Two-Dimensional Empirical Mode Decomposition and its Applications

Wesley Brown

April 28, 2022

1 Introduction

Analyzing wave functions is integral for numerous medical practices. In order to understand a patient's condition, physicians must have an accurate reading of the body's biological signals. Signals such as electrocardiograms (ECG), respiratory signals, blood pressure and circadian rhythm can become muddled by interference, making an accurate reading more difficult. For the sake of a patient's health, methods of circumventing these problems become necessary.

To solve this problem, interference signals must be removed from scans. There are multiple methods by which to remove this interference, including Wavelet Transforms, Fourier Transforms and Empirical Mode Decomposition (EMD). Each of these methods takes a signal input and outputs multiple component signals sorted according to frequency content. Since interference is largely a high frequency phenomenon, it may be extracted from these components.

With multiple methods of decomposing signals for practical and potentially lifesaving applications, we must ask the question: which of these methods works most efficiently? Fourier Transforms efficiently decompose signals into waves with constant periods and amplitudes across a period of time, making it less useful for analyzing complex signals. While Wavelet Transforms are suitable to analyze signals with varying periods and amplitudes, the calculations are dependent on a particular wavelet and are complicated to extend to two-dimensional signals. To show why EMD is a notable method of signal analysis, we will look at these other time frequency (TF) methods in section 2 as well as EMD in section 3.

Beyond observing these one-dimensional signals from the body, it is also worthwhile to analyze two-dimensional images obtained by medical scans such CT or MRI scans. Just like one-dimensional signals, these images can be plagued by interference. In section 4 we will discuss how to adapt EMD for two-dimensional applications.

There are also methods to improve both one-dimensional and two-dimensional EMD. The EMD algorithm utilizes interpolation to decompose signals. Out of two methods of interpolation, cubic spline interpolation (CSI) and shape preserving interpolation (SPI), the latter has been shown to be more efficient for one-dimensional applications [7]. However, in sections 5 and 6, we will test which method works best for two-dimensional applications and analyze how the results show that shape preserving interpolation works more efficiently for two-dimensional applications.

2 Background

2.1 Fourier Transforms

The Fourier transform analyzes the features of a wave signal by treating that signal as the sum of more simple sine and cosine functions. The Fourier transform is a function which takes an input, $g(t)$, of some given signal. The transform winds a signal in a circular motion at some variable frequency. This frequency constantly changes while we record the signal's center of mass using real and complex coordinates. The resulting output is a function with the winding frequency as an input. The output $\hat{g}(t)$ is a complex number which corresponds to the strength of a frequency in the original function. With these frequencies recorded, we can then separate the original signal into more simple sine and cosine functions [1]. The function is described by the following equation:

$$\hat{g}(t) = \int_{-\infty}^{\infty} g(x) e^{-2\pi i x t} \, dx$$

where e is the element that performs the winding effect on the signal.

Because the Fourier transform assumes that the amplitude and phase of a wave component remains constant across time, it is limited in its ability to analyze frequency and more complex signals like those produced by heart rate and other vital signs. Therefore, we need a method which accounts for changing amplitudes and phases of a given signal component. Empirical Mode Decomposition (EMD) functions as such a time-frequency (TF) method [8].

2.2 Wavelet Transforms

Wavelet Transforms use multiresolution analysis to obtain time-frequency representations of signals. The process begins with a window function known as a mother wavelet, which is then dilated or compressed and shifted as necessary. By multiplying a given signal with these dilated and shifted versions of the mother wavelet, we are able to localize the signal before examining its frequency content. In this way, we may obtain multiple frequencies across the time axis, resolving the Fourier Transforms limitation. [3].

While the Wavelet Transform is able to analyze signals with variable amplitudes and phases, the calculations are quite complex and depend on the arbitrarily chosen mother wavelet. These complex calculations make applications in two dimensions computationally intensive and inefficient.

