

**Machine learning and statistical forecasting of high
resolution load and wind power data using wavelet
transformations**

Master's Project Report

Presented to the Department of Engineering of Cornell University

**In Partial Fulfillment of the Requirements for Master of Engineering,
Biological and Environmental Engineering**

By

Alankar Sharma

Project Advisor: C. Lindsay Anderson

Degree Date: May 2016

Biographical Sketch

Alankar completed his bachelor's degree in Environmental Engineering from Indian School of Mines, Dhanbad, India in 2015. His collaborations with research institutions and laboratories on environmental data analytics and his interest in exploring the field of clean energy systems encouraged him to pursue Master of Engineering in Biological and Environmental Engineering. Through energy focused coursework, he gained deep insights into engineering and economic principles of renewable and conventional energy systems. He wishes to utilize his data analytics skills towards inter-disciplinary projects encompassing electrical studies, environmental studies and computer programming, which address the current needs of managing the electric grid network and promote renewable energy.

Table of Contents

Abstract	1
Introduction	1
Discrete Wavelet Transform and Multi Resolution Analysis	3
Artificial Neural Networks	7
Autoregressive Integrated Moving Average model	8
Study area description	11
Methodology	12
Preconditioning of data	12
Objectives	14
To evaluate performance of Discrete Wavelet Transforms against other Smoothing and Polynomial Fitting Algorithms through statistical metrics	14
To analyze the optimal decomposition level that can be achieved with lower errors, i.e. optimal white noise removal level	16
Time Resolution Manipulation	17
Forecasting of time series	17
Results	24
To evaluate performance of Discrete Wavelet Transforms against other Smoothing and Polynomial Fitting Algorithms through statistical metrics	24
To analyze the optimal decomposition level that can be achieved with lower errors, i.e. optimal white noise removal level	32
Time Resolution Manipulation	45
Forecasting of time series	46
<i>Artificial Neural Network (ANN)</i>	46
<i>Wavelet fed Artificial Neural Network (WANN)</i>	47
<i>Auto Regressive Integrated Moving Averaging (ARIMA)</i>	51
Conclusions	53
Further Work	54
References	54
Appendix	57

Abstract

The project intends to explore the application of wavelet transformation technique, a signal processing algorithm, towards analyzing wind energy and demand load time series data. High resolution data in energy systems are inherently harder to analyze because of the introduced variability in the time series. This added volatility of data throws off forecasting models. The applied wavelet transformation technique manipulates the time resolution of data along with the frequency component by isolating a base data series along with the characteristic details of the data curve. Wavelet transforms can achieve multiple objectives that are pertinent to the energy data reporting techniques. The use of wavelet transforms are analyzed for altering resolution of load and wind power data and performance of conversions are quantified through multiple metrics. Forecasting methods of energy data are then studied and the use of wavelet transformed data series is observed through a simple feedforward machine learning network model. The machine learning forecasting method is then analyzed with the ARIMA, a statistical forecasting method, to check the superiority of machine learning forecasting models.

Introduction

With the exponentially increasing growth of renewable energy sources into the electric grid, a requirement for the detailed temporal dynamics has emerged in the current scenario. Traditional load forecasting and management of dispatch routines have been performed by following different operational strategies according to the timescale under consideration. For example, scheduled generation resource adjustments can be made in the hour-ahead market time scales according to the ramping behavior of power sources. Load following and up or down regulation are conducted over minute ahead time scales while power frequencies need to be maintained over minute ahead time scales. While highly accurate short term forecasting methods existed previously, the generation side variability of power production throws the models off.

The forecasting processes are complex because the load and wind power profiles are rarely stationary, i.e. they have a significant associated randomness but exhibit a strong seasonal structure to their occurrence. Data patterns repeat over different timescales ranging from daily variations to monthly patterns. Most of the traditional dataset analysis techniques are suited for stationary

processes and might not be appropriate for non-stationary data. Therefore, performance evaluation of such transient processes that manifest non stationarity has been ambiguous. To comprehend characteristics of load and wind power profiles, non-stationary data analysis tools such as wavelet transformations need to be explored.

Wavelet transforms were developed in 1992 by [1], [2] and [3] in the 1990s. Wavelet Transformations are based on convolution of an original time series over an analyzing function called the mother wavelet. Wavelet Transforms present a distinct advantage over other spectral frequency transforms such as Fourier or Short Term Fourier by analyzing simultaneous time frequencies with flexible resolutions aiding in Multi Resolution Analysis of time series data. Thus, Wavelets can be employed to analyze short term events hidden in an intermittent, non-stationary or stationary time series. Thus, application of Wavelets have ranged from image compression techniques, numerical time series analysis, and environmental system's flow identification other than conventional digital signal processing.

The WT becomes a powerful analyzing tool for stationary, non-stationary, intermittent time series, especially, to find out hidden short events inside the time series. Because of its advantages, the WT have been applied in the various fields such as digital signal processing, image coding and compressing, numerical analysis and digital simulation, system and flow identification and so on, and they still are increasingly evolving [4].

Wavelets have found application in power spectra analysis of short term solar power fluctuations [5], analysis of atmospheric data [6] and more recently, [7] analyzed the application of Wavelets for Wind power series and [17] have presented an introductory neural network based forecasting model utilizing Wavelet transforms.

The wavelet transforms generate time series of step changes or deltas of the data distribution at increasingly large time scales. The wavelet approach is unique for discriminating white noise from these deltas by preserving independent information contained in each of these signals. Hence the wavelet analysis can isolate intrinsically different statistical behavior of wind power generation and demand loads in a region. [9][10][11][12][13]

Wind power deltas have been previously investigated at different time scales to show that the distributions peak near the center but the shape fitting the distribution was not found. [14] [12] [13] Typical meteorological wind prediction models have been used to forecast wind speeds that

can be converted to wind power outputs [14] to which variability needs to be added for representing fluctuations accurately at short term scales. The common practice is to assume that the fluctuations follow Gaussian distributions [14][15][16] and add up on the smooth base signal. [7] found that the probability distributions of wind deltas are strongly non-Gaussian, and the data exhibit several properties typical of self-similar statistical processes [18][19]

It was concluded that wind and load correlations are significant at time scale greater than a few minutes and have a strongly influenced by the time-of-day and the season. The magnitude of the wind deltas (estimated by the root-mean-square) increase as a function of time-scale, with the rate of increase well fit by a power-law. Both the frequency distributions and the temporal autocorrelation functions of the wind deltas are independent of time scale. The probability distribution of the wind deltas has an exponential shape in the center, but follows a power-law in the tails. This property is consistent with the physics of atmospheric turbulence, and suggests a physically-motivated approach to extrapolating the statistical characteristics of wind power to large spatial regions and high levels of capacity.

Discrete Wavelet Transform and Multi Resolution Analysis

The Discrete Wavelet Transforms can be used to examine data at different time scale resolutions [20] [21] [22]. A wavelet transform decomposes an input signal onto a basis function that can be localized within the time series and characterized by the scale and the index parameters. To analyze a function $f(t)$ defined on the interval $t \in [0, L]$ using a discrete wavelet transform (DWT), first a wavelet generating function $\psi(t/L)$ is chosen. To be admissible as a generating function ψ must be integrable; by convention it is defined on the interval $[0, 1]$ and centered at $1/2$. A family of orthonormal wavelets is generated from ψ using dilation and translation:

$$\psi_{jk}(t) = (2^j/L)^{\frac{1}{2}} \psi((2^j t/L - k) ; k = 0, \dots, 2^j - 1$$

where j is the index for the scale and k is the index for the location. The function ψ_{jk} is centered at $kL/2^j$, and larger j corresponds to finer scale of variation [23] A square-integrable function $f(t), t \in [0, L]$ can be decomposed in the wavelet basis as,

$$f(t) = \sum_j \sum_k d_{jk} \psi_{jk}(t)$$

where the d_{jk} are a set of real numbers referred to as the wavelet coefficients. As the DWT is orthonormal, the wavelet coefficients satisfy Parseval's theorem

$$\left(\frac{1}{L}\right) \int_0^L |f(t)|^2 dt = \left(\frac{1}{L}\right) \sum_j \sum_{k=0}^{2^j-1} |d_{jk}|^2$$

More intuitively, the DWT can be thought of as a hierarchical sequence of weighted summing and differencing operations. Moving from the finest to the coarsest scales, the DWT operates on the smoothed version of the function at scale j to create a smoothed version at a scale $j - 1$, and a second function capturing the “details” which have been removed by the smoothing [24]. The sequence terminates at the largest scale where the smooth part of the function is simply a constant. This constant, plus the series of details at each scale j , comprise the wavelet transform of the function.

The DWT is easily applied to uniformly spaced discrete data. In this application, we use the simplest possible wavelet transform based on the Haar function:

$$\psi(t) = \begin{cases} -1 & \text{for } t \in [0, \frac{1}{2}) \\ 1 & \text{for } t \in [\frac{1}{2}, 1) \\ 0 & \text{otherwise} \end{cases}$$

The dataset is a time series $f_k \equiv f(t_k)$ with $t_k = kT_0$ for $k = 1, \dots, 2^M$. In the Haar basis the sums and differences underlying the DWT have a very simple form. In the first iteration, the smoothed signal f^1 is constructed by taking the simple average of adjacent values, while the detail signal d^1 is defined as the difference of adjacent values:

$$f_k^1 = 1/2(f_{2k} + f_{2k-1})$$

and

$$d_k^1 = (f_{2k} - f_{2k-1}).$$

The time resolution of the time series f^1 and d^1 is $T_1 = 2T_0$. The series d^1 gives the change in the value of f over the original time interval T_0 , recorded at intervals of $2T_0$ [8] [6] [21]. These smoothing and differencing operations are repeated on the signal f^1 , to create the two series f^2 and d^2 , which have a time resolution of $T_2 = 2T_1 = 2^2T_0$. The next iteration of the process uses

f^2 to produce a smoothed series and a series of details at time scale $T_3 = 2^3T_0$. With each iteration, the time interval between adjacent values in a series doubles, and the total number of elements in the time series decreases by a factor of two. The process terminates at scale $2^M T_0$, with f^M equal to the average of f . All the information about the original function is contained in the series of details d^m . These details are just wavelet coefficients of $f(t)$, with $m = M - j$, and the index k used to enumerate successive values in each series. Hence, the deltas defined by d_k at scale $2^m T_0 \equiv T_m$ are equal to the wavelet coefficients.

This approach differs from common regression where time series of step changes is defined as $d'_k = x_k - x_{k-1}$. As both d'_k and d'_{k+1} depend on x_k , some degree of serial correlation is artificially introduced into the time series d'_k . The same thing happens when a smooth signal is created using a running average. In contrast, the orthonormality of the DWT ensures that the deltas at different time scales contain independent information.

Discrete Wavelet Transforms can be implemented by considering Multiresolution Analysis of data. The input time series is fed through a digital High Pass Filter and Low Pass Filter that constitutes a filter bank that decomposes the signal into two equal high and low frequency components. The resulting components can be down sampled by a factor of two with no loss of information. For most signals, the information contained in the low frequency component obtained through the low pass filter, is characteristic of the signal, i.e. the time series under observation. [7] conducted some initial studies regarding Multi Resolution Analysis of wind power data series. It was found that different time scale resolutions conserve the autocorrelation function shape that validates the consistency of wavelet transformed data to be used as a substitute for raw data when forecasting power.

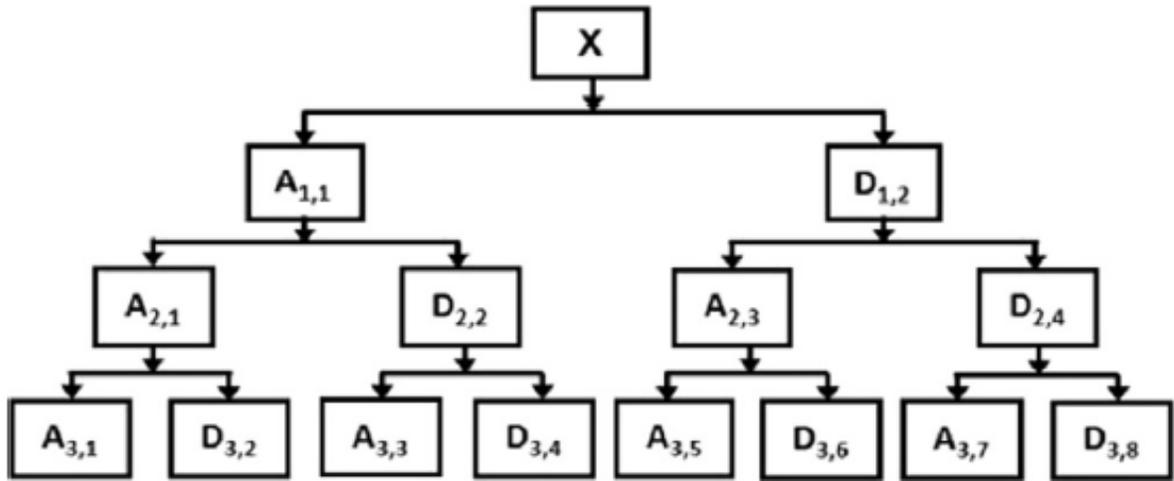


Figure 1: Wavelet Decomposition tree representation with elimination of Detail coefficients for analysis

To analyze the white noise content of the input time series, the detail coefficients of the wavelet decomposition are dropped systematically and the wavelet series is reconstructed only preserving Detail Coefficients of consecutive decomposition levels. As represented in Fig1, input data X is decomposed into approximation and Detail Coefficients A_i and D_i . At first decomposition level, X breaks down into A_1 and D_1 . For the current study, the detail coefficient D_1 is dropped off and the wavelet is reconstructed using the remaining detail coefficients, i.e. $D_2, D_3, D_4, D_5, D_6, D_7$. The reconstructed signal would not include the white noise lost from the D_1 coefficient. The process is repeated by iteratively dropping detail coefficients upto the equivalent decomposition levels, i.e. D_1, D_2 are dropped for decomposition level 2, D_1, D_2, D_3 are dropped for decomposition level 3 etc.

Forecasting electricity load with advanced wavelet neural networks AWNN, [17] achieved MAPEs of 0.268% for 5 minute forecasting horizon, 0.938%-1.716% for hour ahead forecasting considering high(5 minute) and low resolution(1 hour) data respectively. Forecasting errors increased as the forecasting horizon increased with 12 hour forecasting resulting in a 4.887% error. This serves as a reference to the current study to assess accuracies of AWNN methods.

Artificial Neural Networks

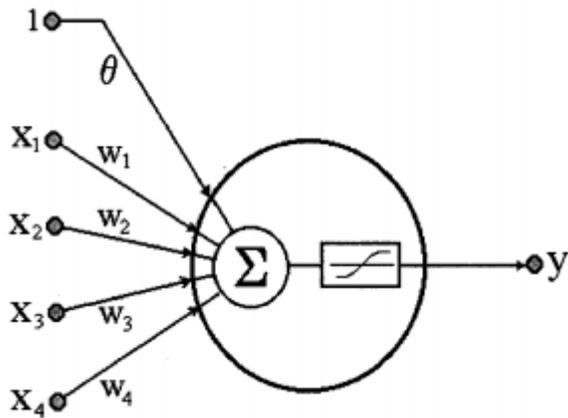


Figure 2: An Artificial Neuron

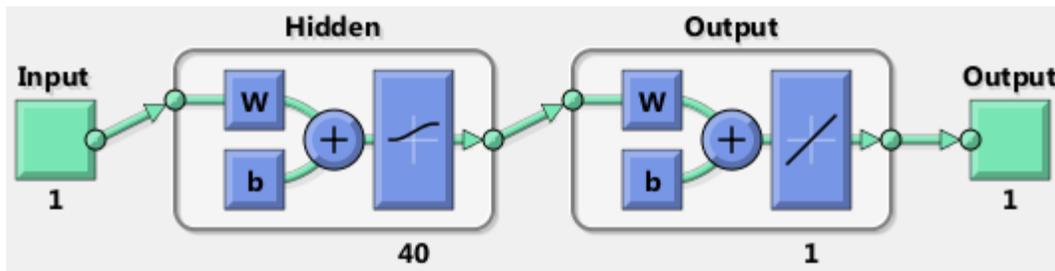


Figure 3: The modeled ANN, two layer feedforward neural network

Artificial Neural networks are designed to work like the neurons of the human brain. The artificial neuron, represented in Fig 2 makes up the building blocks of an ANN. The neural network receives inputs, i.e. the raw data series which is used to forecast values. An ANN can accept multiple inputs (X_1, X_2, X_3, X_4) to generate predictions Y . For the current study, a single input and a single output are applicable since no descriptive variable has been used for the historical time series data. The model is a two layered feedforward network based model. The first layer or the first stage linearly combines the inputs and passes it to a non-linear activation function. Weights (w_1, w_2, w_3, w_4) are assigned to each connection with a bias term θ represented in the Fig 2 as a fixed input equal to 1. The chosen activation function should be a non-decreasing one and the log sigmoid function was chosen for this study. The logistic sigmoid function can be expressed as, ($y = 1/(1 + e^{(-x)})$). A single input feedforward architecture network is built for the current study 40 modeled neurons in the two layers between the input and the output layers (hidden layers.)

