Time Series Decomposition with Singular Spectrum Analysis

Connor Wilson, Walter Virany, Gabe Wallon

July 18, 2022

1 Abstract

Single Spectrum Analysis (SSA) is a technique used to decompose time series data into a trend, seasonal components, and noise. This allows us to analyze time series data through the lens of components, namely slow varying (trends) and periodic. In this paper we research and apply SSA to two time series with different characteristics. We then compare the process of applying this technique to these two data sets and the results we obtain from it.

2 Attributions

Walter Virany, Gabe Wallon, and Connor Wilson all worked together to find data to analyze using SSA. Walter implemented the SSA algorithm in MATLAB. Gabe composed the mathematical formulation section and Connor created the LaTeX document. Everyone contributed in analyzing and discussing the results.

3 Introduction

Singular Spectrum Analysis (SSA) is a method for deconstructing time series into a sum of components, namely a slow varying trend, seasonal components, and 'structureless' noise. SSA can be used in a broad range of applications due to a lack of assumptions made by the model; Assumptions which are common in other time series and forecasting analysis techniques[1]. In SSA the primary tool used is Singular Value Decomposition (SVD). The purpose of this paper is to understand the method of SSA and to demonstrate implementation of SSA to a time series that has minimal external factors associated with it versus a time series that has an abundance of external factors. We are interested in comparing the SSA procedure when applied to data with a dominant oscillatory attribute versus data with a clear upward trend. In section 4 there is an in-depth mathematical formulation of how SSA decomposes a time series and reconstructs it as a sum of components. In section 5 SSA is applied to two sets of data, the first set of data is the average monthly temperatures in Boulder, Colorado from January, 1991 to March, 2022, and the second set of data is the housing price in the "Mountain" geographic region from January, 1991 to March, 2022 from the House Price Index (HPI). HPI is a broad measure of the change in single-family housing price over a period of time. HPI measures repeated sales of houses to determine how the housing market fluctuates in a given area. The choice behind these data sets was because the climate in Colorado has minimal external factors, it will be roughly the same each year. On the other hand, the housing market changes based on various different factors. In this paper, external factors refers to variables other than time that basic SSA does not account for. Section 6 hosts our conclusion and the future work associated with SSA which could be done.

4 Mathematical Formulation

There exist several versions of SSA, which arise from modifications being made to the "Basic SSA" algorithm depending on characteristics of the data. For example, a stationary time series - one where the data have no observable relationship to time - is more accurately analyzed using "Toeplitz SSA"[5]. In this paper, we will be applying Basic SSA to our data sets. There are two main stages to basic SSA with each stage consisting of two steps. The first stage is decomposition which includes embedding and singular value decomposition (SVD). The second stage is reconstruction which consists of eigentriple grouping and diagonal averaging.

4.1 Decomposition

4.1.1 Embedding

Embedding is the first method applied in the procedure of SSA. This step maps your original one-dimensional time series $Y = (y_1, y_2, ..., y_N)$ to a multi-dimensional matrix $X = [\mathbf{y}_1 \dots \mathbf{y}_K]$, referred to as the trajectory matrix of the series. You then set the $\mathbf{y}_i$ column vectors to size L, referred to as the window length. These column vectors resemble $\mathbf{y}_i = (y_i, \dots y_{i+L-1})^T$ for $1 \leq i \leq K$ where $K = N - L + 1$. You can easily see that X is a *Hankel* matrix as all of the anti-diagonal entries are equal, or $X_{i_1,j_1} = X_{i_2,j_2}$ when $i_1 + j_1 = i_2 + j_2$.

$$X = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_K \\ y_2 & y_3 & y_4 & \cdots & y_{K+1} \\ y_3 & y_4 & y_5 & \cdots & y_{K+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_L & y_{L+1} & y_{L+2} & \cdots & y_N \end{bmatrix}$$

The choice of the window length L depends upon certain attributes of the data you are analyzing, including the size of your data set. For data where a periodic component is expected, the window length should be chosen to be a multiple of the period T where $\frac{T}{N}$ is an appropriately large number given the size of the data set. For data where there is a dominant or clearly visible trend, the window length parameter can be chosen from a wide range, whereas most trends are more easily detected using longer window lengths.[1].

4.1.2 Singular Value Decomposition

Now the procedure calls for us to find the SVD decomposition of the trajectory matrix $X_{L\times K}$. For matrices $X_{L\times K}$ where $K >> L$ it is easier to find the SVD of X^T and then transpose it, than it is to find the SVD of X. Let $S = XX^T$, $\{\lambda_1, \lambda_2, \cdots, \lambda_d\}$ be the eigenvalues of S in decreasing order, and $\sqrt{\lambda_i} = \sigma_i$ be the singular values of X. Solving the equation $det(S - \lambda I) = 0$ (where I is the identity matrix in this case), we can find an orthonomal set of eigenvectors of S (or singular vectors of X^T) to be the columns of matrix U, $\{\mathbf{u}_1, ..., \mathbf{u}_d\}$, where d is the maximum i such that $\lambda_i > 0$. Now let matrix V have columns vectors $\mathbf{v}_i = \frac{1}{\sigma_i} X^T \mathbf{u}_i$, so the SVD decomposition of X^T is as follows $X^T = V\Sigma U^T$ (matrix Σ is a diagonal matrix with the singular values on the diagonal.) Therefore, $X = U\Sigma V^T$ and can be written as $X = X_1 + ... + X_d$ where $X_i = \sigma_i \mathbf{u}_i \mathbf{v}_i^T$. These matrices $X_1, ..., X_d$ are referred to as elementary matrices. The gathering $(\sigma_i, \mathbf{u}_i, \mathbf{v}_i^T)$ is referred to as an eigentriple of a matrix. In most real world applications it is found that the rank d of X is equal to L.