For our purposes we need a simple, data-defined decomposition scheme which allows for diverse frequencies and amplitudes within a single component and which may be easily extended to two-dimensional applications. Empirical Mode Decomposition functions as such a scheme.

3 EMD

EMD utilizes a multi-step iterative process to separate signals into components. Broadly, the process uses interpolation methods on a signal to obtain a new function which represents the highest frequency oscillations present in the signal. That function is subtracted from the original signal. We repeat the interpolation step until some stopping criterion is met. The resulting output functions can be viewed as the components of the original signal [6].

This process begins by considering local oscillations on the input signal, $x(t)$. We first look at the local detail which corresponds to the oscillation between two extrema. This local detail lets us define the high frequency values of the input signal. If we look at two consecutive minima, say t_- and t_+, then we can define the local detail as $\{d(t), t_- \leq t \leq t_+\}$.

With a defined set of high frequency signals, we can also determine the set of local low frequency signals, the local trend $m(t)$, such that $x(t) = m(t) + d(t)$ for $t_- \leq t \leq t_+$ [6]. This process can be done across all oscillations of $x(t)$. Furthermore, the process can be reiterated to receive more finely detailed components, or intrinsic mode functions (IMF's), of $x(t)$.

Because EMD relies on local oscillations along the time axis rather than broad trends, it can be more useful for determining the components of more complex and inconsistent functions, unlike Fourier Transforms and Wavelet Transforms

We can summarize the EMD process into these six steps [6]:

1. Determine the extrema along the time axis of input $x(t)$.

2. Interpolate across the maxima and the minima, respectively, to yield upper and lower envelopes of the signal.

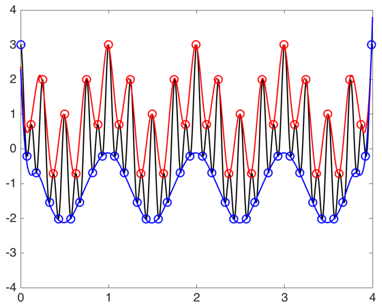
3. Find the mean $m(t)$ between these two functions at every point t.

4. Subtract, or sift, $m(t)$ from $x(t)$.

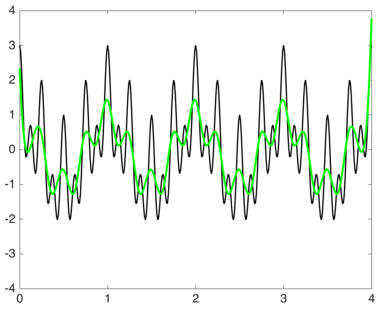
5. Repeat steps 2-4 on $x(t) - m(t)$ until we meet some predetermined stopping criterion. The output is an IMF $f_1(t)$.

6. Repeat steps 2-5 on $x(t) - f_1(t)$ to find the remaining IMF's, $f_2(t), ..., f_n(t)$, where $f_n(t)$ is whatever remains after all substantial signal data has been sifted.

Figure 1 illustrates the interpolation and sifting process.



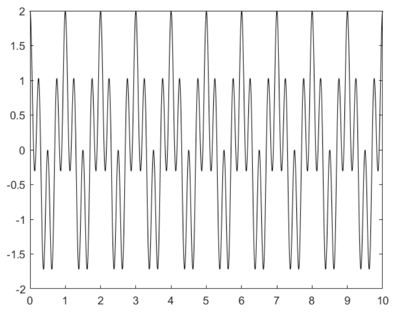
(a) Using interpolation, we form continuous function across all local maxima and minima, the red function and the blue function respectively.



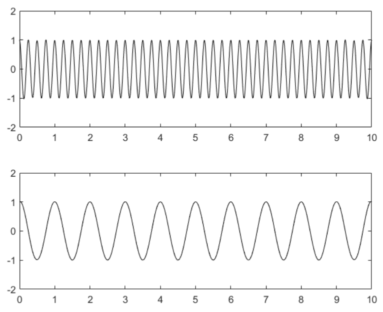
(b) Taking the mean of the red and blue function across every point on the time axis, we receive the green function. Subtracting that function from $x(t)$, the black input signal, we perform one sift.

