propagation from the static methods. The result of this method is that we did not ignore any coordinates from Function B at all. Of course, there is still some error by doing both Static and Dynamic. We are still "guessing" what the true data is at a certain time frame for Function B relative to Function A. As there is no way to know what the true data is at that time, the best we can do is take the average of the points.

Static vs Dynamic

To begin to see the obvious difference between Static and Dynamic, we can look at the following figure below.

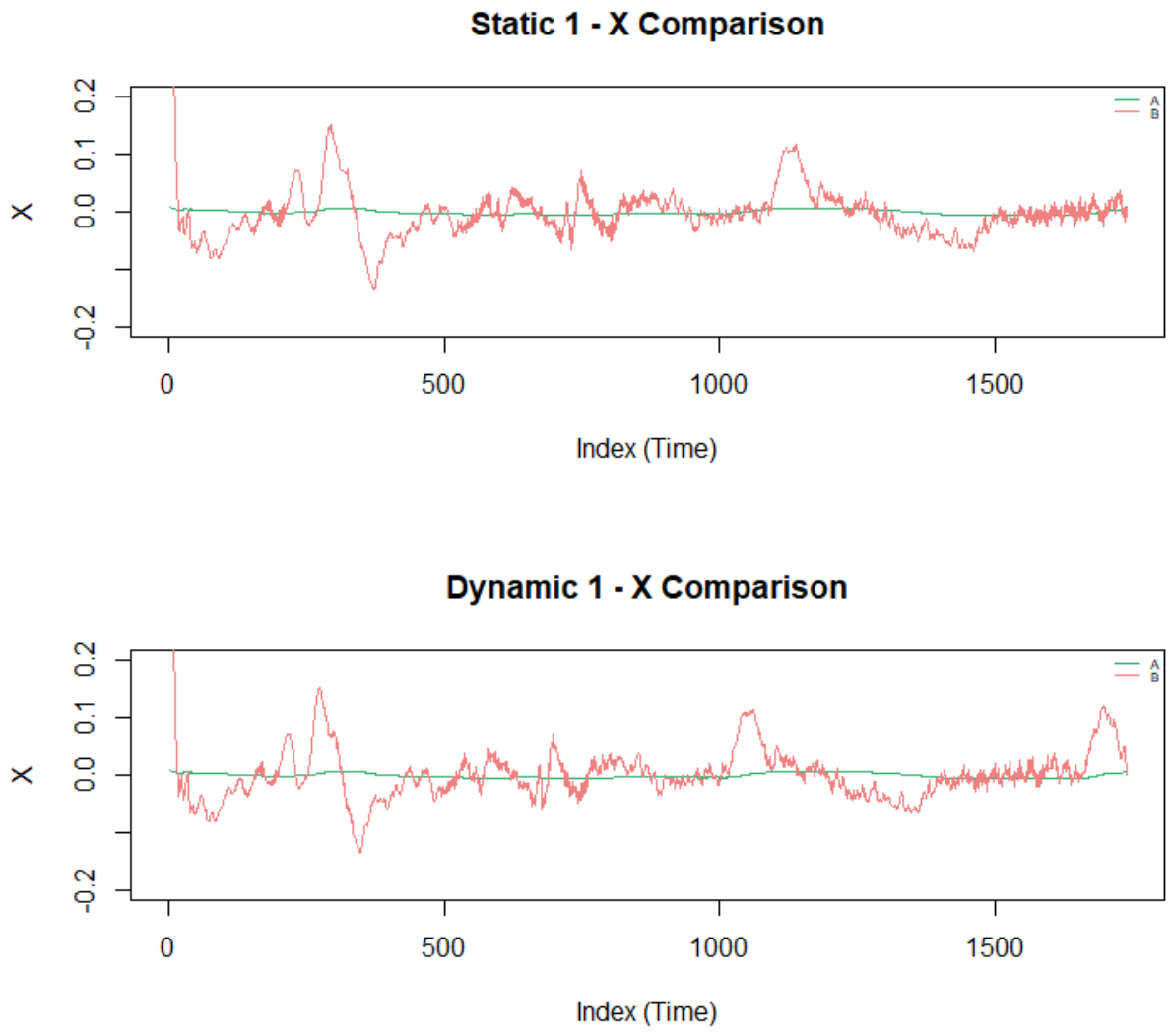

Figure 21 - Static vs Dynamic – X

As discussed previously, the static method essentially cuts-off a portion of the end of Function B. In Figure 21, we can see that with the dynamic method, the cut-off potion is included in the comparison. And that part that was cut-off was a pretty significant portion of Function B.

This significance can also be shown in the means table (Table 1). The mean differences between X and Y, and also the degree was significantly more different between static and dynamic.

This shows how much the error propagation from the static method affected the overall result versus the dynamic method.

Overall

In conclusion, we extracted derivatives and gradients of a 2D Function A and compared the results to the Function B. The derivative of Function A and Function B were very different in almost all aspects. There were four different methods that were used to compare the derivative of Function A with Function B: M1 vs M2 were relatively the same in all aspects. Although M2 had potential errors due to "guessing" with average. Function A had very small fluctuations when compared to Function B, and it had much less noise when compared to Function B. The static methods were simpler to implement but had error propagation and caused many coordinates to be ignored. The solution for this was to dynamically choose how many steps to take in Function B per step in Function A. This resulted in much less error propagation and allowed all coordinates in Function B to be compared to Function A, resulting in a much better comparison between the derivative of Function A and Function B.