4.2 Reconstruction

4.2.1 Grouping

Now that there is a decomposition of the original trajectory matrix, $X = X_1 + \ldots + X_d$, we must form disjoint sets of the elementary matrices then sum the elements in the group, a process called eigentriple grouping. Let $I \subset \{1, \ldots, d\}$. Then the matrix $X_{I_1} = \sum_{i \in I} X_i$. The goal is to create a new decomposition for the original trajectory matrix $X = X_{I_1} + \ldots + X_{I_m}$, where each X_I has the maximum possible similarity to Hankel matrix form. If these components are Hankel matrices themselves, the original time series Y can be written as a sum of components $Z_1, \ldots, Z_m$ where each element in the original time series is $y_i = z_{I_1} + \ldots + z_{I_m}$. The degree to which these X_{I_i} matrices are similar in form to Hankel matrices correlates with how well they represent separable components of the original time series. The resemblance of each X_{I_i} to a Hankel matrix is generally not very strong, which motivates our next method in this procedure. This grouping step can also be omitted from the procedure, equivalently, every disjoint set of X_i matrices, or every group, only consists of one matrix.

4.2.2 Diagonal Averaging

After eigentriple grouping, we now transform each matrix X_{I_i} into a new time series Z_i of length N, $Z_i = (z_1, \ldots, z_N)$, using a method known as *diagonal averaging* or *hankelization*. In implementing this algorithm, the average of the antidiagonals is taken by adding each entry in the antidiagonal and dividing by the number of entries. For example, applying this method to some matrix X with entries $X_{i,j}$, the first entry of the constructed time series Z would be $z_1 = \frac{x_{1,1}}{1}$ and the third $z_3 = \frac{x_{2,1}+x_{2,2}+x_{1,2}}{3}$ and so on.

$$z(k) = \begin{cases} \frac{1}{k} \sum_{j=1}^{k} x_{j,k-j+1} & 1 \leq k \leq L \\ \frac{1}{L} \sum_{j=1}^{k} x_{j,k-j+1} & L < k < K \\ \frac{1}{N-k+1} \sum_{j=k-K+1}^{N-k+1} x_{j,k-j+1} & K \leq k \leq N \end{cases}$$

It can be proven that hankelization provides the closest possible hankel-representation of each X_{I_i} matrix. In other words, out of all hankel matrices of the same size as X_{I_i}, the one found using hankelization is the closest to X_{I_i} by means of the Frobenious norm[1]. After diagonal averaging is applied to each X_{I_i}, we now have a decomposition of the original time series such that $Y = Z_1 + \ldots + Z_m$, where each Z_i is a time series of the same length as Y. These independent series are referred to as the components of the original time series and the ultimate goal of SSA is to analyze these components.

5 Examples and Numerical Results

5.1 Applying SSA to Average Monthly Temperatures in Boulder, Colorado

We start by implementing SSA on the average monthly temperatures recorded in Boulder, Colorado from January, 1991 to March, 2022. We expected to see a very clear annual cycle in the average monthly temperatures. Figure 1 shows the original time series before implementing the SSA algorithm.

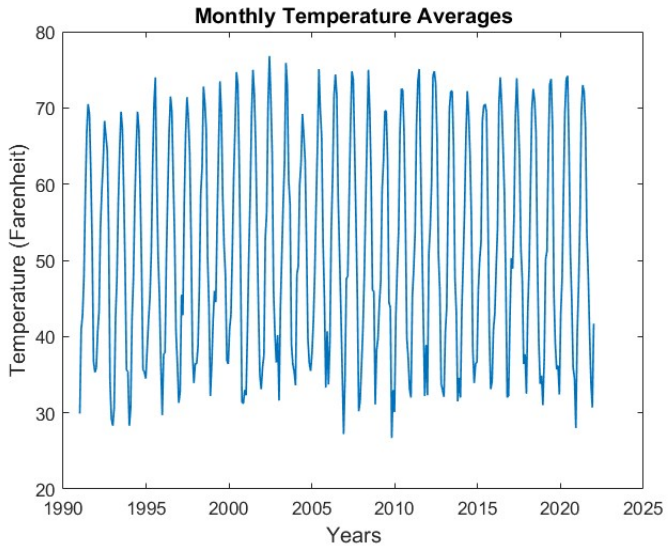

Figure 1: Original Time Series of Monthly Average Temperature in Boulder, Colorado

From this, we can observe a periodic cycle of approximately twelve months, but can see no obvious long-term trends in the data. In order to perform SSA on the time series in question, we first need to determine a window length, L. Since we suspect an annual period, and our data is monthly, we made our window length 12.

5.1.1 Extracting the Trend

The first step of the SSA process is to embed the time series into the trajectory matrix, X. We then perform SVD on X, yielding the matrices U, Σ, and V. Through this process, we find the singular values of X, shown in Figure 2.