Figure 1: Illustration of the interpolation, mean, and sifting process where the input signal $x(t)$, is the black function.

For a concrete example of EMD in practice, we can look to the following example. The following figure shows an input function on the left (a) which we define as $f(t) = \cos(8\pi t) + \cos(2\pi t)$. Our implementation of the EMD algorithm correctly estimated the high and low frequency IMF's (b), outputting two approximations of the functions, $g(t) = \cos(8\pi t)$ and $h(t) = \cos(2\pi t)$.



(a) The input function



(b) The output of high frequency and low frequency intrinsic mode functions (IMF's)

Figure 2: Implementation of EMD on a simple one-dimensional wave function.

Concerning the EMD algorithm, there are a couple questions we must ask. Firstly, during step 5, how do we know how many times to sift before we obtain an accurate IMF? In other words, How many times should we find the mean $m(t)$ and subtract it from $x(t)$? The other question relates to step 6. How do we know when we've obtained all the IMF's? Let's address the first question concerning stopping criteria.

3.1 Stopping Criteria

Because EMD considers the oscillations between extrema, it follows that we would have to analyze oscillations to determine when an IMF has been found. We first must more concretely define what makes an IMF.

There are two characteristics to ascribe to IMF's [7].

1. The upper and lower envelopes of the IMF should approximately be symmetric across the time axis.

2. The number of extrema on the IMF should be equal to its number of x-crossings give or take one.

The IMF must maintain these characteristics for a set number of sifts to be considered an IMF. We call this criterion the S-Number Method [5]. For the purposes of these tests, we let $S = 10$, meaning that sifting would stop if the output had the characteristics of an IMF for 10 consecutive sifts. Sifting would stop altogether if more than 15 consecutive sifts had to take place. We chose $S = 10$ because our tests showed this number to work best for efficiency and accuracy.

3.2 Determining the Number of Components in a Signal

There must be some method of determining when there are enough components of $x(t)$. There are a few ways to determine whether all notable IMF's have been found. Firstly, we can determine when to stop sifting the remainder based on its monotonicity. If there are no more local extrema, then there are no more high or low frequency components to find. Therefore, we would have all the components of the input signal.

We can also check how close the remainder is to monotonicity based on how many extrema are left on the remainder function. If there is only one local maxima or minima left on the remainder, making for a parabola shaped function, then the function is relatively close to monotonicity, meaning all high and low frequencies of the input have been found.

We can also determine whether to continue sifting by checking the remainder's maximum amplitude. To perform this check, we simply choose some small value, say 0.1, after iterating on the input $x(t)$, if each point along the remainder of $x(t)$ is less than 0.1 then we can determine that the remainder is small enough to be considered a zero function. We can then stop sifting.

4 Two-Dimensional EMD

While there exist multiple ways of generalizing EMD for two dimensions (e.g., multivariate interpolation), we are focusing on the multidimensional ensemble EMD method described by Chih-Sung Chen [2]. This method uses the same algorithm as one-dimensional EMD, by applying the algorithm to individual rows and columns of an image. After performing EMD on each individual row and column of an image, we combine the resulting IMF's using a predetermined configuration. The multidimensional ensemble EMD (MDEEMD) describes the configuration used for our testing.