The estimation of parameter values is governed by a training function that minimizes a loss function. The Levenberg-Marquardt backpropagation function has been used for the ANN model. The LM function is the fastest among all other training functions and is thus a suitable work horse for analyzing large datasets such as the ones in use for this study. [25] The last stage of creating the ANN model, the validation stage, calculates the performance statistics of the model. In sample errors are used to calibrate the forecasting model during this stage while out of sample errors are used for final testing of the forecasting model. The number of neurons in the hidden layers can be adjusted to get the best forecasting performance or the number of hidden layers can be increased at the cost of computing power for increased performance. The method of selecting parameters is largely an iterative hit and trial.

[26] analyzed daily forecasting accuracies but being one of the very early study, obtained run times of 3 to 7 hours with considerable error, though established the basis of applying Neural Network Techniques for Load forecasting. [27] summarizes previous literature on load forecasting using ANNs. While most literature uses ambient temperatures as a descriptive variable for load forecasting, [28] used only historic load data series for future forecasting.

Autoregressive Integrated Moving Average model

A time series approach to load and wind power forecasting stems from the idea that there are several patterns to the generation of such electricity sources. The variability can be analyzed by studying the seasonality of data since the patterns emerge on different time scales, daily, weekly, monthly. The ARMA model can characterize the stochastic behavior of short term load and wind power patterns. Wind power has conventionally been forecasted and calculated indirectly from wind speed predictions. While literature regarding algorithmic prediction of direct wind power is sparse, sufficient literature is present on wind speed prediction and modeling. [29] employed a stochastic modeling approach to wind power generation using a discrete Markov model based on transition matrix while ARMA based wind speed models were employed by [30] and [31]. ARIMA models work differently for stationary and non-stationary time series processes. To understand the concept of stationarity, the Auto Correlation Coefficients and the Partial Auto Correlation Coefficients of the time series need to be studied. Let $y(t), \dots, y(N)$ denote a time series modeled by a random process $Y(t)$. The Auto Correlation and the Partial Auto Correlation Coefficients (ACC, PACC) then describe the temporal correlation of the time dependent process. [32] The

linear correlation of the adjacent data in the random process is expressed by the ACC ($\hat{\rho}$) that can be estimated as [33]

$$\hat{\rho}(k) = \frac{1}{N-k} \frac{\sum_{i=1}^{N-k} [y(i)y(i+k)] - \hat{m}_Y^2}{\hat{\sigma}_Y^2}, \text{ for } k = 0, 1, \dots, N-1$$

where \hat{m}_Y and $\hat{\sigma}_Y^2$ are the sample mean and variance of the time series respectively.

The Partial Auto Correlation Coefficients describes the correlation between $Y(t)$ and $Y(y+k)$ with no linear dependence of $Y(t+1), Y(t+2), \dots, Y(t+k-1)$. With the ACC of the sample known, the PACC $\hat{\gamma}_Y(k)$ can be calculated iteratively.[33]

$$\hat{\gamma}(k+1) = \frac{\hat{\rho}_Y(k+1) - \sum_{j=1}^k \hat{\gamma}_Y(k,j)\hat{\rho}_Y(k+1-j)}{1 - \sum_{j=1}^k \hat{\gamma}_Y(k,j)\hat{\rho}_Y(j)}$$

Where $\hat{\gamma}_Y(0) = 1$, and

$$\hat{\gamma}_Y(k,j) = \hat{\gamma}_Y(k-1,j) - \hat{\gamma}_Y(k)\hat{\gamma}_Y(k-1,k-j).$$

The critical limits for hypothesis testing of the white noise are defined as $\pm 2S_{PACC}$ where S_{PACC} is the standard error of the PACC of the sample given by, [33]

$$S_{PACC} = \sqrt{1/N}$$

Wind power and demand loads are non-stationary processes $Y(t)$ and they may have time dependent mean $m_Y(t)$ and variance $\sigma_Y^2(t)$. Thus, such processes require a differencing transformation to convert the nonstationary random process to a stationary process. For the time varying mean, the stochastic trend model is applied [33]. The model is applied by differencing the random process d times such that,

$$Z(t) = (1-B)^d Y(t)$$

where B is the backshift operator, $B^j Y(t) = Y(t-j)$.

The $ARIMA(p, d, q)$ model for a non-stationary time process $Y(t)$ can thus be defined as,

$$\left(1 - \sum_{i=1}^p \varphi_i B^i\right) (1-B)^d Y(t) = \theta_0 + \left(1 - \sum_{i=1}^q \theta_i B^i\right) a(t)$$

where $\{\varphi_i\}$ are the AR coefficients; $\{\theta_i\}$ are the moving average (MA) coefficients; $a(t)$ is a Gaussian process with zero mean and variance σ_a^2 ; the parameter θ_0 is the deterministic trend term when $d > 0$. In the case of $d = 0$, i.e. ARMA model, the θ_0 term can be related to the mean \hat{m}_Y of the process by

$$\theta_0 = \hat{m}_Y(1 - \varphi_1 - \dots - \varphi_p).$$

A seasonal ARIMA model is defined as $ARIMA(p, d, q) X (P, D, Q)S$,

where p = non-seasonal AR order, d = non-seasonal differencing, q = non-seasonal MA order, P = seasonal AR order, D = seasonal differencing, Q = seasonal MA order, and S = time span of repeating seasonal pattern.

With the seasonality of data included, the ARIMA model would to behave differently. With a defined seasonality (S) of data, the autoregressive model would predict x_t term of a time series based on the x_{t-S} term in case of the first order seasonal model. For the second order seasonal model, the x_t term would be predicted based on x_{t-S} and the x_{t-2S} terms. The seasonal first order Moving Average would use θ_{t-S} as the predictor and the second order model would use θ_{t-S} and θ_{t-2S} as the predictors. The seasonal differencing term D is defined as the difference between a value and a value with the lag that is a multiple of S . [34] Thus, for a seasonality S , the seasonal difference is $(1 - B^S)x_t = x_t - x_{t-S}$. The seasonal differencing removes the seasonal random walk non-stationarity while the non seasonal differencing removes the trends presents in the data. Using a d order non-seasonal differencing term gives $(1 - B^d)x_t = x_t - x_{t-d}$ when a trend is present. When both trend and seasonality is present in a data series, predictions are made: $(1 - B^S)(1 - B^d)x_t = (x_t - x_{t-d}) - (x_{t-S} - x_{t-(S+d)})$

The problems with employing ARIMA models for power prediction arises from the prediction inaccuracies and numerical instability that can throw off the model. Using 5 minute resolution data increases the instability by a great factor and also greatly increases the computational requirements for predictions.

Study area description

The Bonneville Power Administration is the federal agency responsible for marketing the output of 31 federal hydroelectric dams and a single nonfederal nuclear power plant [35] located in the Columbia River Basin. Bonneville has been the largest supplier of electricity in the Pacific Northwest; Oregon, Washington, Idaho, Oregon and Montana; and controls approximately 75 percent of the transmission capacity in the region. [35]

BPA started integrating wind power gradually from 1990s through contracting with power plants. BPA now has almost 5100 MW of wind power integrated in the transmission systems and is headed for expanding its renewable portfolio by additional 3000-4000 MW of wind energy by 2025 thus contributing to renewable energy targets of the Northwest states.

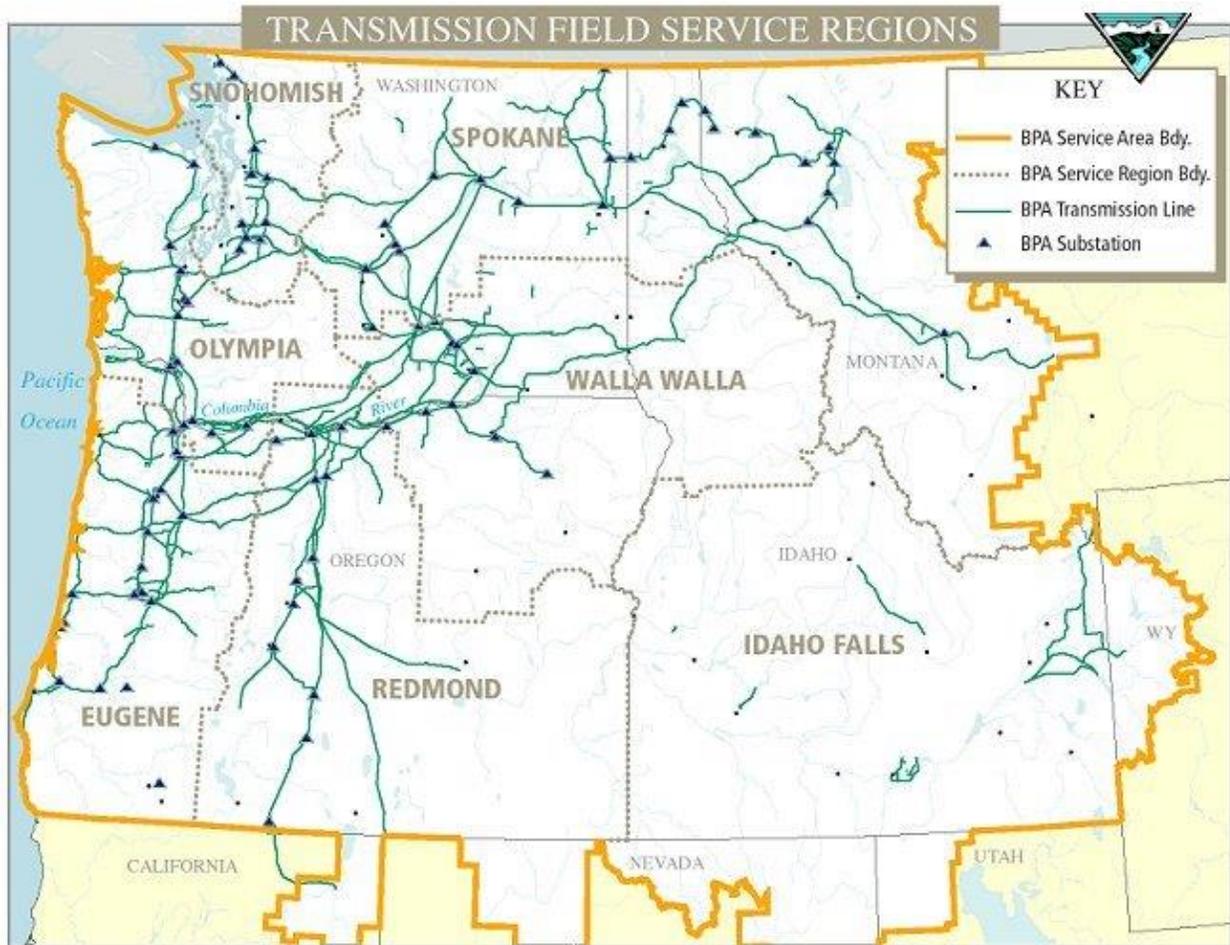


Figure 4: BPA service area

The increasing renewable penetration and diverse energy portfolio has brought about important transmission stability concerns and posed for a greater need of understanding intermittency of generating sources. Wind energy is negatively correlated to the load demands for most parts of power production. This implies wind power generation when it is not needed. TO accommodate this power input into the grid system, other sources, predominantly hydro, need to be ramped down. Ramping sources have physical limitations that can adversely affect the network stability. James Conca, [36] writer with Forbes highlights the negative reasons of wind power inclusion in the system and the economic incentives that drive wind integration in BPA regions.

5 minute resolution datasets of wind power generation and demand load were download from data servers of BPA for the years 2007 to 2014 for the current study.

Methodology

Preconditioning of data

All of the historic data series, wind and load data for 2007-2014, were loaded into a single project file named “main project.mat”. Several non-available data points were found within the data series. The script “Nan_correction.m” corrects the non-availability of such data points. A base year was declared to be used for all analysis and for training Neural Networks, 2007 was taken as the base year. The load and wind data series was converted into a 2D matrix with rows representing the days of the year and columns signifying the time of the day through a loop. A day input is requested from the user to show outputs of that particular day during plotting. Next, the zero padding of the wavelet is conducted. Daily profiles of load and wind from the arranged two dimensional data is decomposed at a user defined decomposition level and Daubechies 4 wavelet. A decomposition vector and a book keeping vector are obtained through the decomposition function. The decomposition vector is essentially a concatenation of the approximation and detail coefficients (Fig 1) obtained through decomposing the original data and the book keeping vector aids in reconstruction of data by providing appropriate referencing upon computation. The detail coefficients are then extracted from the Decomposition vector at each decomposition level equal to and less than the one that was accepted from the user, equated to zero and a new decomposition vector was created by concatenating the approximation coefficient at the user specified decomposition level and the zero padded detail coefficients. A two dimensional matrix for the

wavelet data output is created of the same form as the two dimensional input data where rows refer to the days of the year and columns represent time indices by using the wavelet recreation function. This recreated matrix is the wavelet recreated dataset that is then compared to outputs of smoothing and fitting algorithms and fed in to the artificial neural network to compare performance when using wavelet recreated data and the results from using raw data as input to the neural network.

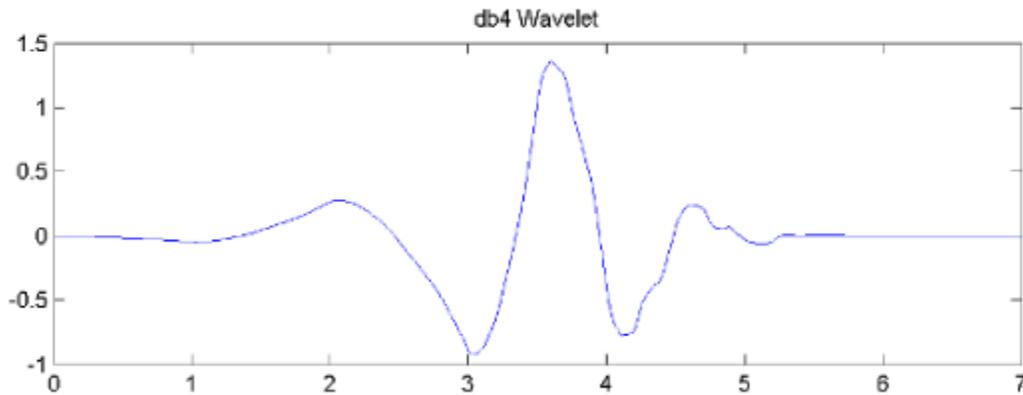


Figure 5: Daubechies 4 wavelet: Daubechies' external phase wavelet with 4 vanishing moments.

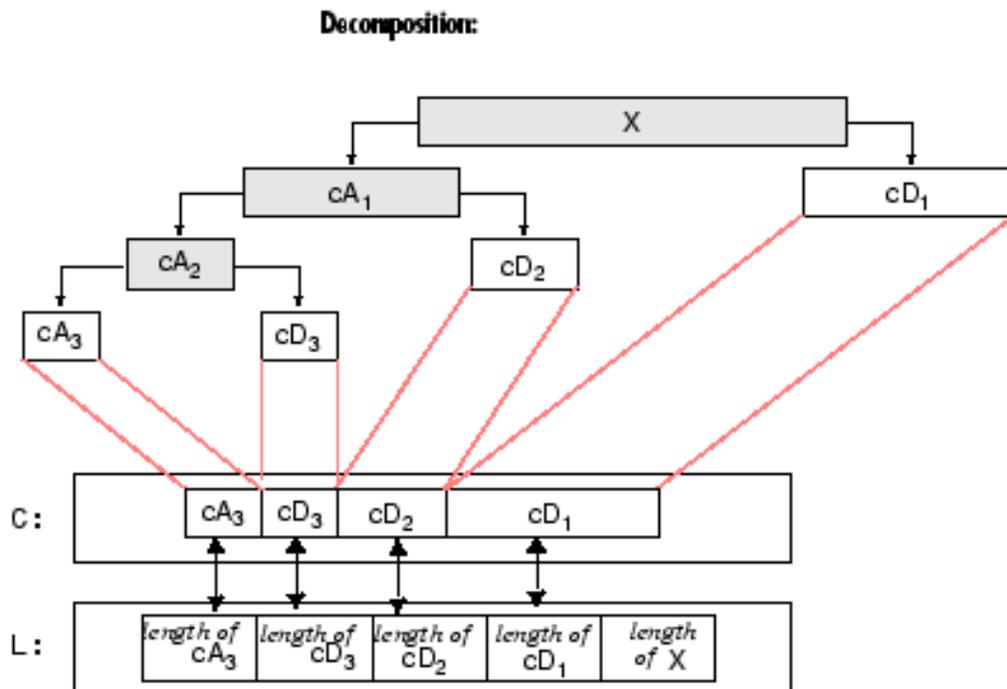


Figure 6: Decomposition of a time series X with C representing decomposition vector and L representing book keeping vector

Objectives

To evaluate performance of Discrete Wavelet Transforms against other Smoothing and Polynomial Fitting Algorithms through statistical metrics

Smoothing

Smoothing algorithms create an approximation function that captures characteristic pattern in the dataset while filtering out the noise or other rapid phenomena. Data smoothing works by modifying data points in the time series to reduce the individual points while increasing the points that are lower than the adjacent points. Smoothing performance of wavelet technique was analyzed in comparison to four other smoothing algorithms: statistical moving averaging of data points, Savitsky-Golay filter convolution smoothing, and LOESS (Local regrESSion) and LOWESS (Locally Weighted Scatterplot Smoothing) that work on linear and nonlinear least square regression methods. A smooth curve through the data points is called the Loess Curve while if the smoothed values can be defined by a weighted linear least square regression, it is known as the Lowess curve. While Moving Averaging has been the norm for data smoothing operations considering wind power and load data, it was found that smoothing performance is the worst for moving averages among all the algorithms and the wavelet approach. The wavelet approach produced smoother data compared to Moving Averages but cannot perform other well defined smoothing techniques.