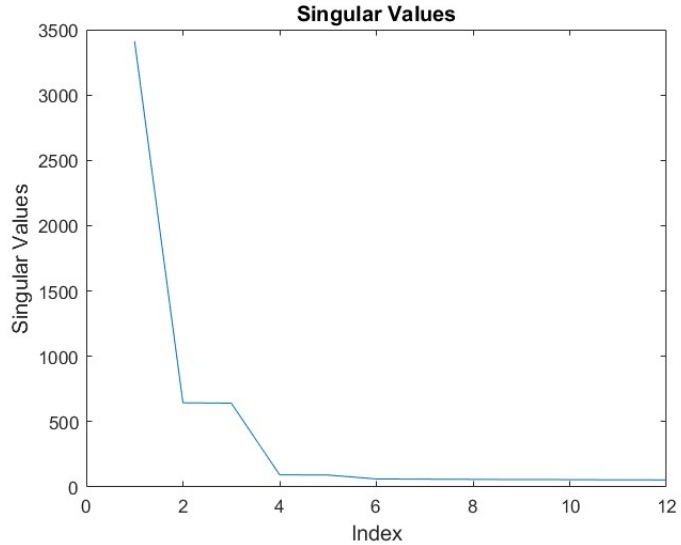

Figure 2: Scree plot of SSA for Monthly Temperature Data

Here the 'elbow' is observed between the second and third components, so everything before the elbow can be attributed to the overall trend of the data. Figure 3 shows the plots of each eigenvector. The first eigenvector shows constant values, indicating that there is no significant long-term trend in the data.

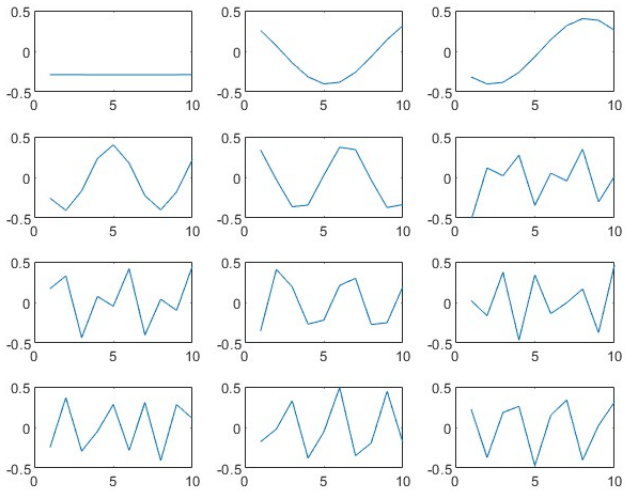

Figure 3: Eigenvectors of Step 1

At this point in the procedure, we choose to omit the grouping step because of our relatively small number of eigentriples. Figure 4 shows the reconstructed time series created using the diagonal averaging method. The first plot shows the trend, and the other plots represent high-frequency oscillations that don't contribute to the overall trend of the data.

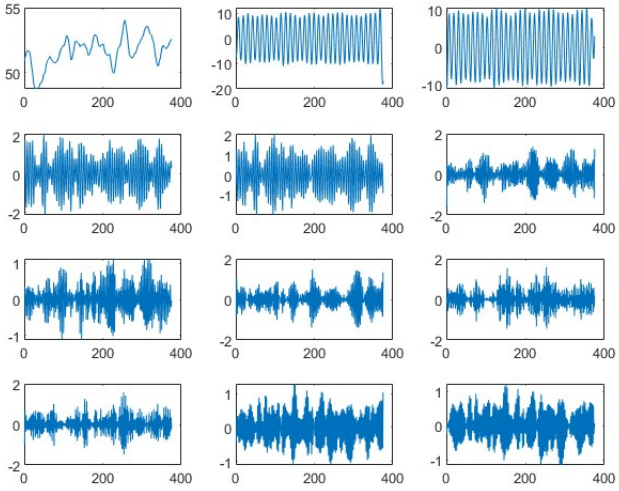

Figure 4: Reconstructed Time Series of First Step

Figure 5 shows the extracted trend of the data plotted over the original time series.

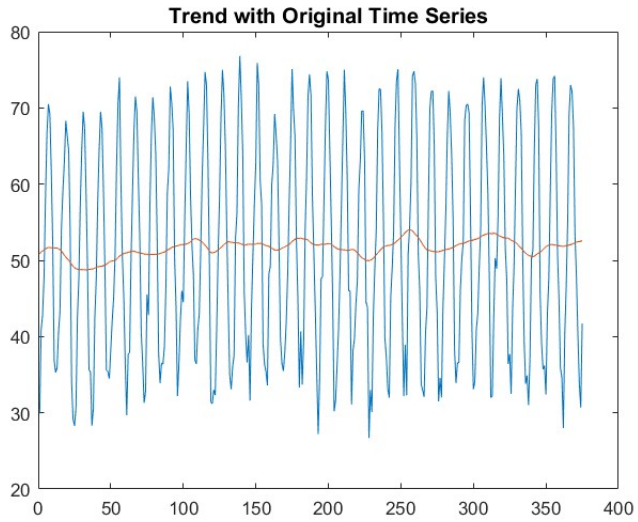

Figure 5: Trend and Original Time Series

5.1.2 Extracting the Seasonal Component

The residuals are calculated by subtracting the trend from the original time series. This is shown in Figure 6.