First, we represent the input image as an $i \times j$ matrix such that

$$X(i,j) = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} \\ \vdots & \vdots & \vdots & \vdots \\ x_{i,1} & x_{i,2} & \cdots & x_{i,j} \end{pmatrix}.$$

We now have i one-dimensional horizontal signals and j one-dimensional vertical signals. We can run EMD on each of these signals as we normally would for one-dimensional signals. First we run EMD on each of the Horizontal signals. We suppose that each signal has m components (intrinsic mode functions). Then, we have the following array of IMF's which we will call the RX matrices where R denotes row decomposing [2]:

$$RX(1,i,j) = \begin{pmatrix} rx_{1,1,1} & rx_{1,1,2} & \cdots & rx_{1,1,j} \\ rx_{1,2,1} & rx_{1,2,2} & \cdots & rx_{1,2,j} \\ \vdots & \vdots & \vdots & \vdots \\ rx_{1,i,1} & rx_{1,i,2} & \cdots & rx_{1,i,j} \end{pmatrix}$$

$$RX(2,i,j) = \begin{pmatrix} rx_{2,1,1} & rx_{2,1,2} & \cdots & rx_{2,1,j} \\ rx_{2,2,1} & rx_{2,2,2} & \cdots & rx_{2,2,j} \\ \vdots & \vdots & \vdots & \vdots \\ rx_{2,i,1} & rx_{2,i,2} & \cdots & rx_{2,i,j} \end{pmatrix}$$

$$\vdots$$

$$RX(m,i,j) = \begin{pmatrix} rx_{m,1,1} & rx_{m,1,2} & \cdots & rx_{m,1,j} \\ rx_{m,2,1} & rx_{m,2,2} & \cdots & rx_{m,2,j} \\ \vdots & \vdots & \vdots & \vdots \\ rx_{m,i,1} & rx_{m,i,2} & \cdots & rx_{m,i,j} \end{pmatrix}.$$

Each RX matrix contains the m^{th} component of each signal in $X(i,j)$. Every matrix represents a different component of X. Every row in a particular matrix represents a different row of X. In other words, the first row of the matrix $RX(m,i,j)$ is the m^{th} component of the first row of the matrix $X(i,j)$, The second row of the matrix $RX(m,i,j)$ is the m^{th} component of the second row of the matrix $X(i,j)$ [2], and so on such that

$$X(i,j) = RX(1,i,j) + RX(2,i,j) + \cdots + RX(m,i,j).$$

With the signal broken down by rows, we now need to break down each of those components by column. Suppose that there will be n components for each column. Similar to the previous step, we'll put together an array of CRX matrices $CRX(m, n, i, j)$, where C denotes column-wise decomposition, m denotes the row component being worked on, and n denotes the column component.

Like the RX matrices, it follows that

$$RX(1, i, j) = CRX(1, 1, i, j) + CRX(1, 2, i, j) + \cdots CRX(1, n, i, j)$$
$$RX(2, i, j) = CRX(2, 1, i, j) + CRX(2, 2, i, j) + \cdots CRX(2, n, i, j)$$
$$\vdots$$
$$RX(m, i, j) = CRX(m, 1, i, j) + CRX(m, 2, i, j) + \cdots CRX(m, n, i, j).$$

We can represent each CRX matrix as the following:

$$CRX(m, n, i, j) = \begin{pmatrix} crx_{1,1,i,j} & crx_{2,1,i,j} & \cdots & crx_{m,1,i,j} \\ crx_{1,2,i,j} & crx_{2,2,i,j} & \cdots & crx_{m,2,i,j} \\ \vdots & \vdots & \vdots & \vdots \\ crx_{1,n,i,j} & crx_{2,n,i,j} & \cdots & crx_{m,n,i,j} \end{pmatrix}$$

The described matrix contains the m row-wise decomposition components and their n column-wise decomposition components [2].

With all the row-wise and column-wise components obtained, we must recombine them in an order that outputs complete two-dimensional components ordered according to frequency. The $m \times n$ CRX matrix itself contains only partial representations of a two-dimensional IMF. Because EMD first filters out high frequency signals, the first row and first column of the CRX matrix will contain comparable elements. Therefore, by adding together the elements of the first row and first column, we can obtain a full two-dimensional IMF. The same principle can be applied to the rest of the elements in the CRX matrix. We therefore have a configuration which we can represent with the following illustration.