Curve Fitting

A polynomial regression fitting is analyzed on the data sets. Polynomial regression work on the basis of linear regression where the relationship between a dependent, y , and an independent, x , variable is modeled as a n^{th} degree polynomial. Polynomial regression fitting is conducted based on the method of least squares minimization.

Curve fitting and smoothing are distinguishable from one another as curve smoothing does not involve the use of an explicit function form for producing the result and only produces smoothed data values. The extent of smoothing can also be controlled by defining a tuning parameter while curve fitting produces a best fit estimation.

Standard Deviation of Error

The standard deviations of errors between the smoothed/fitted output and the raw data input is calculated. Lower standard deviations throughout the time scale of the time series is a positive indicator that the adjusted data points are not too spread apart from the original data.

$$Err_i = y_i - y_0$$
$$SDE = \sqrt{\frac{1}{N} \sum_{i=1}^N (Err_i - \mu_{err})^2}$$

Where, Err_i = Error of estimation, y_i = Smoothed/Fitted values, y_0 = Original raw data, N = Number of elements, SDE = Standard Deviation of Errors, μ_{err} = Mean of errors

Root Mean Square Error

The root mean square error will be used as a metric to report errors of smoothing, fitting and wavelet algorithms from the raw data. RMSE is a very common metric used for error reporting of numerical predictions. RMSE is a good indicator of large errors that are amplified and punished more severely than in Mean Absolute Error reporting.

$$RMSE = \sqrt{\frac{1}{n} (\sum_{i=1}^n (y_i - y_0)^2)}$$

Where, n = Number of observations, y_i = Smoothed/Fitted values and y_0 = Original raw data

Gradient

The statistical behavior of load and wind power wavelet output time series will be characterized based on standard deviation error from the raw data and gradients to characterize the smoothness of the wavelet output time series and outputs after applying smoothing and fitting algorithms. A highly volatile and abruptly changing gradient profile is indicative of an unsmooth curve. While lower variations in the gradient is a desirable property, it should be noted that the gradient should match closely to the gradient profile of the original data for recreation. The mean of daily gradients for all historic years is plotted for analysis. The mean of daily gradients for all historic years is plotted for analysis.

$$\nabla F = \frac{\delta F}{\delta x} \hat{i} + \frac{\delta F}{\delta y} \hat{j}$$

Where ∇F = Gradient of function F and $F = F(x, y)$.

To analyze the optimal decomposition level that can be achieved with lower errors, i.e. optimal white noise removal level

The recreated wavelet signals will have varying degrees of information loss according to the respective decomposition level. This procedure is conducted by removing the detail coefficients till the specified decomposition level during the reconstruction of the wavelet signal, as described previously. It is anticipated that errors will increase as more detail coefficients are lost during signal reconstruction owing to the fact that loss of white noise would actually imply loss of characteristic signal components. Losses at different decomposition levels will represent loss of noise at different time scales according to the Wavelet theory. Table shows the time scale of noise loss that corresponds to each decomposition level of the original time signal. Since different generating sources need some ramping up time to fulfil demand gap, the load profiles can be modified in a manner that removes any extra noise from the original time series data and only preserves pertinent data that is accurate for the time scale under observation. For example, diesel generators are fast ramping power sources and would come online within 5 minutes or 10 minutes. At decomposition level 1, the 5 minute time resolution data will be modified such that information at every alternative step will be preserved, that is, information at every time index that is a multiple of 10 will be present. Similarly, at decomposition level 6, information at every 320 minute time step will be preserved. 320 minute \equiv 5.3 hours is a typical time required for wind energy generator to make arrangements for power production scheduling. Hence wavelet decomposition can assist in providing a leaner, noise-free power curve for reference to power generators for better power planning and management.

Decomposition Level	Time resolution	Type of ramp
0	5 min	Fast
1	10 min	Fast
2	20 min	Fast
3	40 min	Medium
4	80 min	Medium

5	160 min	Slow
6	320 min	Slow
7	640 min	Very Slow
8	1280 min	Very Slow

Time Resolution Manipulation

It was established that wavelet transformation averages the original time series through the approximation coefficient. This implies that if the raw time series has a time resolution of T_0 , the approximation coefficient obtained at decomposition level 1 will have a time resolution of $T_0/2$. Subsequent decomposition levels (n) would yield time resolution of T_0/n . This can be exploited to alter time series of the power datasets released by utility companies to achieve uniformity of data releases. The original raw data series was decomposed using the Wavelet Transform at a specified decomposition level (n) and the approximation and detail coefficients were preserved for use. Now, the wavelet reconstruction process is constructed with altered approximation and detail coefficients. Every alternate time stamp value of the original time series was assumed to act as an approximate coefficient and the detail coefficient used was a zero element matrix, just to fill in the missing requirement of the detail coefficient while effectively removing the detail coefficients from the newly reconstructed wavelet. The decomposition vector was recreated by concatenating the approximation and detail coefficients. The wavelet signal was reconstructed using this decomposition vector and the original book keeping vector. The manually set approximation coefficient had a time resolution of $T_0/2$. After reconstruction of the wavelet signal, a time resolution of T_0 is obtained, equivalent to that of the original signal albeit without the contribution of the detail coefficients. The approximation coefficient with alternate time stamp values of data was also interpolated to produce a series with double the number of time steps to be compared besides the wavelet transformed “upscaled” time series.

Forecasting of time series

Artificial Neural Network (ANN)

The forecasting Artificial Neural Network was created to accept the raw input data series and predict values based on historic data at different time scales. A base year was declared for creating the neural network and initialize its weights and parameters. The method for training and initializing the neural network differs

according to the time scale under observation though the construction of the neural network stays uniform throughout. The created network uses the “divideint” division function that divides the target data into training, validation and testing sets using interleaved sections according to the specified division ratios. The target data was divided equally for this step, so 1/3 ratio was used for each set. Next, the maximum fail parameter was set at 6, this represents the maximum allowable validation failures during the training step of the network. Increasing the number of allowable failure at validation to a higher number did not yield better results and resulted in much higher computation times. Next, the transfer function between the first and second layer of the network was specified as the log sigmoid function. It can be set to the tan sigmoid too but logsig function is used because it generates outputs between 0 and 1 as the neuron’s net input may be negative or positive. The tansig function generates outputs in the -1 to 1 range. The purelin function also exists that generates outputs in the range $[-\infty \text{ to } \infty]$. Since the load and wind data need to be strictly positive values, logsig function is the only function that should be used. The network is then trained based on an input and a target data. The input and target data are normalized based on their maximum values such that, $Normalized\ input = Input / Max(Input)$. Data normalization is required to substantially cut down training and prediction times. It should be noted Neural Network inputs and target data should be row vectors for correct predictions, column inputs produce a single output instead of a range of outputs. Errors were calculated as the Mean Absolute Percentage Errors. MAPEs for a time scale are defined as $MAPE = mean(\frac{|Prediction - Actual\ value|}{Actual\ value})$ at every time step within that time scale. The cumulative time scale MAPE is then defined as the average value of all the MAPEs at that time scale. So, $MAPE_{yearly} = (MAPE_1 + MAPE_2 + MAPE_3 + MAPE_4) / 4$ where 1, 2, 3, 4 represent the 07-08, 09-10, 11-12, 13-14 year pairs. $MAPE_{monthly} = (MAPE_{Jan-Feb} + MAPE_{Feb-March} + MAPE_{March-April} + MAPE_{April-May} + MAPE_{May-June} + MAPE_{June-July} + MAPE_{July-Aug} + MAPE_{Aug-Sept} + MAPE_{Sept-Oct} + MAPE_{Oct-Nov} + MAPE_{Nov-Dec}) / 12$. Similarly six month time scales, day ahead time scales and hour ahead time scales in addition to month ahead and year ahead time scales are analyzed by the same pair-wise approach. For wind power forecasting, only hour ahead and day ahead forecasting was conducted as any time scale greater than those would greatly increase the variability and the number of time indices when wind speed essentially remains zero hence throwing off the forecasting model. ANNs are only effective for short term forecasting of wind power. MAPEs are problematic when considering wind power forecasting because the wind power is zero very often which returns MAPEs of Infinity or NaN quite often, effects that can completely throw off the reported mean MAPE. Thus, MAPEs that were greater than 1 (100% error), Infinity and Non Available MAPEs were removed prior to calculating the mean MAPE for overall error calculation. The range of MAPEs and the average MAPEs were reported for each time scale.

Wavelet fed Artificial Neural Network (WANN)

The Wavelet fed Artificial Neural Network model works the same way as the Artificial Neural Network only utilizing wavelet recreated data as prediction inputs. The wavelet algorithm was applied on the raw data which was still one dimensional and continuous for the whole year. The decomposition level to be used for wavelet decomposition and recreation was specified iteratively using a loop to analyze the decomposition level that best suits the forecasting needs. MAPEs were used as the metric for classifying performance of the models. MAPE reporting was conducted similar to ANN though MAPEs were reported for every decomposition level of the wavelet transform.

Auto Regressive Integrated Moving Average (ARIMA) model

While identifying the ARIMA model to be used, the time series plot of the data, the Auto Correlation Function plot and the Partial Auto Correlation Function plot are the features to be observed. The time series plot of the wind power and demand loads gives a general idea about the actions that need to be taken for successful estimation of the ARIMA parameters. First off, obvious upward and downward trends were noted. Linear trends need to be removed using the first order differentiation while quadratic terms require a second order differencing. The 5 minute time resolution load data has a stark quadratic trend owing to the seasonality of load usage. Power demand is higher in the winter months, January-February and then in November-December and lower demand in the summer months. Thus the load curve is expected to be a quadratic shaped function (Fig: 6). Quadratic shaped profiles are differentiated to the second to remove trends. Thus a differencing term of 2 is used on the load profiles being used. Wind power generation profiles on the other hand are not quadratic in nature (Fig 7) and the maximum power remains almost the same throughout the year. Thus a first order differencing is conducted in the case of wind power profiles to account for the slightly higher wind speed months of November and December,

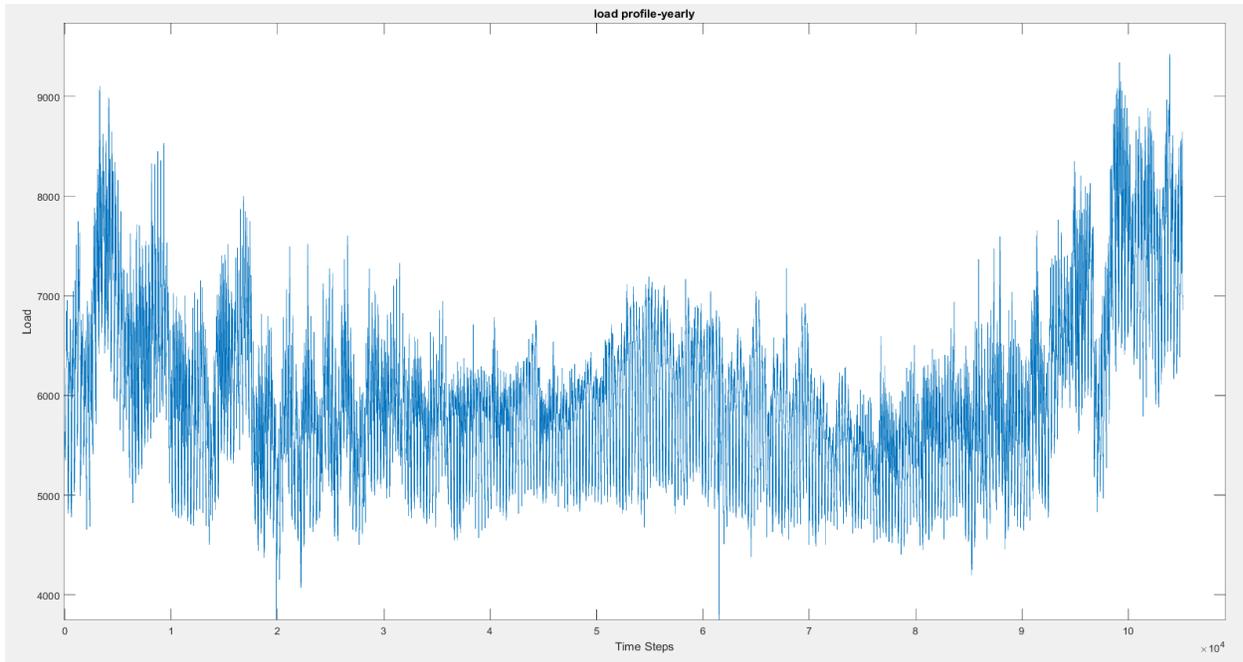


Figure 6: Yearly load demand profile for 2008

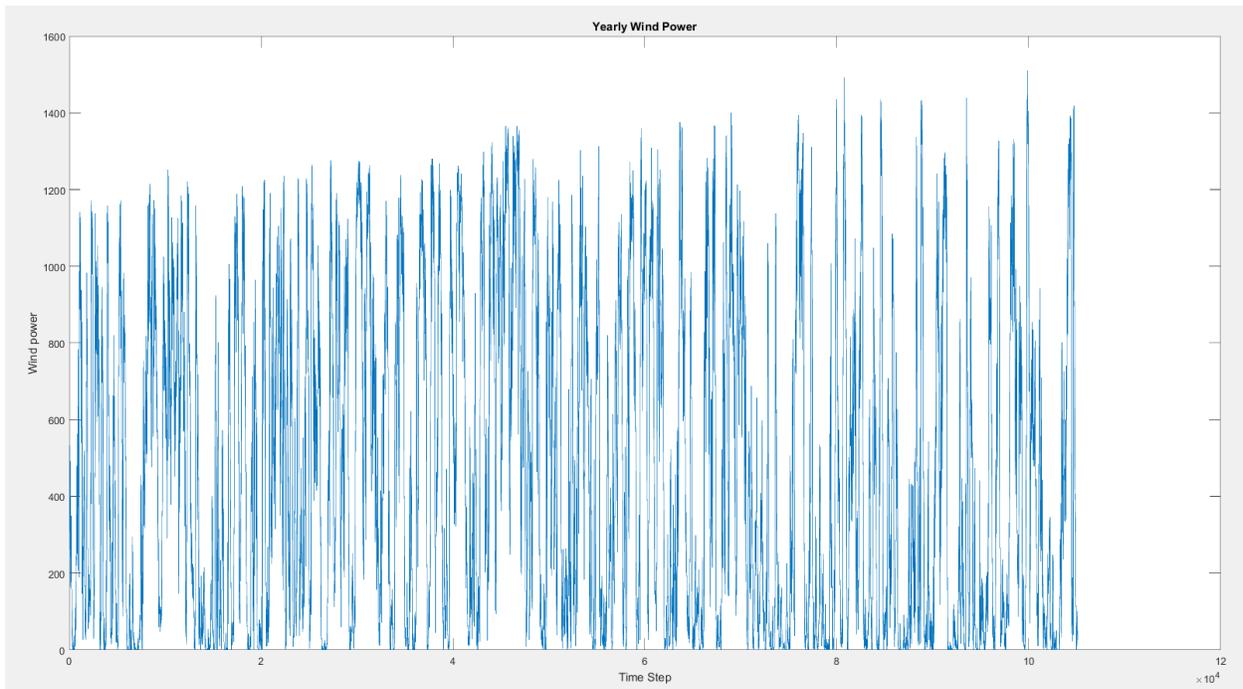


Figure 7: Yearly wind power generation profile of 2008

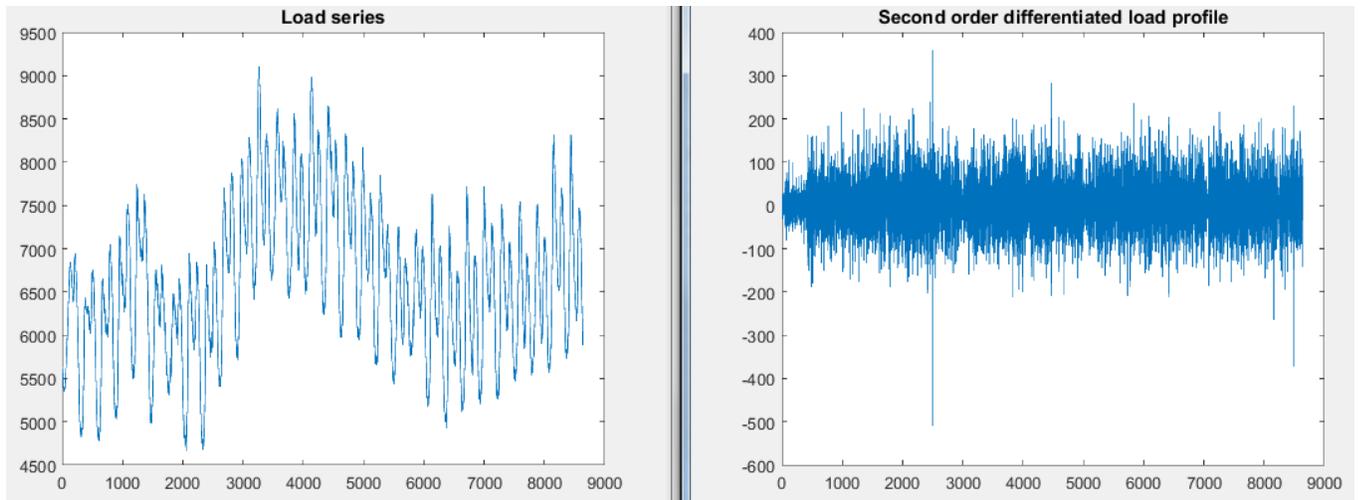


Figure 8: Effect of first order differentiation on the wind power series (daily profile)