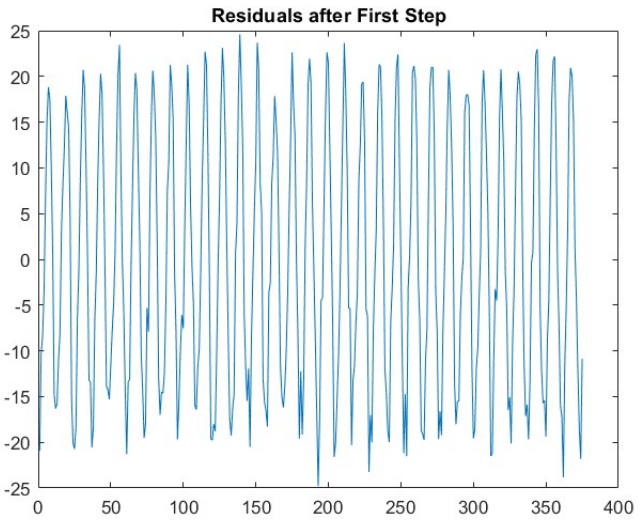

Figure 6: Residuals of First Step

In order to extract the seasonal component of the data, we must repeat the SSA algorithm again on the residual time series. The first step of this process is to decide on an optimal window length. As Cheng suggests, the window length, L, should be some maximum number less than or equal to $\frac{N}{2}$ that is a multiple of our suspected period, 12 [2]. So, we choose L = 180 for our window length in this step. Figure 7 is the scree plot of singular values for the residuals. We observe an elbow from the third to fourth eigenvectors.

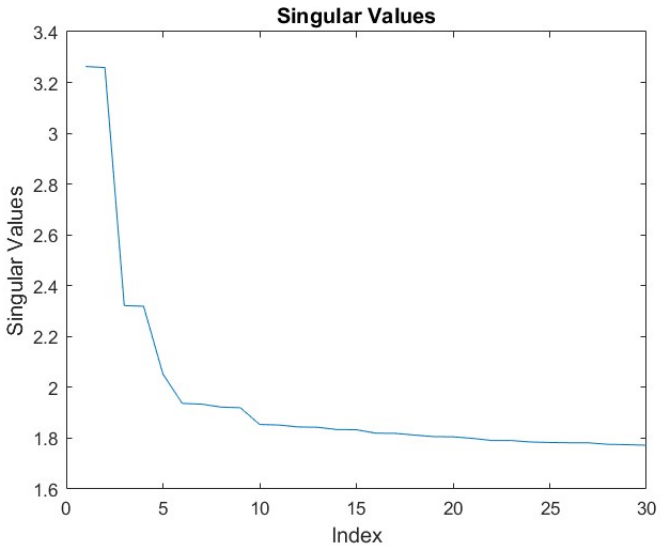

Figure 7: Scree Plot of Step 2

Figure 8 shows the eigenvectors for the residuals. A period of approximately 12 months is observed in the first and second eigenvectors, accounting for the majority of the seasonal component, and approximately 6 months in the third and fourth eigenvectors. This indicates that there is a very strong annual cycle in the average temperature in Boulder, as originally expected.

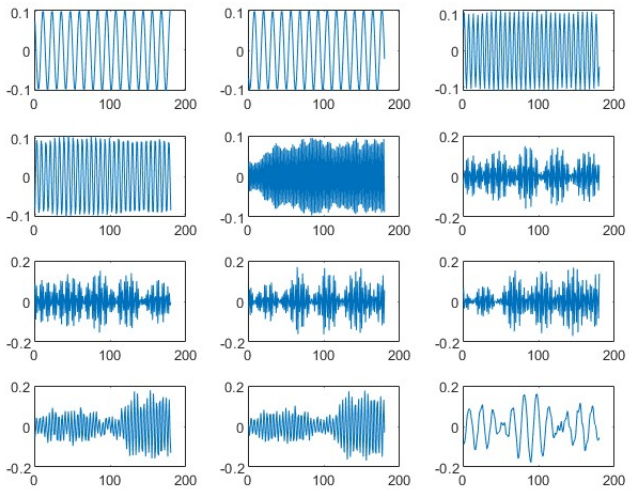

Figure 8: Eigenvector of Step 2

Figure 9 shows the p-vertex polygons for the 1 vs. 2 and 3 vs. 4 plots. From this we can observe a 12-month cycle and a 6-month cycle from the 1 vs. 2 and 3 vs. 4 p-vertex polygons, respectively. These p-vertex polygons represent sine/cosine series with zero phase difference and equal amplitudes [2].

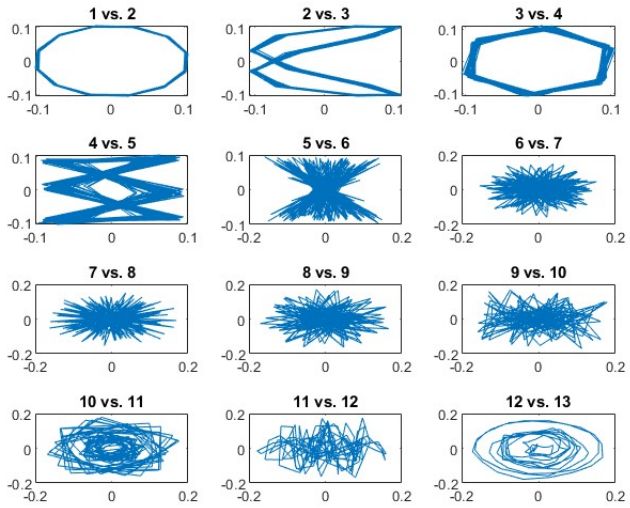

Figure 9: Eigenvector Pairs of Step 2

Thus, the first, second, third, and fourth components contribute to the seasonal component. The seasonal component is shown in figure 10.