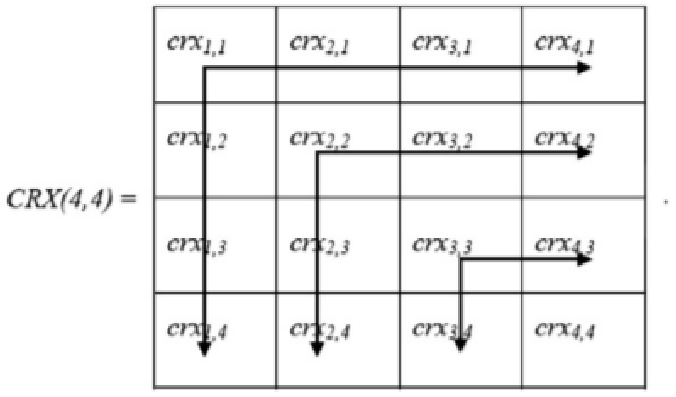


Figure 3: Configuration for finding two-dimensional IMF's [2]

Using this figure as a basis, we then have the complete components of the original image. For example, when $m = n = 4$, the CRX matrices will be combined into the following four IMF's:

$C_1 = \sum_{m=1}^{4} crx_{m,1} + \sum_{n=1+1}^{4} crx_{1,n} = crx_{1,1} + crx_{2,1} + crx_{3,1} + crx_{4,1} + crx_{1,2} + crx_{1,3} + crx_{1,4}$

$C_2 = \sum_{m=2}^{4} crx_{m,2} + \sum_{n=2+1}^{4} crx_{2,n} = crx_{2,2} + crx_{3,2} + crx_{4,2} + crx_{2,3} + crx_{2,4}$

$C_3 = \sum_{m=3}^{4} crx_{m,3} + \sum_{n=3+1}^{4} crx_{3,n} = crx_{3,3} + crx_{4,3} + crx_{3,4}$

and

$C_4 = \sum_{m=4}^{4} crx_{m,4} + \sum_{n=4+1}^{4} crx_{4,n} = crx_{4,4}$.

We can use the following general equation to represent two-dimensional components:

$C_l = \sum_{i=l}^{m} crx_{i,l} + \sum_{j=l+1}^{n} crx_{l,j}$

where l represents which IMF is being constructed based on the level of the CRX matrix currently being observed [2].

This same process can be generalized to any dimension, however, for the purposes of our testing, we will be primarily looking at two-dimensional applications.

5 Interpolation Schemes

5.1 Cubic Spline Interpolation

Cubic spline interpolation is the widely used method for interpolation between a set of points. It's largely favored because of its ability to output a smooth piece-wise polynomial that passes through the set of points.

To accomplish this smoothness, cubic spline interpolation puts a number of constraints on the output piece-wise polynomial. Given a set of points (x_0, y_0), (x_1, y_1),..., (x_n, y_n) cubic spline interpolation specifies a method of constructing cubic polynomials $s_0(x), s_1(x), ..., s_{n-1}(x)$ such that each $s_i(x)$ interpolates (x_i, y_i) and (x_{i+1}, y_{i+1}). The cubic spline S is then defined by

$$S(x) = \begin{cases} S_0(x), & x \in [x_0, x_1], \\ \vdots & \vdots \\ S_{n-1}(x), & x \in [x_{n-1}, x_n] \end{cases} .$$

In addition, a few characteristics must be true to ensure that S is sufficiently smooth. In order for the set of functions S to be continuous it must be true that $s_0(x_1) = s_1(x_1), s_1(x_2) = s_2(x_2), ..., s_{n-2}(x_{n-1}) = s_{n-1}(x_{n-1})$. In order to achieve smoothness in addition to continuity, the first and second order derivatives of S must also be continuous at the knots $x_1, ..., x_{n-1}$. With these constraints, the cubic spline interpolation method calculates a set of cubic polynomial functions that all fit together smoothly.