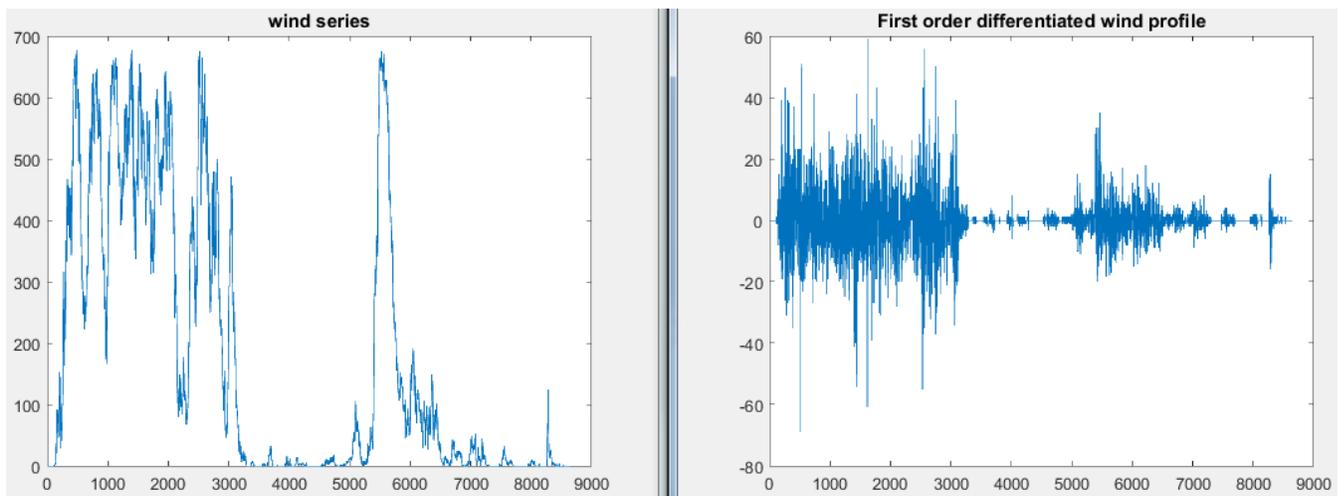


Figure 9: Effect of first order differentiation on the wind power series (daily profile)

The AR and MA terms are inferred from the Auto Correlation Function (ACF) and Partial Autocorrelation Function (PACF) plots according to the following rules:

- a) The lag beyond which the ACF cuts off within the confidence bounds indicates the number of MA terms to be used.
- b) The lag beyond which the PACF cuts off within the confidence bounds indicates the number of AR terms to be used.

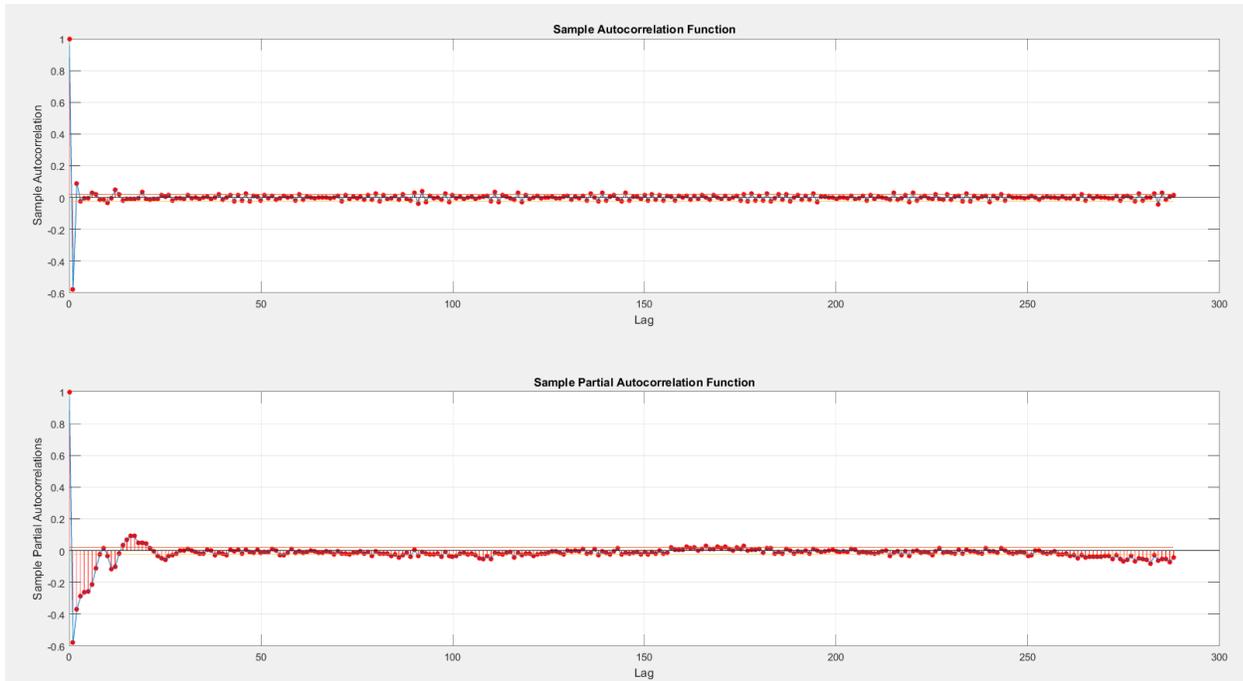


Figure 10: ACF and PACF of second order differenced load time series

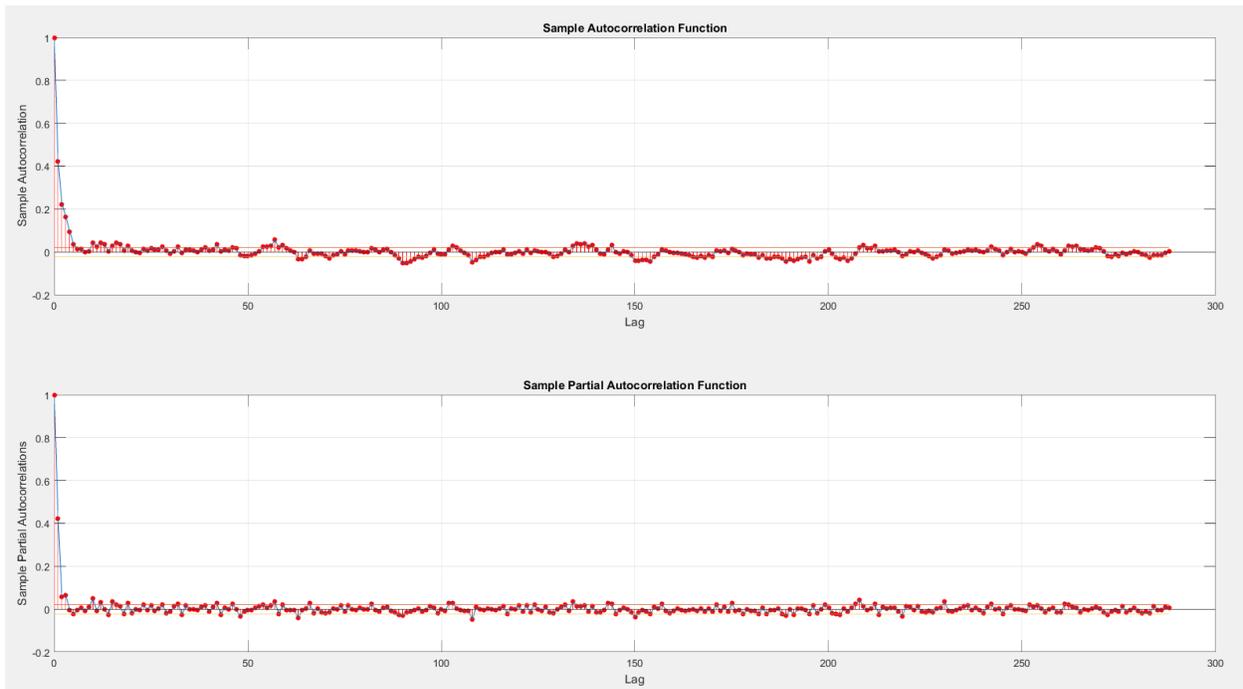


Figure 11: ACF and PACF of first order differenced wind power time series

Time lags of 288 units \equiv 1 day are chosen because the patterns repeat on a daily basis.

Observing the ACF and the PACF plots, for the load time series,

- a) Since the PACF at the first lag is negative, an MA term is required for the series. The lag beyond which the ACF cuts off indicates the number of MA terms. i.e. MA = 2. MA term rectifies over differencing of time series
- b) An AR term is chosen if the PACF of the differenced time series cuts off sharply or the lag 1 value is positive indicating under differencing. Since the third order differentiated time series PACF cuts off after lag 8 i.e. AR = 8
- c) The series has a seasonal pattern that represents the electricity usage over a day repeated throughout the month. A day corresponds to 288 units of time steps in the case of load profiles. Therefore, the auto correlation at lag 288 is observed to determine the seasonal terms of the ARIMA model. The 288th term is positive, hence a SAR term is used in the model.
- d) The AR term of 8 does not make sense to be used, hence an AR term of 1 used to correct for minor over differencing. Thus an ARIMA(1,2,2) is expected to perform the best for the load profiles. An ARIMA(1,2,1) model is also used alongside for performance comparisons.

Observing the ACF and the PACF plots, for the wind power time series,

- a) Since the PACF at the first lag is negative, an MA term is required for the series. The lag beyond which the ACF cuts off indicates the number of MA terms. i.e. MA = 5. MA term rectifies over differencing of the time series
- b) An AR term is chosen if the PACF of the differenced time series cuts off sharply or the lag 1 value is positive indicating under differencing. Since the third order differentiated time series PACF cuts off after lag 8 i.e. AR = 8
- c) The series has a seasonal pattern that represents the electricity usage over a day repeated throughout the month. A day corresponds to 288 units of time steps in the case of load profiles. Therefore, the auto correlation at lag 288 is observed to determine the seasonal terms of the ARIMA model. The 288th term is positive, hence a SAR term is used in the model.
- d) The AR and MA terms of 8 and 5 does not make sense to be used, hence an AR term of 1 and MA term of 1 is used to correct for minor over/under differencing. Thus an

ARIMA(1,1,1) is expected to perform the best for the load profiles. An ARIMA(1,1,0) model is also used alongside for performance comparisons.

Once the ARIMA models have been established, the estimate function initialized them according to the input data (using 2007 as the base data series). The infer function then plots standardized residuals for the ARIMA models. Next, the actual forecasting is carried out for the ARIMA models using the forecast function. The forecasting is a day ahead time frame that is based on a pre sample response input. The presample response input is set to use the previous 3 days to forecast the next (i.e. fourth) day. The forecasting is conducted iteratively for 8 hours to get a distributed error assessment.

The MAPEs of each day are averaged for both the models and the range of MAPEs (of daily values) and the overall mean MAPE is reported.

Results

To evaluate performance of Discreet Wavelet Transforms against other Smoothing and Polynomial Fitting Algorithms through statistical metrics