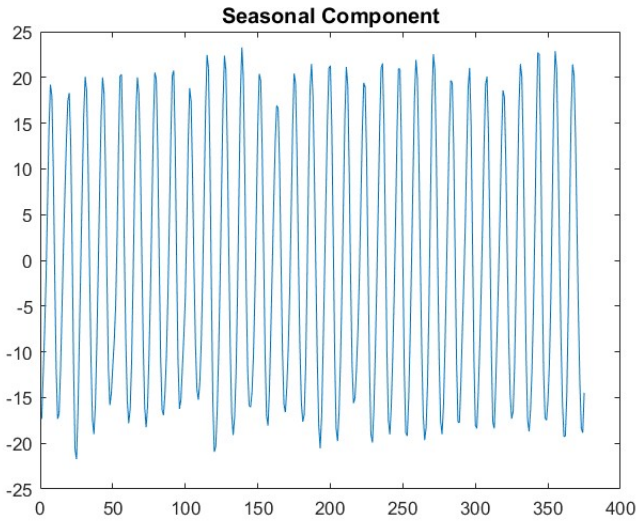

Figure 10: Seasonal Component of Monthly Average Temperature Time Series

5.2 Applying SSA to Housing Price Index in Mountain Region

The next part of the analysis is to implement the Basic SSA algorithm on the monthly HPI values recorded in the Mountain region from January, 1991 to March, 2022. Figure 11 shows a plot of the monthly HPI time series.

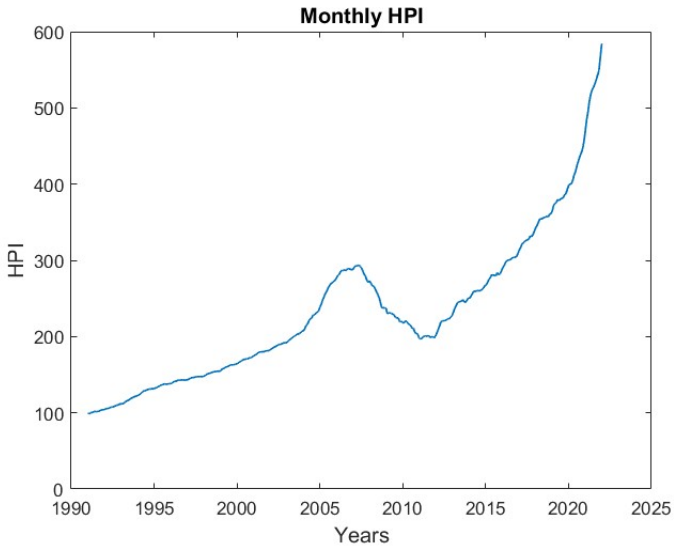

Figure 11: Monthly Housing Price Index over 30 Years

Similarly to the previous time series, the process starts by extracting the trend. Figure 12 shows reconstructed time series corresponding to the long-term trend of the data.

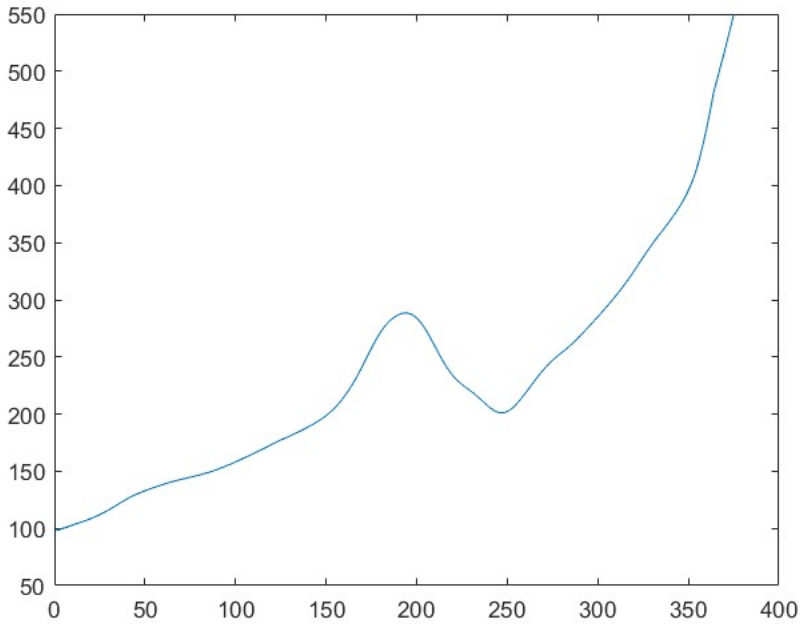

Figure 12: Monthly HPI Trend

These lines are almost identical, however the trend line is noticeably smoother. This can be attributed to the fact that any seasonal components/noise have been separated from the trend, but contribute very little to the overall time series.