For each of these cubic polynomials, we need coefficients for every term of the polynomial. The conditions listed above allow us to find those coefficients, giving us each cubic polynomial of the form $s_j(x) = a_j(x - x_j)^3 + b_j(x - x_j)^2 + c_j(x - x_j) + d_j$. Because we are dealing with dozens and potentially hundreds of data points and polynomials, we perform these calculations using matrices. These calculations can be performed using a built in function in Matlab.

While cubic spline interpolation mostly yields good results for two-dimensional EMD applications, there are a couple potential drawbacks to its use. Because cubic spline interpolation does not take into account changes in concavity, it may read false extrema in the upper and lower envelopes, resulting in a less accurate sift [2]. Therefore, we must find and test an interpolation method which will output more accurate envelopes for use in the sifting step. Shape preserving interpolation, intially proposed by L.L. Schumaker in [4], may be such a method.

5.2 Shape Preserving Interpolation

Shape preserving interpolation seeks to construct each spline in terms of normalized quadratic B-splines, allowing for more smoothness and consistent concavity across splines. To this end, shape preserving interpolation assigns the following knot sequence:

$$\mathbf{x} : x_{-2} = x_{-1} = x_0 < x_1 < x_2 < ... < x_{2n-1} < x_{2n} = x_{2n+1} = x_{2n+2}.$$

The interpolation points coincide the knots with even indices x_{2i}. Therefore, we account not only for the interpolation points but also for particular points in between that will allow us to better specify the shape of the resulting spline. Given our set of points (x_0, y_0), (x_1, y_1),..., (x_n, y_n), we rename them

$$x_{2i} = x_i,\ y_{2i} = y_i,\ i = 0, ..., n.$$

We define the knots with odd indices x_{2i+1} using the algorithm provided in [4]. Essentially, the algorithm calculates the location of each x_{2i+1} so that the monotonicity and concavity present in the data may be preserved. These calculations are based on the slope of the data at the knots x_{2i} as well as the distances between each $(x_{2i},\ y_{2i})$, $(x_{2i+2},\ y_{2i+2})$.

Once the full knot sequence is specified we construct a spline interpolant over these knots and the given interpolation points, giving us a smooth piece-wise polynomial which will account for any sudden or unexpected shifts in concavity [7]. Because of this method's apparent improvements over cubic spline interpolation, we anticipate it making our EMD implementation more accurate.

5.3 Comparing Schemes

To compare these interpolation methods, we must run a series of tests using both methods. We first start by constructing a two-dimensional image, then overlaying a random interference signal on top of it. We will then apply our two-dimensional EMD scheme to decompose the noise-polluted image into IMF's. This will allow us to estimate the interference (present in the high-frequency IMF's) and remove it. Finally, we will be able to estimate the original signal by summing the noise-free IMF's. For these tests we will use the following image which we polluted with noise using randomly generated numbers:

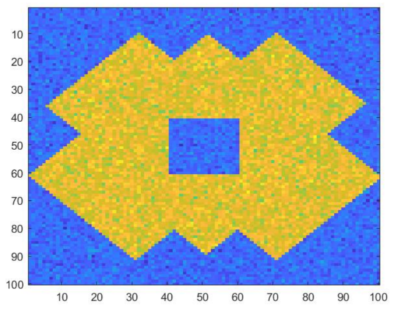


Figure 4: Test image with random interference

We will use five different methods of measurement to capture a full picture of which method works better and why. For these tests we will utilize the root mean square error, a standard error measurement tool used in other literature, which can be represented by the following equation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\text{Original} - \text{Estimate})^2}{N}}, \tag{1}$$

where Original is the input signal without the interference, Estimate is an approximation of the original image found by summing predetermined IMF's, and N is the number of elements, or pixel values, in the image.