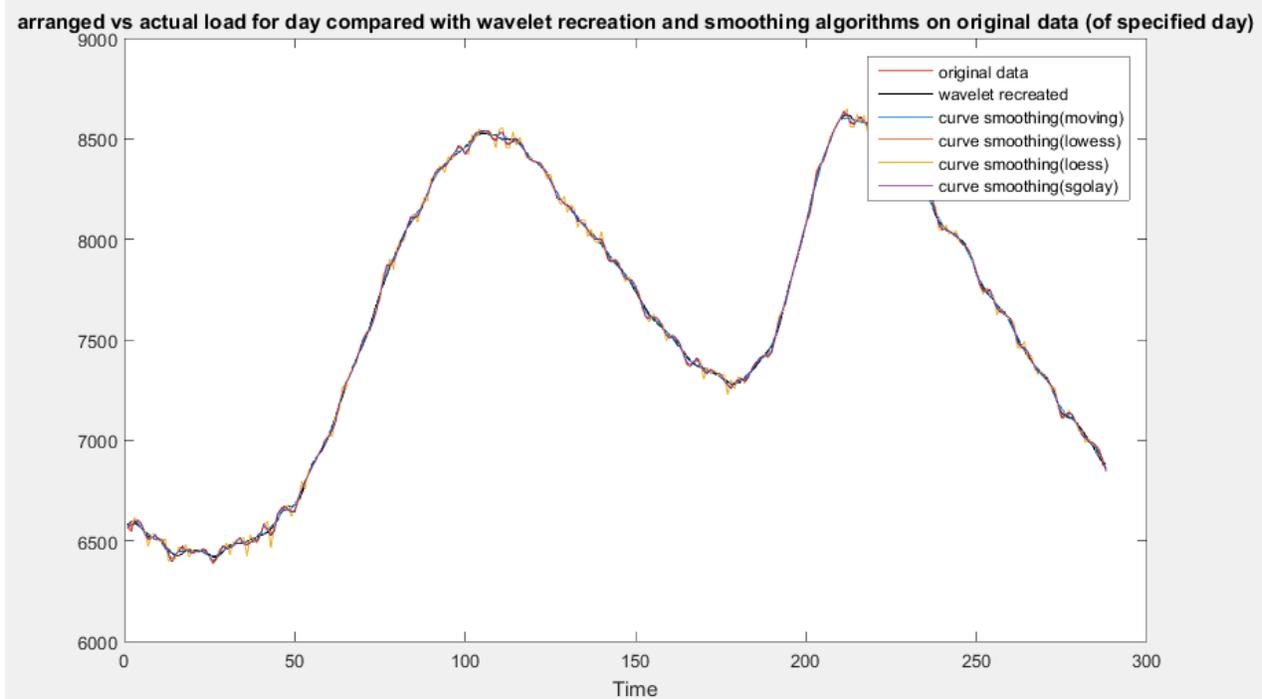


Figure 12: Smoothing and wavelet algorithms on load

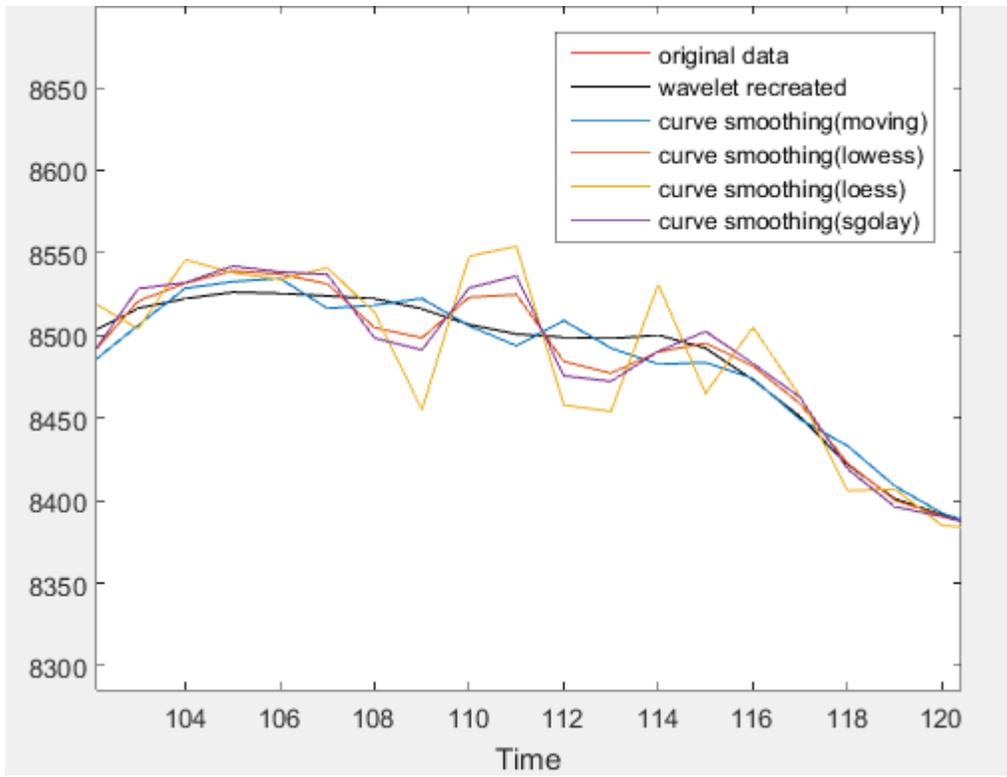


Figure 13: Zoomed section of smoothing and wavelet algorithms on load

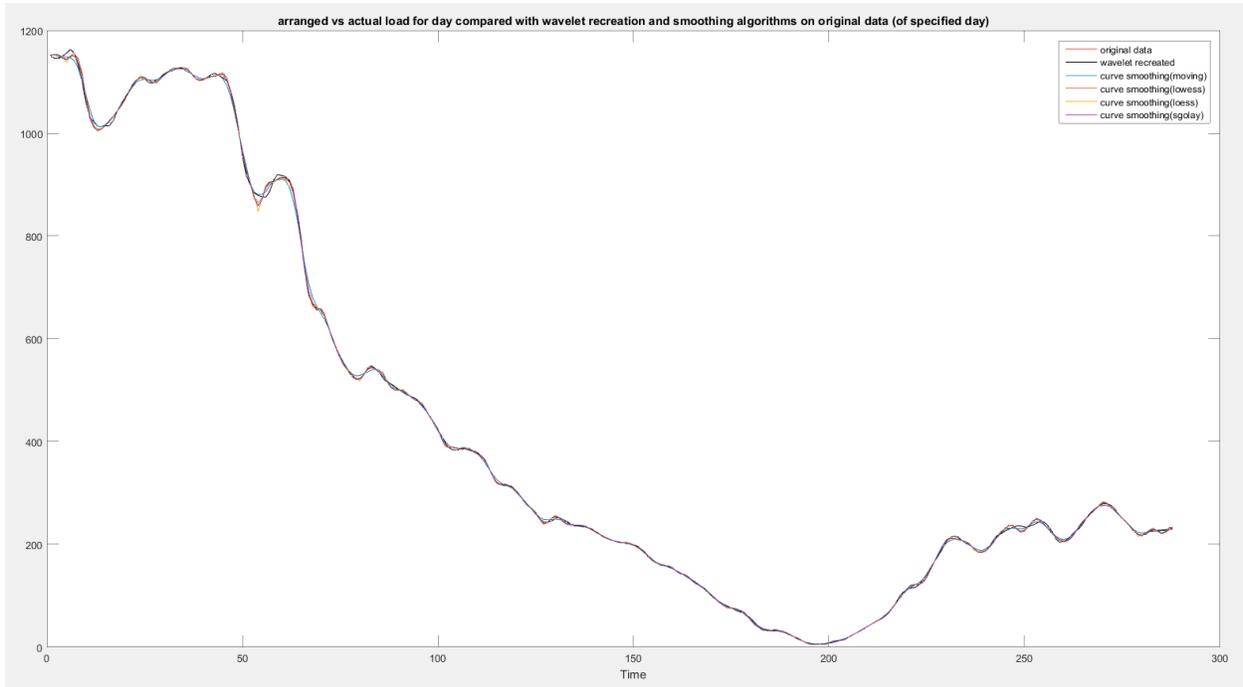


Figure 14: Smoothing and Wavelet algorithms on wind power

The wavelet recreated time series tends to be less “noisy” than other smoothing algorithmic outputs. The black wavelet curve serves as a base signal devoid of the white noise component and hence tends to be smoother than all other smoothing algorithms, owing to the loss of information in the recreation step. Wavelet recreated curve has a lower deviation from the main data curve as compared to the moving averaged series curve. The moving average method of smoothing is the standard method of smoothing temporal data and flatten out artifacts within the series. Wavelet algorithm hence proves to be a better smoothing technique than moving average smoothing. While other smoothing filters perform better than wavelet transforms and moving averaging, these have not gained wide acceptance in data analysis techniques owing to complexity of coding the filters and regressions.

Better performance of wavelet smoothing is observed on wind power time series data too. The effect is discernable at times when data variability is large. At time index 50 in Fig: 14 , wavelet recreated curve tends to capture the rapid change in wind values better than the moving average curve (wavelet curve is closer to the original data curve than moving average curve).

The benefit of using wavelet transforms for smoothing volatile wind power data lies in the fact that other than the better smoothing performance, the difference of the original data curve and wavelet recreated curves tend to contain characteristic details of the data that can be analyzed separately. The moving average does not have the advantage of explaining the deviation from the original data.

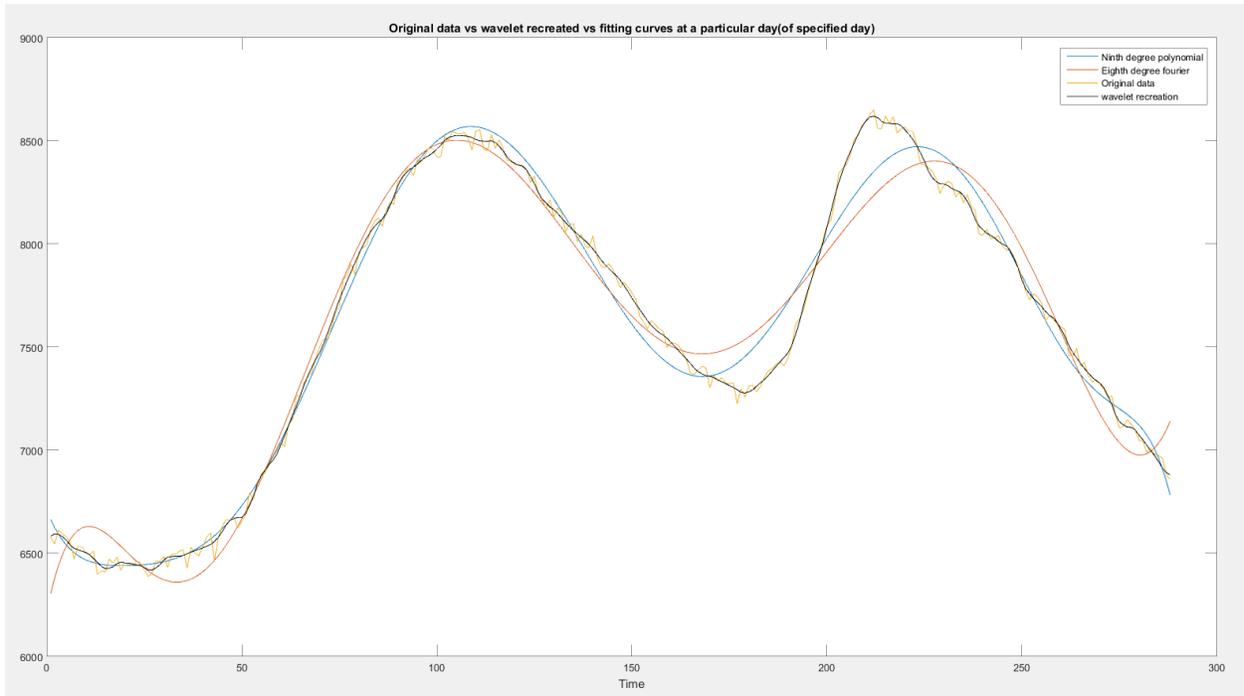


Figure 15: Fitting and wavelet algorithms on load

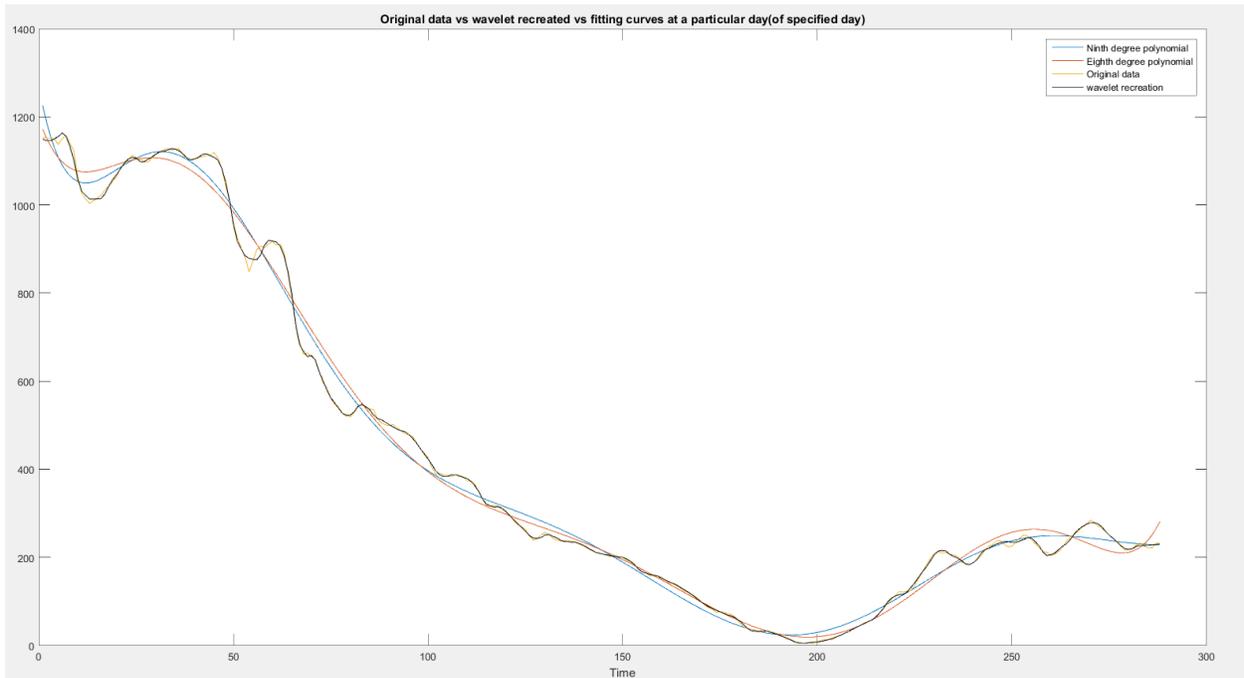


Figure 16: Fitting and wavelet algorithms on wind power

Owing to the volatility of 5 minute resolution data, fitting a polynomial to the original data does not produce good results with weak R^2 values as can be observed from the deviation of the polynomial curves from the original data curve. Wavelet recreation algorithm thus produces much

better fitting results by being much more correlated to the original data. This also highlights the effect of large volatility of high resolution data on creating a fitting curve that can define energy data. With no curves that can define volatile wind power and demand load data to a considerable extent, wavelet algorithm can be directly used to assess best fits.