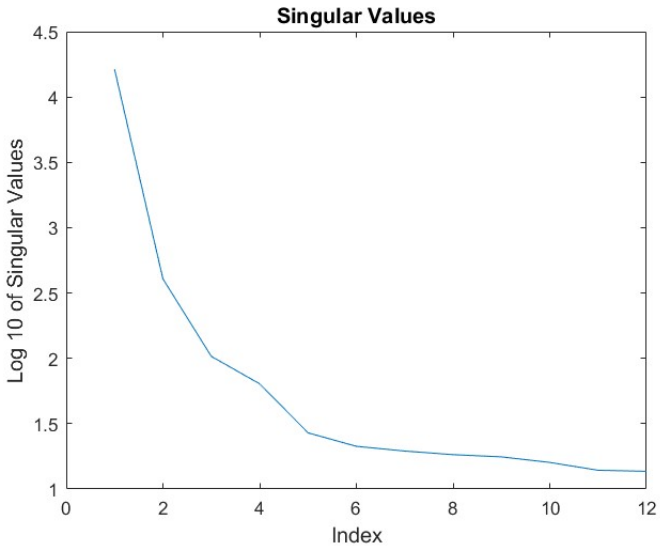

Figure 13: Scree Plots for Monthly HPI Step 1

After performing singular value decomposition, we plot the singular values in Figure 13. There doesn't appear to be any evident "elbow" in the scree plot, but we know that the first and second components contribute heavily to the overall trend because they are multiple orders of magnitude greater than the rest. Figure 14 shows the corresponding eigenvectors.

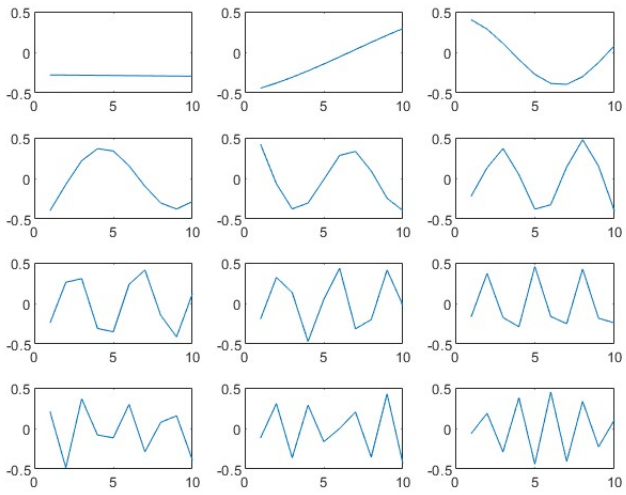

Figure 14: Eigenvectors for Monthly HPI Step 1

The residuals plot is shown in Figure 15, illuminating the incompatibility of this data set with Basic SSA. At first glance, something resembling a 12-month period can be observed; however, it is not consistent. Furthermore, it appears that the fluctuations grow in amplitude.

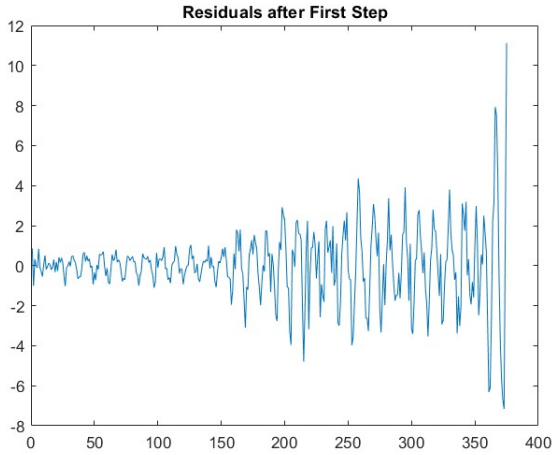

Figure 15: Residuals for Monthly HPI

Following the same process for extracting the seasonal component as in section 5.1. The eigenvectors are plotted in Figure 16. Figure 17 shows the corresponding eigenvector pairs.

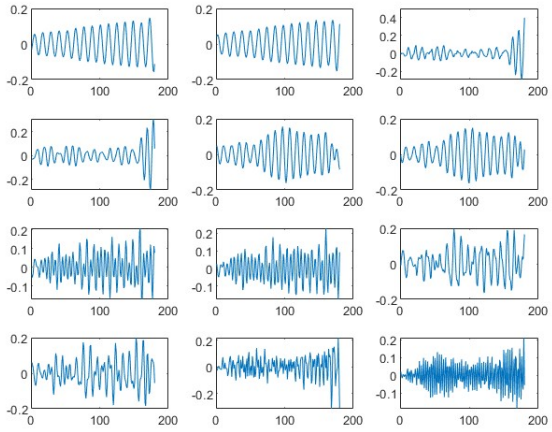

Figure 16: Eigenvectors for Seasonal Component

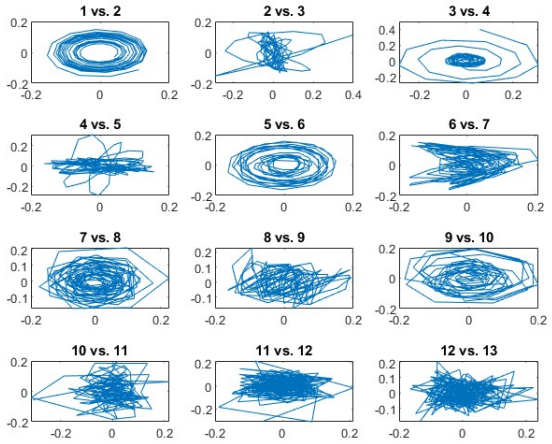

Figure 17: Eigenvector Pairs for Monthly HPI

As we can see, most of the eigenvector pairs can be attributed to noise in the data. We do observe a p-vertex polygon for the 1 vs. 2 plot, indicating a 12-month cycle in the HPI; however, this contributes very little to the overall time series.