We also look to the maximum error, which simply records the greatest difference between a point on the IMF sum and its corresponding point on the original image. We represent the maximum error with the following equation:

$$\text{max error} = \max_{1\leq i\leq N} |\text{Original}(i) - \text{Estimate}(i)|. \tag{2}$$

We also record the minimum error, which similarly records the smallest difference between the input image and the component and can be represented as

$$\text{min error} = \min_{1\leq i\leq N} |\text{Original}(i) - \text{Estimate}(i)|. \tag{3}$$

We also record the mean error, or average difference between the input and component, represented by

$$\text{mean error} = \text{mean}|\text{Original} - \text{Estimate}|. \tag{4}$$

Additionally, we find the standard deviation of the errors between the original image and the component, as represented by

$$\text{SD error} = \text{SD}|\text{Original} - \text{Estimate}|. \tag{5}$$

We finally analyze the error ratio which is defined as

$$\text{ratio} = 1 - \text{corr} \tag{6}$$

where corr is the correlation coefficient between the component and the input image.

To maintain consistency in the results, we will run these tests 100 times, changing the random interference every time we run the tests. The result of the 100 tests will be averaged so that we have a more accurate picture of which interpolation scheme works best. The component we compare to the original image will be the composition of all IMF's except for IMF 1, which is primarily the higher frequency of the interference.

6 Results

Across all measurements shape preserving interpolation proved to be more accurate for decomposing images with random signal interference.

The algorithm was run 100 times to receive an accurate error average. The following table displays the average root mean square error, error ratio, maximum error, minimum error, mean error, and standard deviation of the error where the first row of the table is for Shape Preserving Interpolation (SPI) and the second row is for Cubic Spline Interpolation (CSI).

$S = 10$	RMSE	Error Ratio	Max Error	Min Error	Mean Error	Standard Deviation
SPI	0.132052795	0.029390902	0.883422418	$2.80789 \cdot 10^{-5}$	0.090664773	0.095992222
CSI	0.248829136	0.102321405	1.535730617	$2.37945 \cdot 10^{-5}$	0.180257443	0.171063842

Across all categories, excepting minimum error, shape preserving interpolation performed better than cubic spline interpolation. The small difference in minimum error, only $4.2844 \cdot 10^{-6}$ suggests that the difference of performance in that category is negligible.

The tests show a great difference in the performance of every other category. Cubic spline interpolation has a better RMSE, Error Ratio, Max Error, Mean error, and Standard deviation by nearly 0.1.

7 Examples

The accuracy of shape preserving interpolation can be confirmed visually. The image in **figure 5** shows one image we used in our testing before feeding it through our two EMD variations.

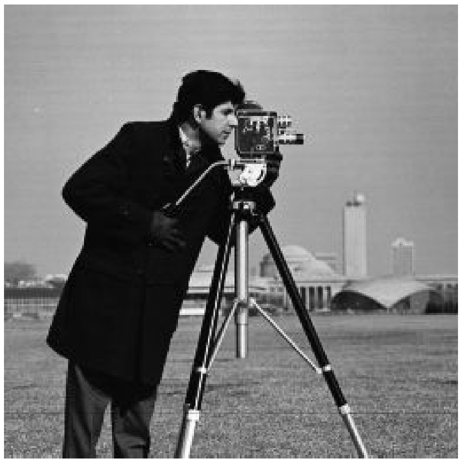
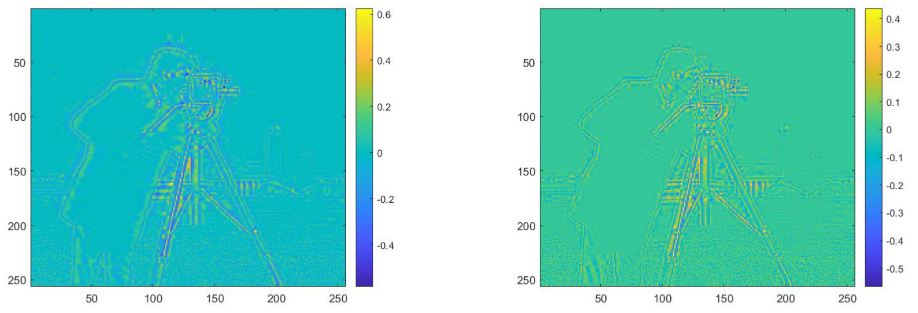