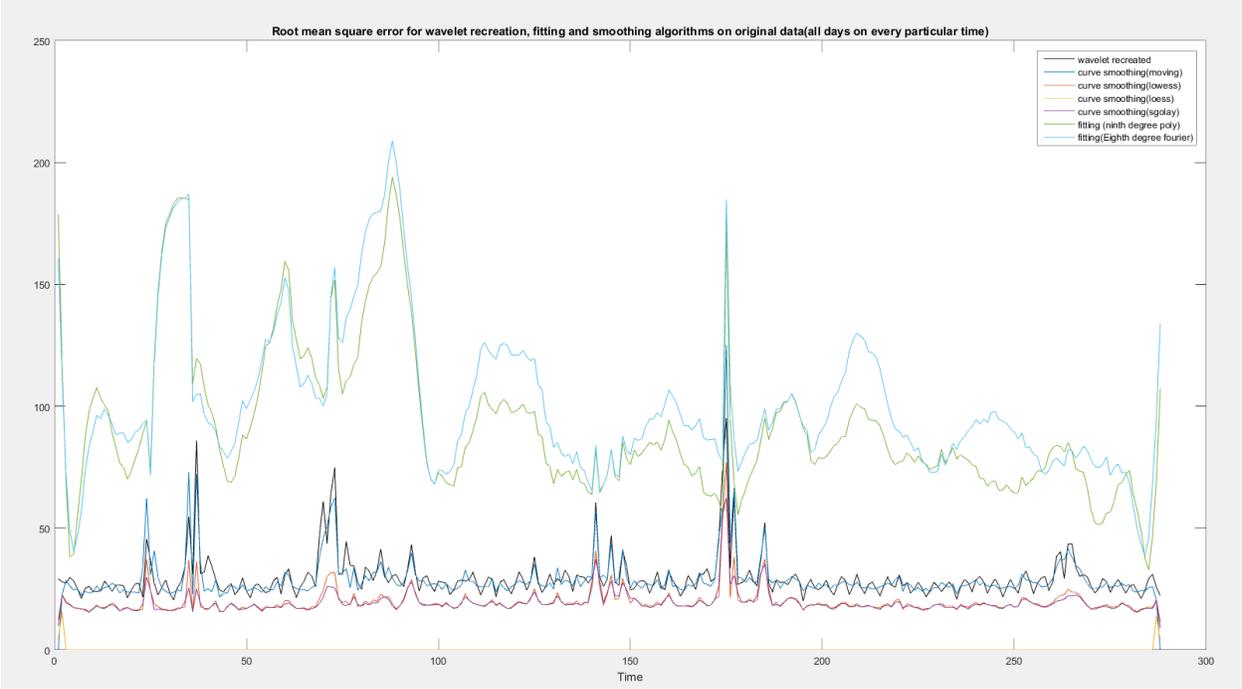


Figure 17: Root mean square errors for load

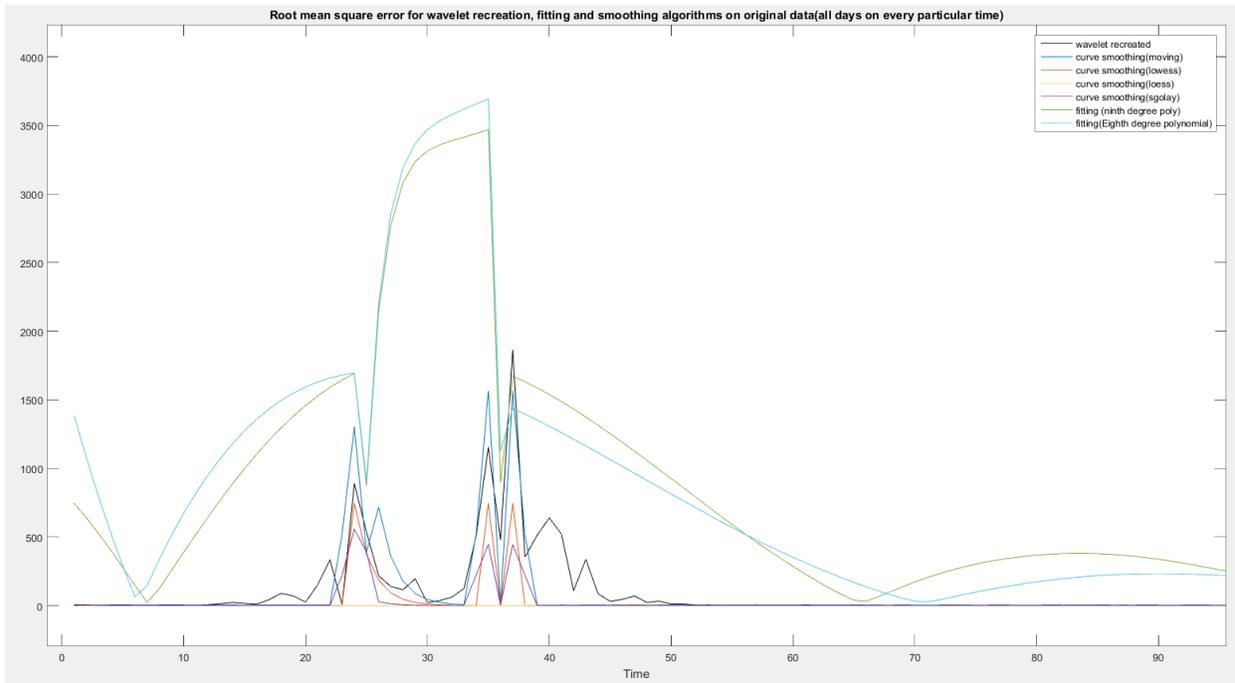


Figure 18: Root man square errors for wind power

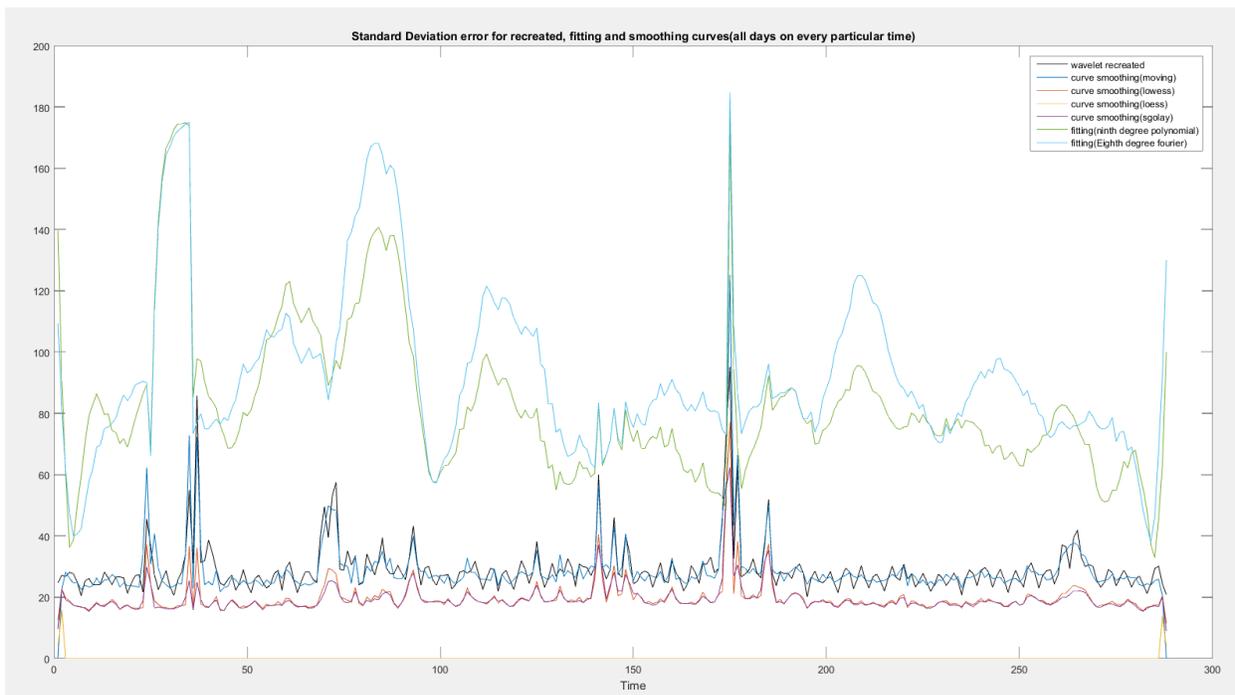


Figure 19: Standard deviations of errors of load

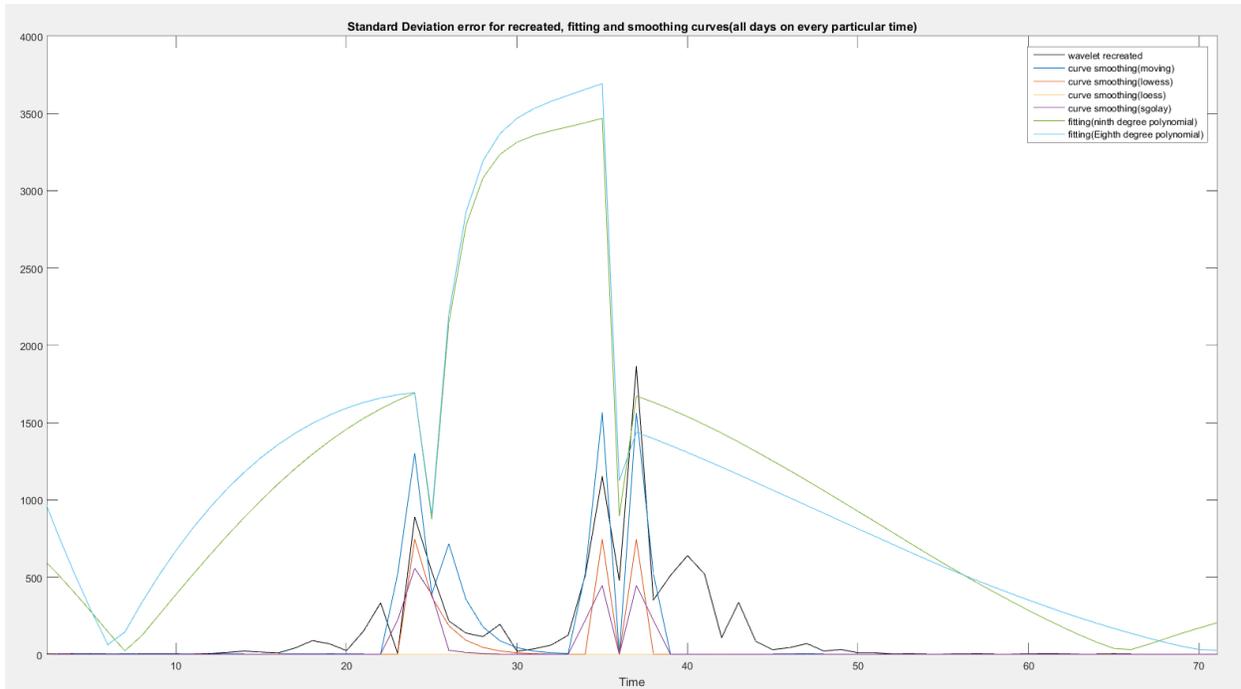


Figure 20: Standard deviations of errors of wind power

Compared to other fitting curves, wavelet recreated loads tend to have lower root mean square errors and standard deviations of absolute errors. While the errors compared to moving averages remain lower at most time indices, especially for the wind power plots, there are time indices where moving averaging produces lower errors. The moving averaging does not capture the volatility of the original data. Wavelet algorithm has well defined dips and peaks in the curve that indicates at the wavelet series being more responsive to the original data. Moving averages thus eliminate characteristic volatility of the original data series that might be of interest to modelers.

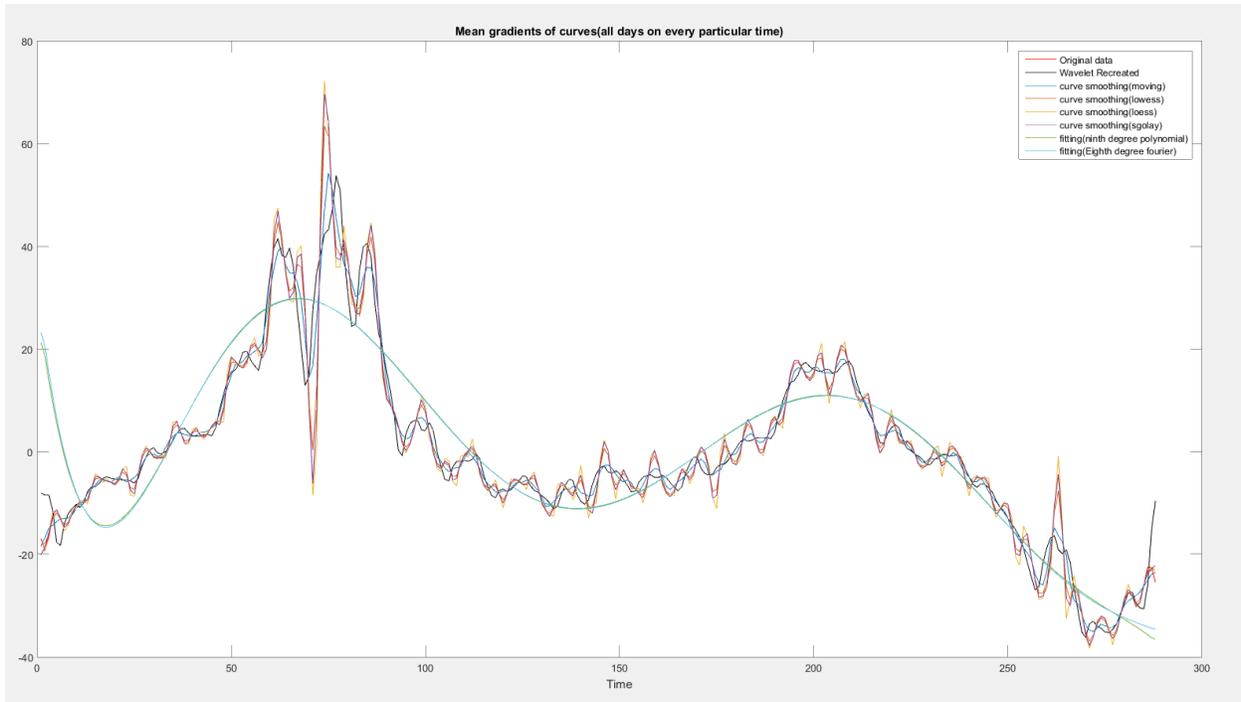


Figure 21: Mean gradients of load

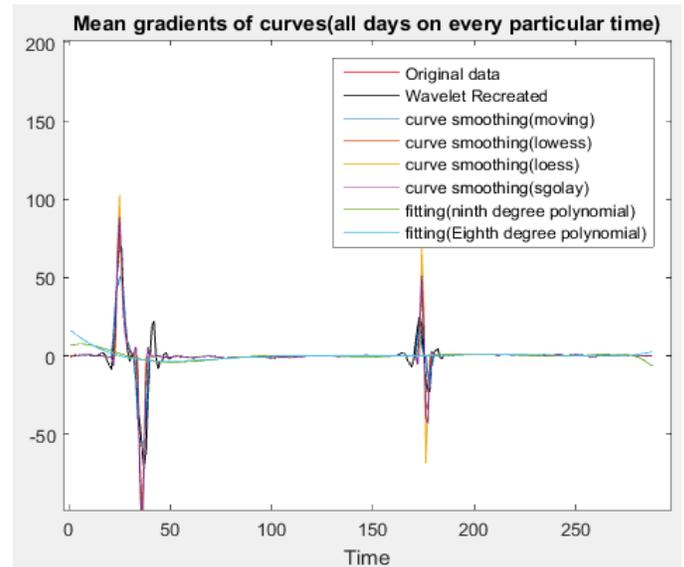
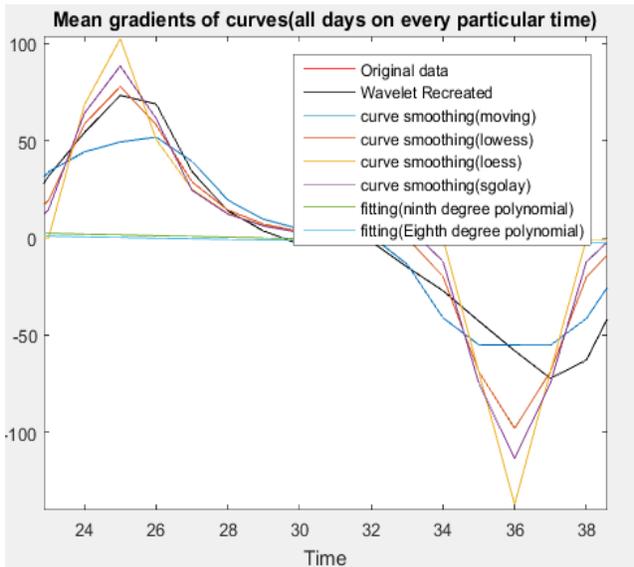


Figure 22: Mean gradients of wind power

Polynomial fitting curves are expected to be amongst all other curves due to a well-defined function being used to represent data. Wind power profiles show large gradient changes when there is an abrupt event changing the power values in a short span of time. The gradient curve of the wavelet recreate time series closely follows the gradient curves of the original data (as can be observed in Fig: 21), indicating that the characteristic shape of the original data is preserved in the

wavelet recreated series. Hence wavelet algorithm preserves the frequency domain information of the time series data indicated by the changes in amplitudes of time-series signal of original data being reflected in the changes in amplitudes of the wavelet recreated data series. Moving averaging algorithm tends to underestimate the gradients at most times and do not capture the characteristic changes of the original data. Peak characteristics and structures tend to get lost when using moving averaging.

To analyze the optimal decomposition level that can be achieved with lower errors, i.e. optimal white noise removal level