6 Discussion and Conclusions

6.1 Conclusion

Singular Spectrum Analysis (SSA) is a tool used to extract trends, seasonal components, and noise from a time series. Using Matlab, SSA was applied to two sets of data: the average monthly temperature in Boulder, Colorado and the housing price index of the mountain region in the U.S.A. The data corresponding to the average temperature in Boulder shows a periodic cycle every twelve months before applying SSA, which made choosing a window length trivial. After plotting the singular values, the "elbow" in the graph is seen between the second and third components showing the overall trend is contributed to the first component. When graphing the eigenvectors, the first eigenvector shows constant values, indicating there is no long-term trend, however a seasonal component is extracted after applying SSA to the residuals. Two seasonal trends are found; a twelve month cycle and a six month cycle. These trends are seen from the eigenvector pairs in Figure 9.

After applying SSA to a time series where the results can be predicted before using the algorithm, SSA was then applied to a more complicated time series–the Housing Price Index in the mountain region. The results are more unpredictable due to a large number of confounding variables in the housing market. Extracting a seasonal component proved to be more difficult with the HPI as seen in Figures 11 and 12, since the reconstructed time series show that the long-term trend is smoother, though quite similar to the original time series. This shows the long-term trend is considerably more prevalent than the seasonal component and excess noise. When extracting the seasonal component and comparing the eigenvectors in Figure 17, eigenvectors 1 and 2 show there is a rough 12-month cycle, however because the p-vertex polygon of the "1 vs. 2" plot is not a tight-knit polygon, we cannot definitively say that the 12-month period is consistent. We suspect that basic SSA yielded inconclusive results for the monthly HPI data due to the fact that the health of the economy is not entirely dependent on time. There are many things that could contribute to fluctuations in the housing market value besides time, such as different economic policies under various presidents, statistical anomalies such as the Global Financial Crisis in 2007, foreign conflicts, etc. Applying the basic SSA algorithm to a time series that appears periodic and has minimal external factors allows for the algorithm to extract well-defined seasonal components. However, when using SSA to extract trends that have many external factors associated with the fluctuations, detecting a consistent seasonal trend proved to be difficult.

6.2 Future Work

In future work, we could leverage a preprocessing technique called *Single centering* to improve the accuracy of our SSA procedure on the temperature data. Single centering involves creating a matrix A and adding it to the SVD decomposition of the original trajectory matrix, such that $X = A + X_1 + \ldots + X_d$. This matrix A is found by averaging the rows of X and then multiplying each row by its respective average. Basic SSA with single centering is preferred when the data have a constant trend and a component oscillating around zero, as found in our temperature data[1]. *Double centering* is a similar process but it essentially also takes into account the averages of the columns of our original trajectory matrix. Basic SSA with double centering has been shown to outperform standard basic SSA in cases where the data have a linear trend and a component oscillating around zero, as in our HPI set[1].

To further our understanding of parameter choices in Basic SSA, we could research some of the statistical methods which are often used throughout the procedure to optimize results. For example, applications

of basic SSA often utilize periodograms in determining how to group eigentriples into matrices resembling a Hankel matrix[1]. Additional insight behind the theory of the grouping method could be gleaned through research about and understanding of multivariate geometry.

SSA is a very powerful tool with several possible extensions to the basic methodology; Separability of time series components can be quantified, forecasts of future values can be made, and signal in red noise can be detected[5].

7 References

[1] Golyandina, Nina E., and Zigljavskij Anatolij Aleksandrovic. Singular Spectrum Analysis for Time Series. Springer, 2013.

[2] Cheng Deng. (2014). Time Series Decomposition Using Singular Spectrum Analysis. [Master's Thesis, East Tennessee State University]

[3] fhfa.gov (Federal Housing Finance Agency Website)

[4] https://psl.noaa.gov/boulder/Boulder.mm.html (NOAA Physical Sciences Laboratory)

[5] Golyandina, N. (2010). On the choice of parameters in Singular Spectrum Analysis and related subspace-based methods. Statistics and Its Interface 3 259–279.