Figure 5: Image before EMD application

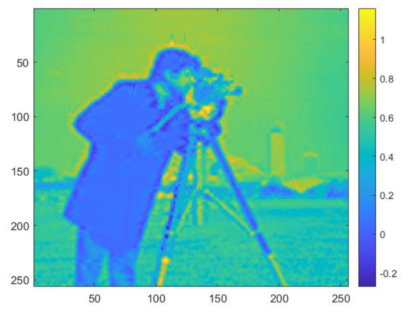
The images in **figure 6** display how shape preserving interpolation makes a better distinction between high and low frequency signals. This improved distinction results in noticeably sharper edge tracing on the image. By contrast, the cubic spline interpolation implementation results in greater blurring around edges.



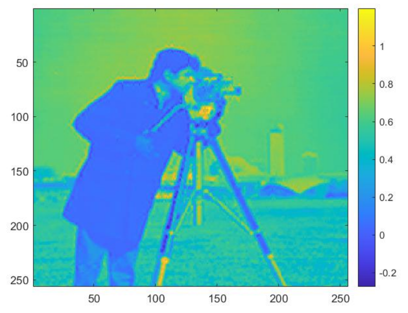
(a) Cubic Spline Interpolation (b) Shape Preserving Interpolation

Figure 6: Visual comparison of shape preserving interpolation and cubic spline interpolation

Similarly, **figure 7** displays a more clean rendering of the lower frequency values found through shape preserving interpolation. The images found using SPI are both clearer are have less image smearing.



(a) Cubic Spline Interpolation



(b) Shape Preserving Interpolation

Figure 7: Visual comparison of shape preserving interpolation and cubic spline interpolation

8 Conclusion

Compared to other methods like Fourier Transforms and Wavelet Transforms, EMD is one of the most efficient and simplest methods for image decomposition. Despite its simplicity, EMD's ability to break down complex signals makes it ideal for use in the medical field. Additionally, shape preserving interpolation is currently the best method for the sifting step of EMD. Shape preserving interpolation improves accuracy for both one-dimensional and two-dimensional applications.

References

[1] Ronald N Bracewell. The fourier transform. *Scientific American*, 260(6):86–95, 1989.

[2] Jeng Yih Chen, Chih-Sung. Two-dimensional nonlinear geophysical data filtering using the multidimensional eemd method. *Journal of Applied Geophysics*, 111:256–270, 2014.

[3] Lu Jiangeng Wu Hau-Tiend Daubechies, Ingrid. Synchrosqueezed wavelet transforms: An empirical mode decomposition-like tool. *Applied and Computational Harmonic Analysis*, 30:243–261, 2011.

[4] Schumaker L. On shape preserving quadratic spline interpolation. 20(4), 1983.

[5] J. C. Nunes, S. Guyot, and E. Deléchelle. Texture analysis based on local analysis of the bidimensional empirical mode decomposition. *Mach. Vision Appl.*, 16(3):177–188, 2005.

[6] Gabriel Rilling, Patrick Flandrin, and Paulo Gonçalves. On empirical mode decomposition and its algorithms. *Proceedings of IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing NSIP-03*, 3, 06 2003.

[7] Maria van der Walt. Empirical mode decomposition with shape-preserving spline interpolation. *Results in Applied Mathematics*, 5.

[8] Maria van der Walt. A survey of time-frequency methods. 2013.