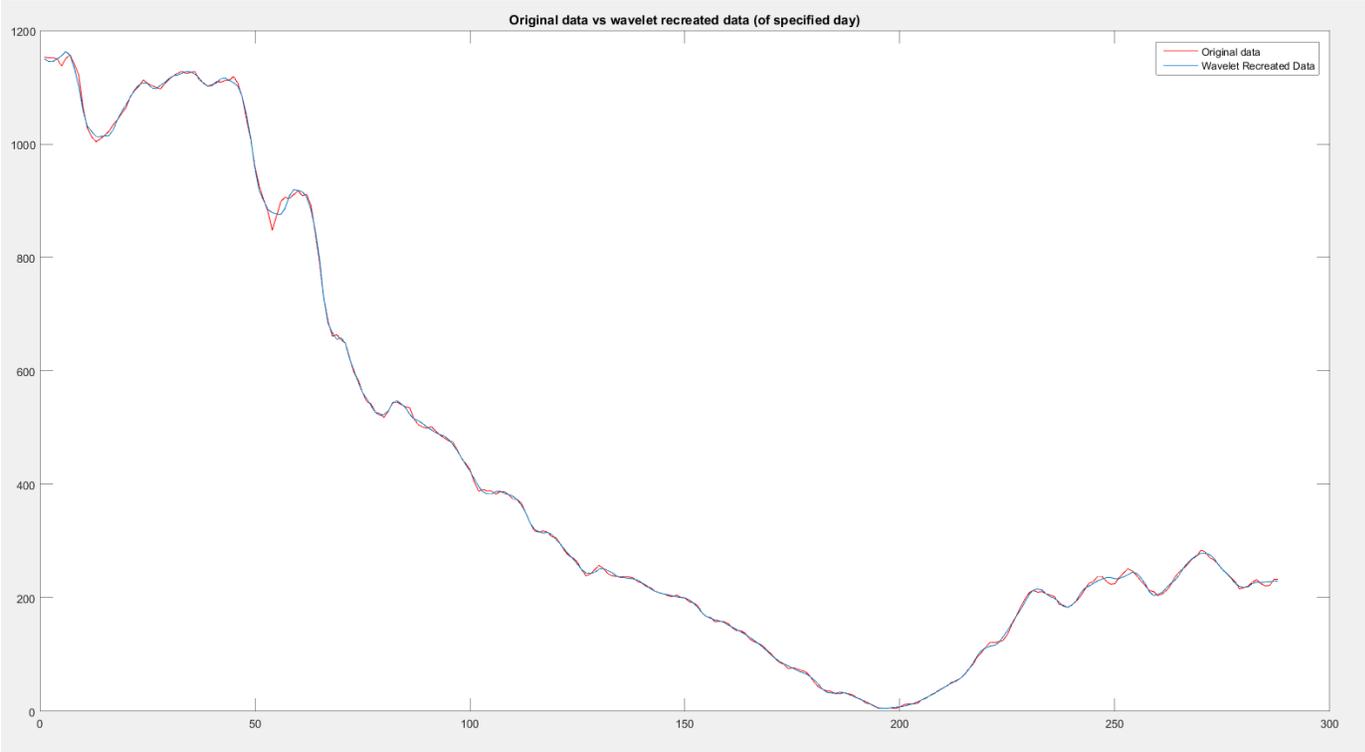


Figure 23: Wavelet recreation at decomposition level 1

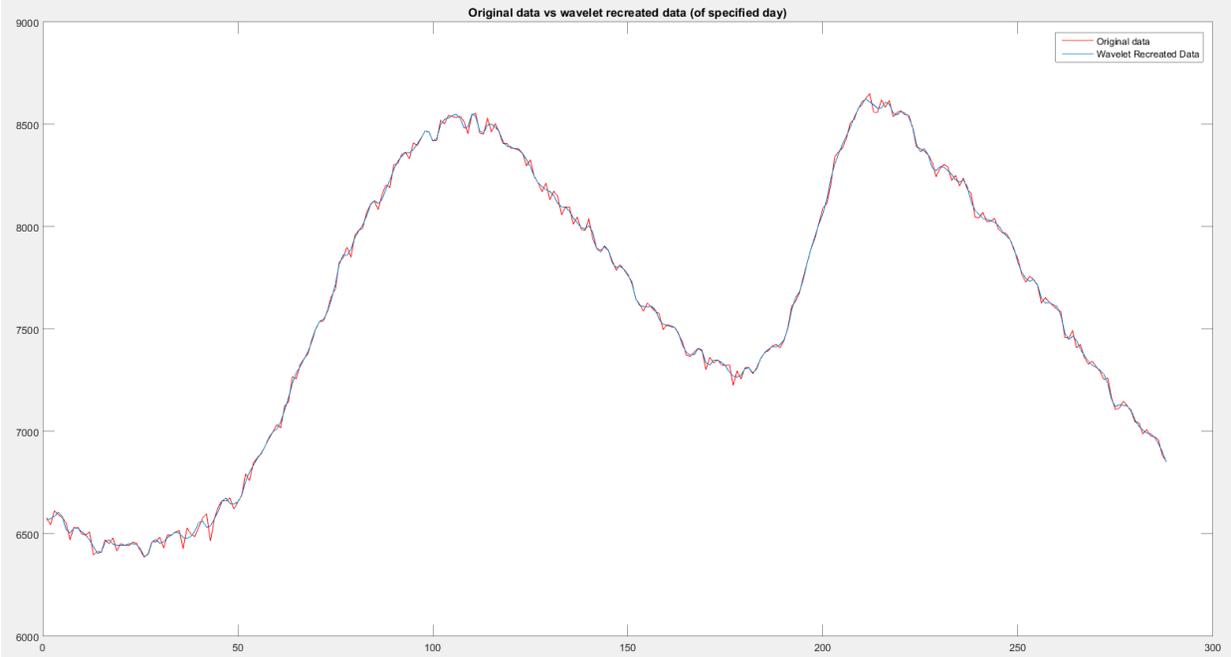


Figure 24: Wavelet recreation at decomposition level 2

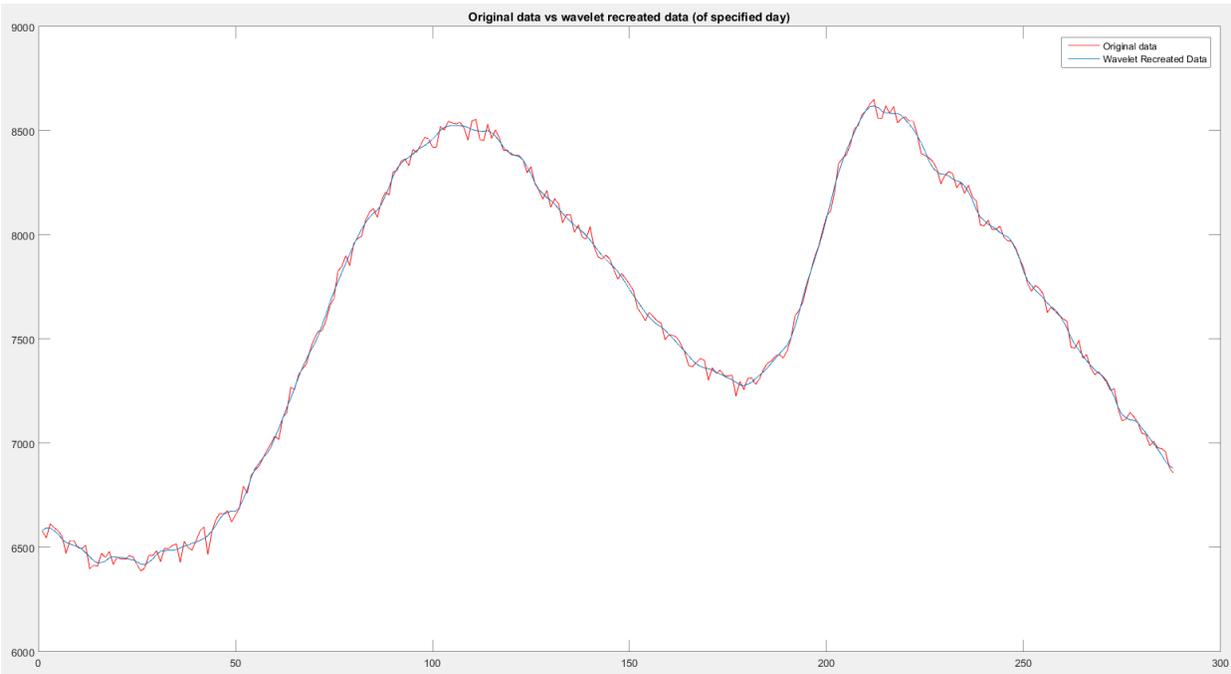


Figure 25: Wavelet recreation at decomposition level 3

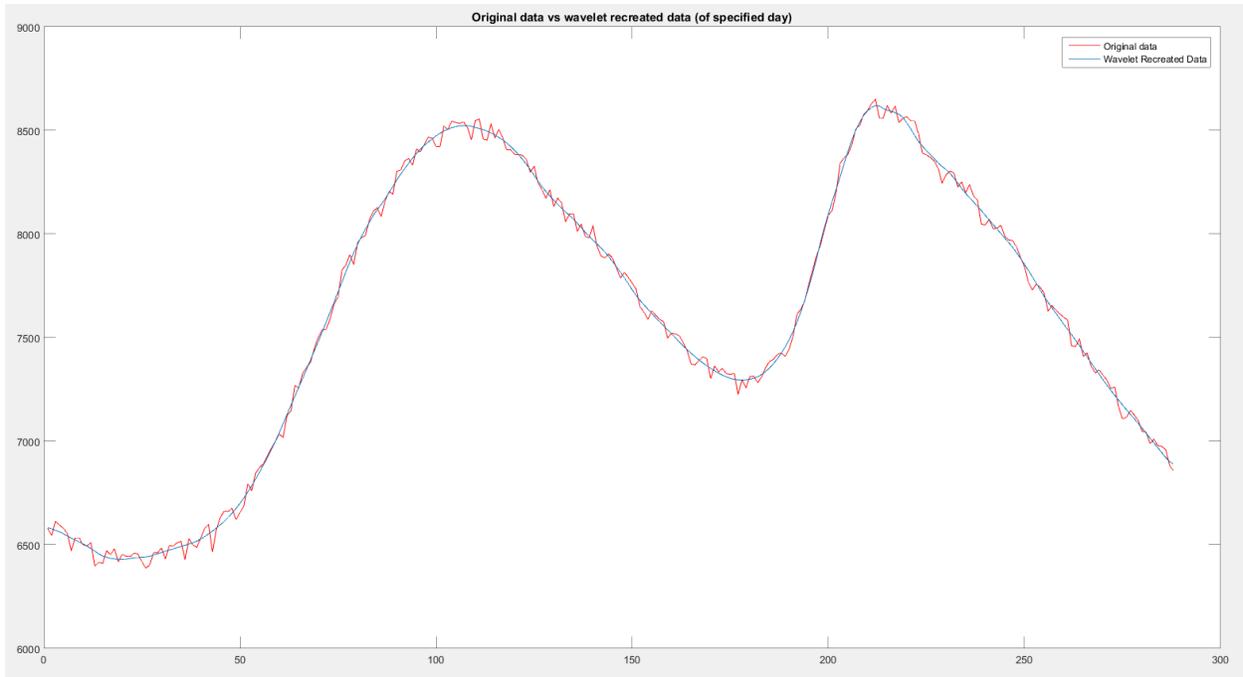


Figure 26: Wavelet recreation at decomposition level 4

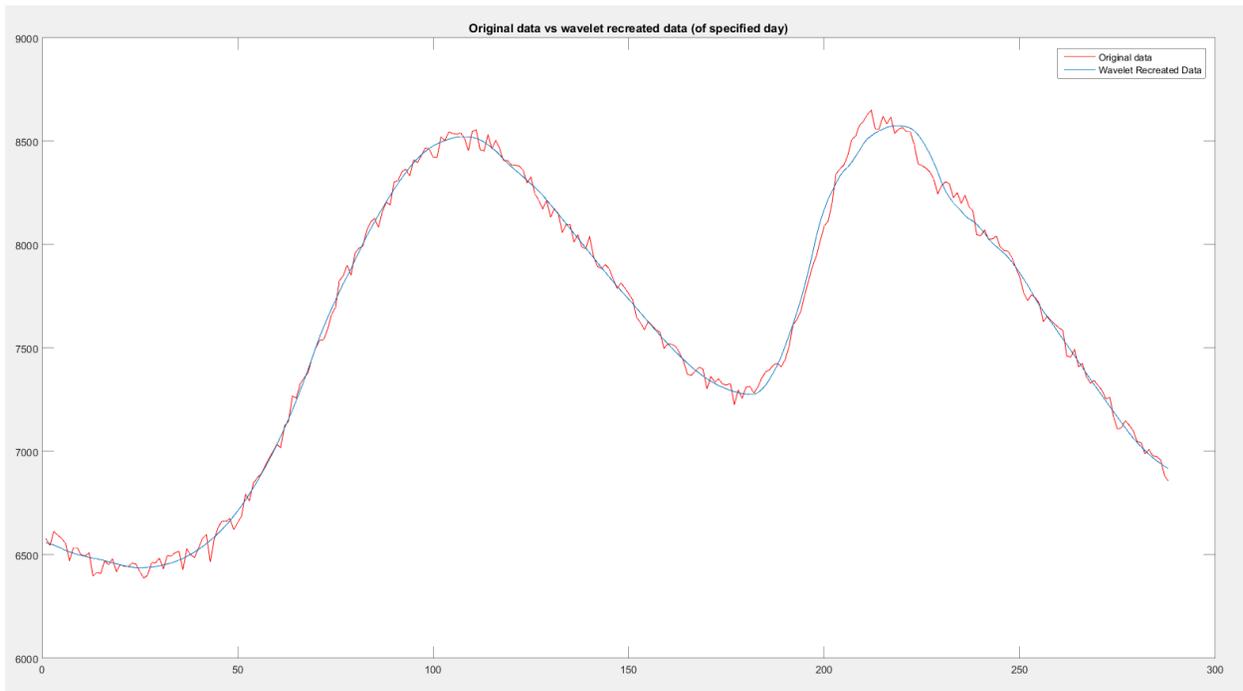


Figure 27: Wavelet recreation at decomposition level 5

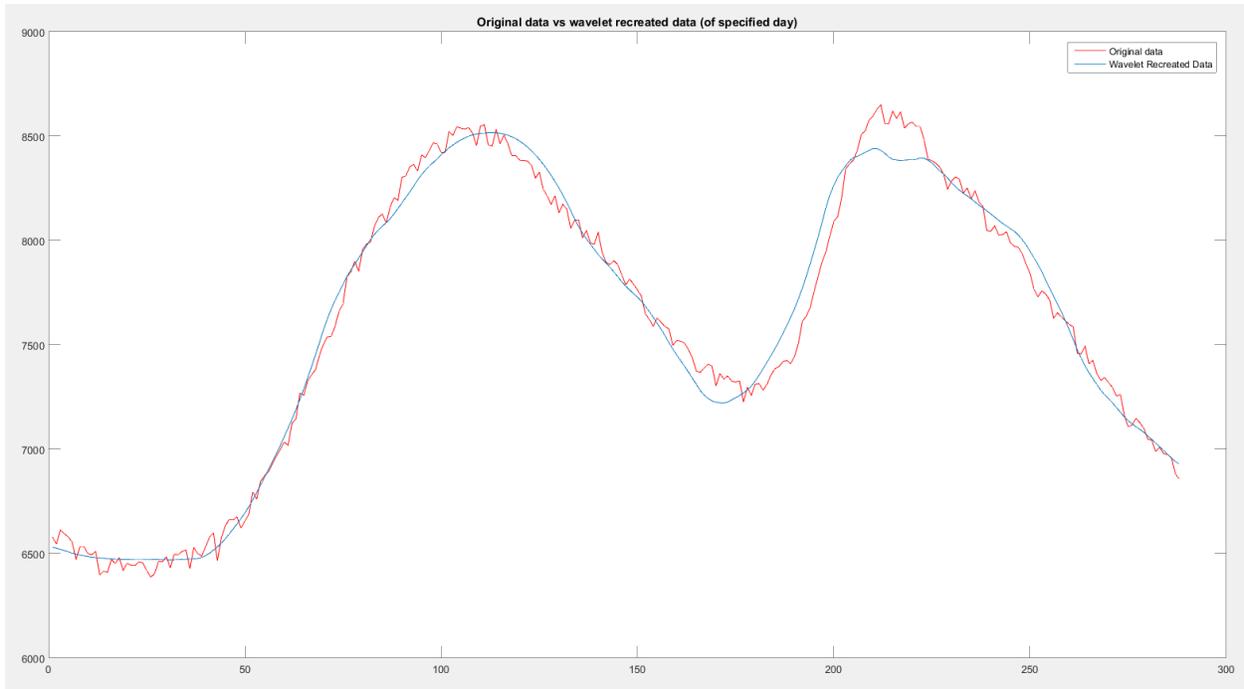


Figure 28: Wavelet recreation at decomposition level 6

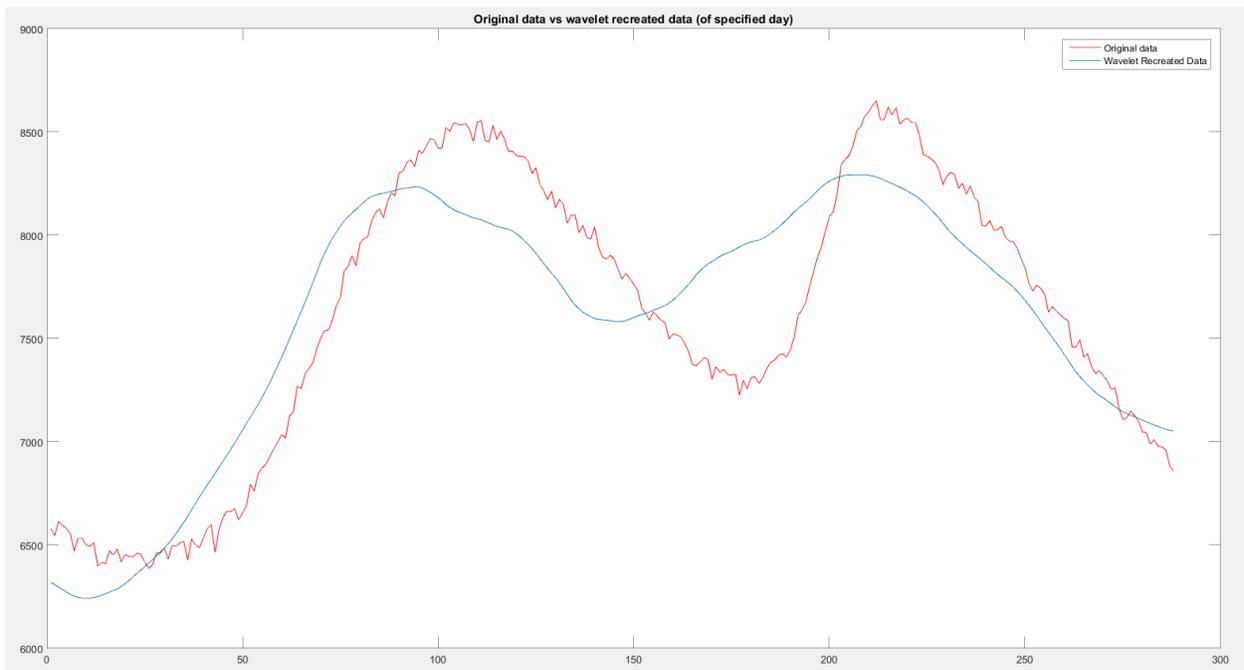


Figure 29: Wavelet recreation at decomposition level 7

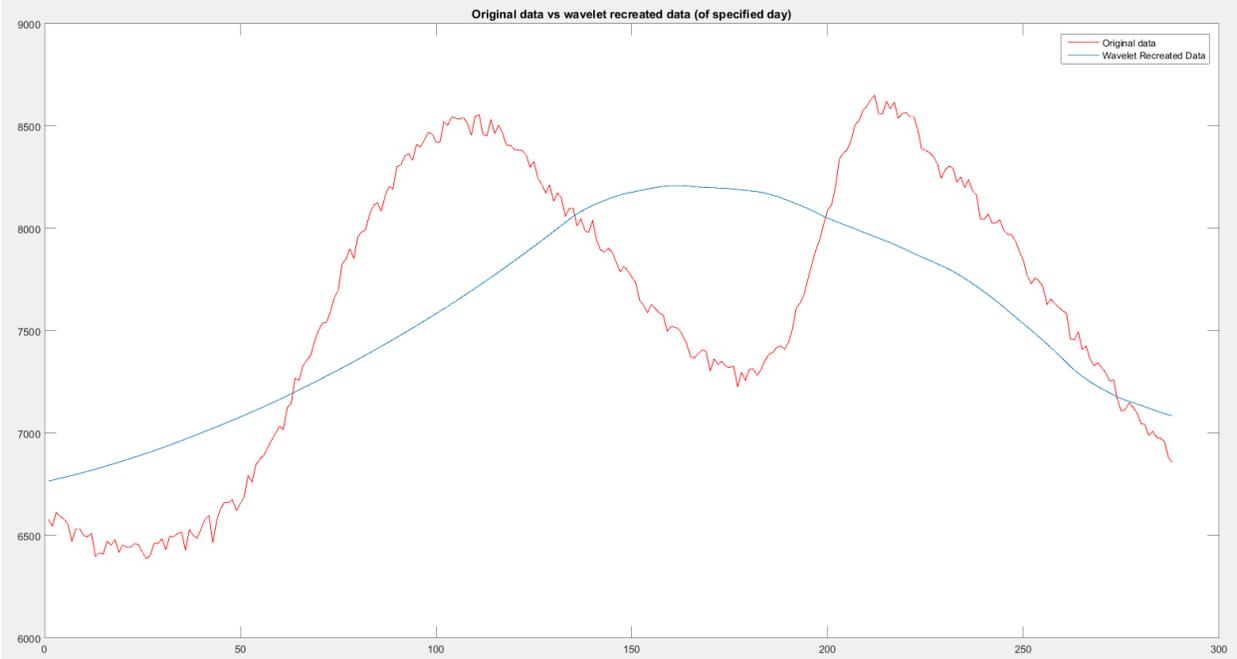


Figure 30: Wavelet recreation at decomposition level 8

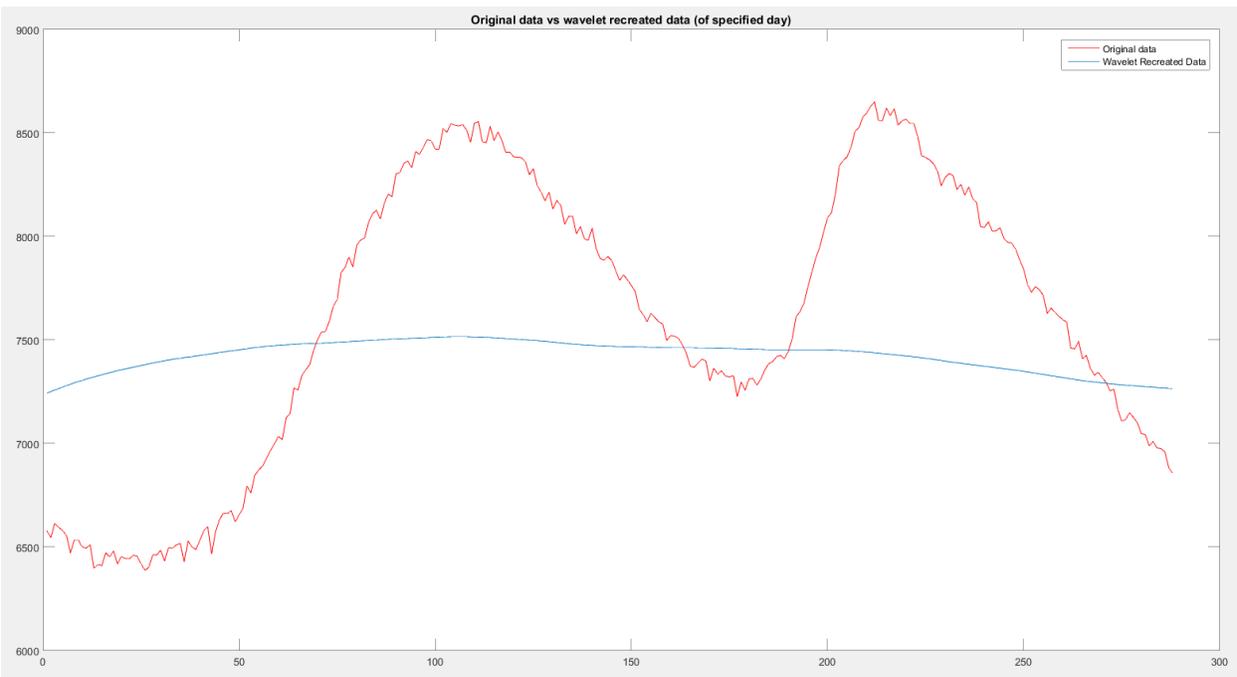


Figure 31: Wavelet recreation at decomposition level 9

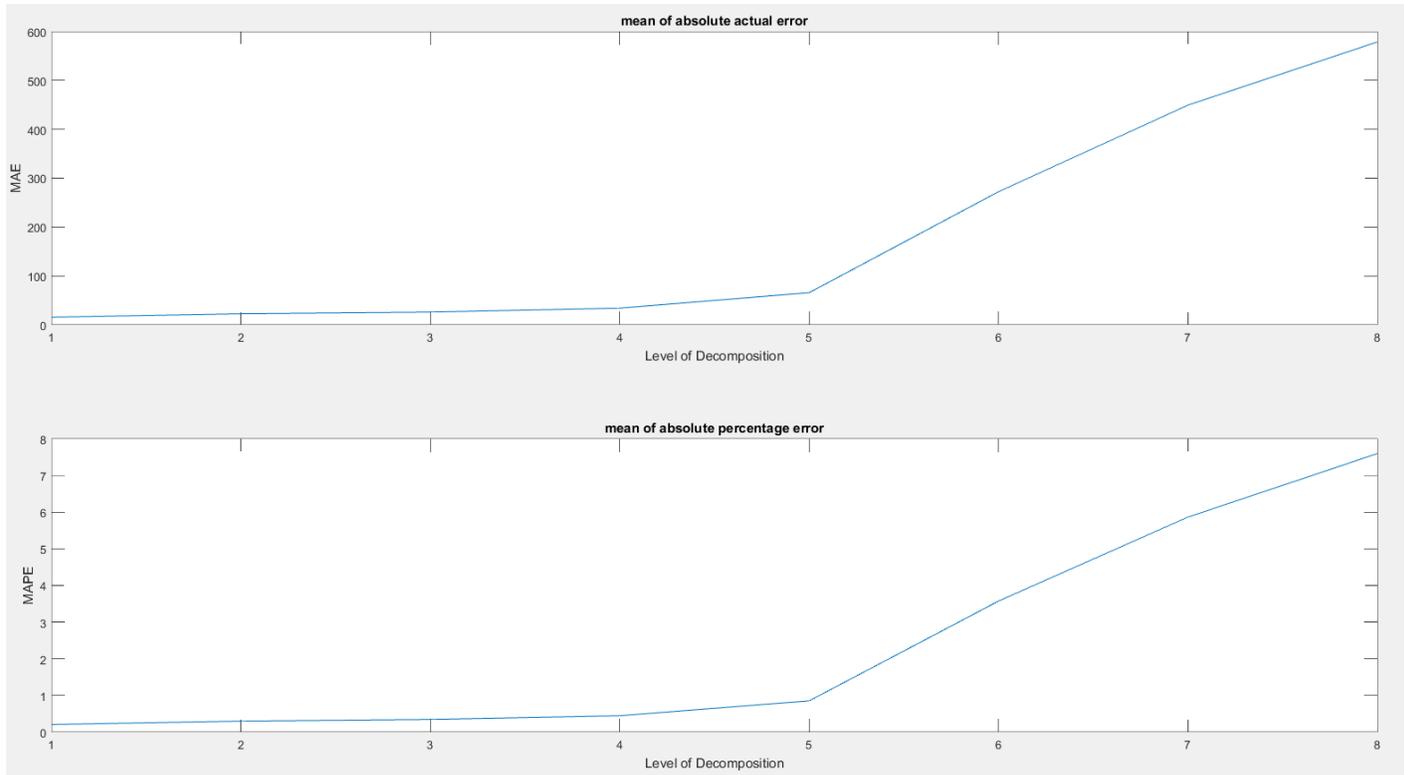