8 Appendix

MATLAB Code:

```
//Initializing L(window length), N(size of original time series), and K(K=N-L+1)
L = 12;
N = 375;
K = N - L + 1;


//Importing temperature data from "tempData.txt"
tempData = importData("tempData.txt");

//Storing tempData in vector Y. This is our raw time series
Y = tempData{:,:};

//Temperature original time series
figure(1);
t = linspace(1991,2022,375);
plot(t,Y,'Linewidth',1);
title('Monthly Temperature Averages','FontSize',12);
xlabel('Years','FontSize',12);
ylabel('Temperature (Farenheit)','FontSize',12);
hold on;


//Creating trajectory matrix X
X = generateX(Y,L,K);


[U,S,D] = svd(X);


//Storing singular values in singVals
singVals = zeros(1,L);
for i = 1:L
singVals(i) = S(i,i);
end


//Plotting singVals
figure(2);
hold off;
plot(singVals);
title('Singular Values', 'FontSize', 12);
xlabel('Index','FontSize',12);
ylabel('Singular Values','FontSize',12);
hold on;


//Transposing
U = U';
```

```
//Plotting each eigenvector from U
//Eigenvectors from SSA of monthly temp data
figure(3);
hold off;
nexttile;
plot(1:12,U(1,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(2,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(3,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(4,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(5,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(6,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(7,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(8,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(9,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(10,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(11,:));
axis([0 10 -0.5 0.5]);
nexttile;
plot(1:12,U(12,:));
axis([0 10 -0.5 0.5]);
hold on;


//Creating elementary X matrices
X1 = decomposeX(S,D,U,1);
X2 = decomposeX(S,D,U,2);
X3 = decomposeX(S,D,U,3);
X4 = decomposeX(S,D,U,4);
X5 = decomposeX(S,D,U,5);
X6 = decomposeX(S,D,U,6);
X7 = decomposeX(S,D,U,7);
```

```
X8 = decomposeX(S,D,U,8);
X9 = decomposeX(S,D,U,9);
X10 = decomposeX(S,D,U,10);
X11 = decomposeX(S,D,U,11);
X12 = decomposeX(S,D,U,12);

//Reconstructing time series with diagonal averaging
g1 = diagonalAveraging(X1, L, K, N);
g2 = diagonalAveraging(X2, L, K, N);
g3 = diagonalAveraging(X3, L, K, N);
g4 = diagonalAveraging(X4, L, K, N);
g5 = diagonalAveraging(X5, L, K, N);
g6 = diagonalAveraging(X6, L, K, N);
g7 = diagonalAveraging(X7, L, K, N);
g8 = diagonalAveraging(X8, L, K, N);
g9 = diagonalAveraging(X9, L, K, N);
g10 = diagonalAveraging(X10, L, K, N);
g11 = diagonalAveraging(X11, L, K, N);
g12 = diagonalAveraging(X12, L, K, N);


//Plotting each reconstructed time series
//Reconstructed time series from SSA of monthly temp data
figure(4);
hold off;
nexttile;
plot(1:375,g1);
nexttile;
plot(1:375, g2);
nexttile;
plot(1:375, g3);
nexttile;
plot(1:375, g4);
nexttile;
plot(1:375, g5);
nexttile;
plot(1:375, g6);
nexttile;
plot(1:375, g7);
nexttile;
plot(1:375, g8);
nexttile;
plot(1:375, g9);
nexttile;
plot(1:375, g10);
nexttile;
plot(1:375, g11);
nexttile;
plot(1:375,g12);


hold on;
figure(5);
```

```
hold off;


//Creating and plotting residuals
z = zeros(1,N);
for i = 1:N
z(i) = Y(i) - g1(i);
end
plot(1:N,z);
title('Residuals after First Step');
hold on;


//Updated window length to close to N/2
L = 180;
K = N - L + 1;


//Trajectory matrix for step 2
Z = generateX(z,L,K);


[U2,S2,D2] = svd(Z);


U2 = U2';


singVals2 = zeros(1,L);
for i = 1:L
singVals2(i) = S2(i,i);
end


//Plotting the Scree plot for the Residuals
figure(6);
hold off;
plot(singVals2);
title('Singular Values', 'FontSize', 12);
xlabel('Index','FontSize',12);
ylabel('Singular Values','FontSize',12);
hold on;


//Plotting the Eigenvectors of the Residuals
figure(7);
hold off;
nexttile;
plot(1:L,U2(1,:));
nexttile;
plot(1:L,U2(2,:));
nexttile;
plot(1:L,U2(3,:));
```

```
nexttile;
plot(1:L,U2(4,:));
nexttile;
plot(1:L,U2(5,:));
nexttile;
plot(1:L,U2(6,:));
nexttile;
plot(1:L,U2(7,:));
nexttile;
plot(1:L,U2(8,:));
nexttile;
plot(1:L,U2(9,:));
nexttile;
plot(1:L,U2(10,:));
nexttile;
plot(1:L,U2(11,:));
nexttile;
plot(1:L,U2(12,:));
hold on;
```

//**Plotting Paired Eigenvector Plots**

```
figure(8);
hold off;
nexttile;
plot(U2(2,:),U2(1,:));
title('1 vs. 2');
nexttile;
plot(U2(3,:),U2(2,:));
title('2 vs. 3');
nexttile;
plot(U2(4,:),U2(3,:));
title('3 vs. 4');
nexttile;
plot(U2(5,:),U2(4,:));
title('4 vs. 5');
nexttile;
plot(U2(6,:),U2(5,:));
title('5 vs. 6');
nexttile;
plot(U2(7,:),U2(6,:));
title('6 vs. 7');
nexttile;
plot(U2(8,:),U2(7,:));
title('7 vs. 8');
nexttile;
plot(U2(9,:),U2(8,:));
title('8 vs. 9');
nexttile;
plot(U2(10,:),U2(9,:));
title('9 vs. 10');
nexttile;
plot(U2(11,:),U2(10,:));
```

```
title('10 vs. 11');
nexttile;
plot(U2(12,:),U2(11,:));
title('11 vs. 12');
nexttile;
plot(U2(13,:),U2(12,:));
title('12 vs. 13');
hold on;


function Xi = decomposeX(S,D,U,i)
Xi = (S(i,i) * D(:,i) * U(i,:))';
end


//Function to perform the diagonal averaging algorithm
function g = diagonalAveraging(X, L, K, N)
j = 1;
g = zeros(1,N);
for k = 1:L
for j = 1:k
g(k) = (g(k) + (X(j,k-j+1)));
end
g(k) = g(k) / k;
end
for k = L+1:K-1
for j = 1:L
g(k) = (g(k) + (X(j, k-j+1)));
end
g(k) = g(k) / L;
end


for k = K:N
for j = (k - K + 1):L
g(k) = (g(k) + (X(j,k-j+1)));
end
g(k) = g(k) * (1/(N-k+1));
end

end

//Generates the X matrix given the time series, Y
function X = generateX(Y, L, K)
for i = 1:L
for j = 1:K
X(i,j) = Y(i + j - 1);
end
end

end
```