Figure 32: Mean of absolute errors and percentage errors at decomposition levels for load

Decomposition Level	Corresponding time resolution (min)	MAPEs – Load	MAPEs – Wind Power
1	10	0.20	1.27
2	20	0.29	1.91
3	40	0.34	4.40
4	80	0.44	5.58
5	160	0.85	14.65
6	320	3.57	61.96
7	640	5.86	48.75
8	1280	7.60	371.84

Wavelet transform curves for load curves recreate the original data series with percentage errors ranging from 0.2% to 7.6%. The errors increase gradually decomposition level 1 till 5 and then increases rapidly to level 8. The wavelet recreated signal tends to be shifted in the time domain from level 6 though still preserving the volatility of the original data.

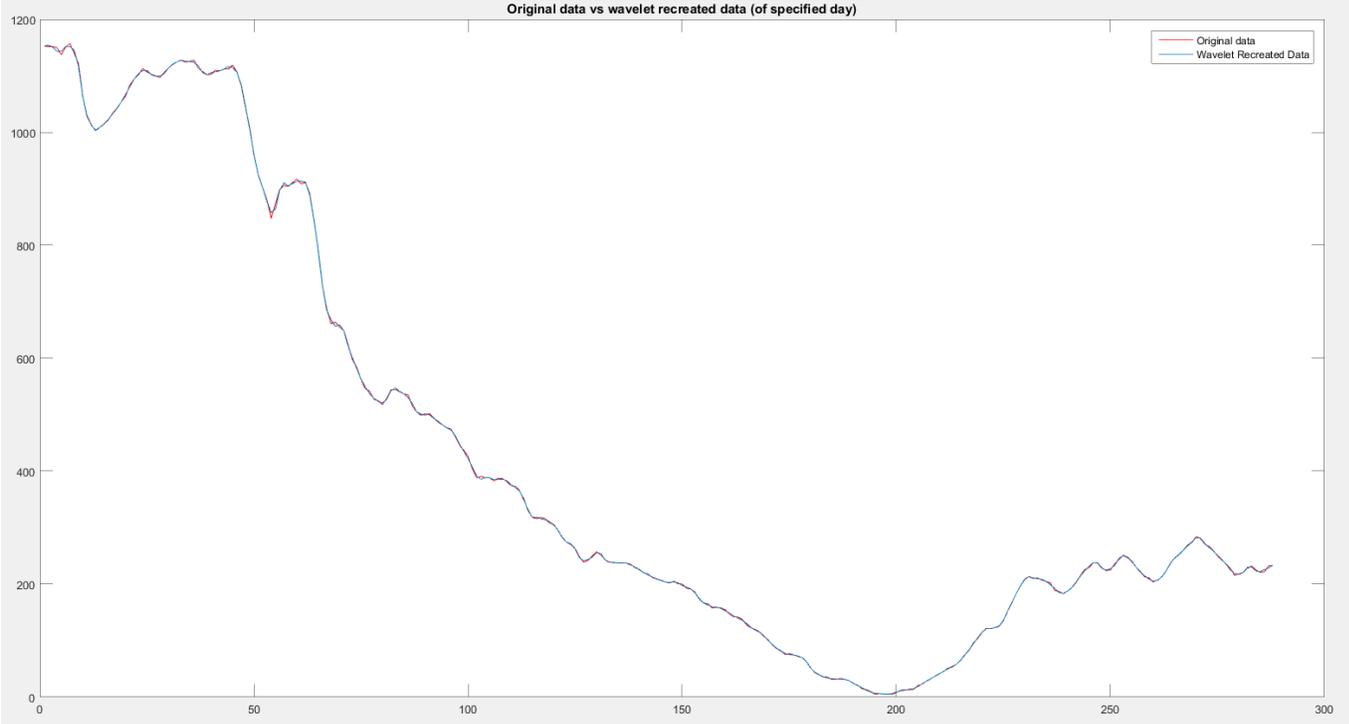


Figure 33: Wavelet Recreation at decomposition level 1 for wind power

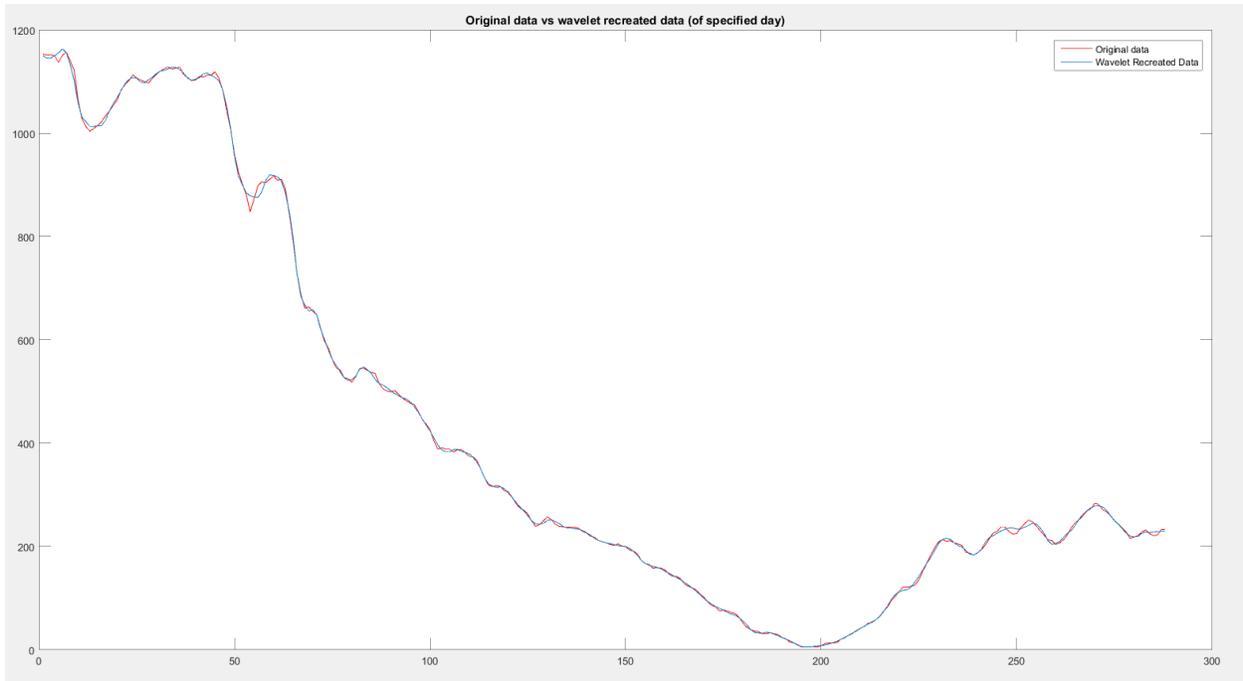


Figure 34: Wavelet Recreation at decomposition level 2 for wind power

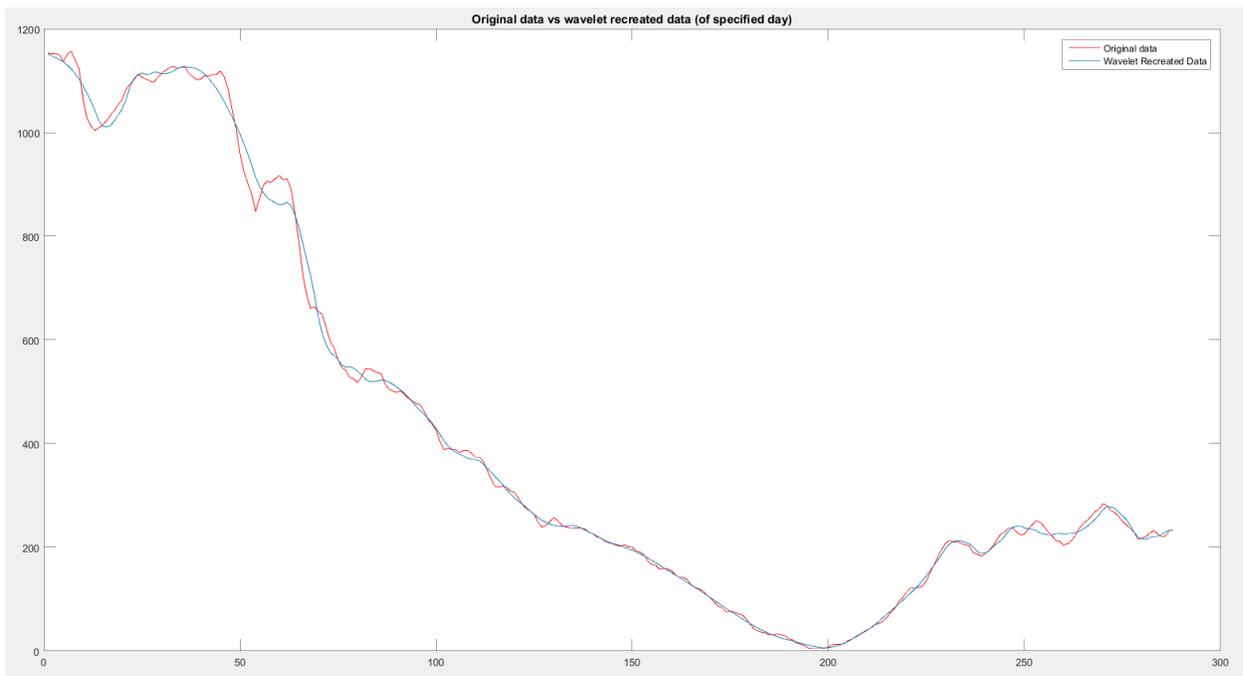


Figure 35: Wavelet Recreation at decomposition level 3 for wind power

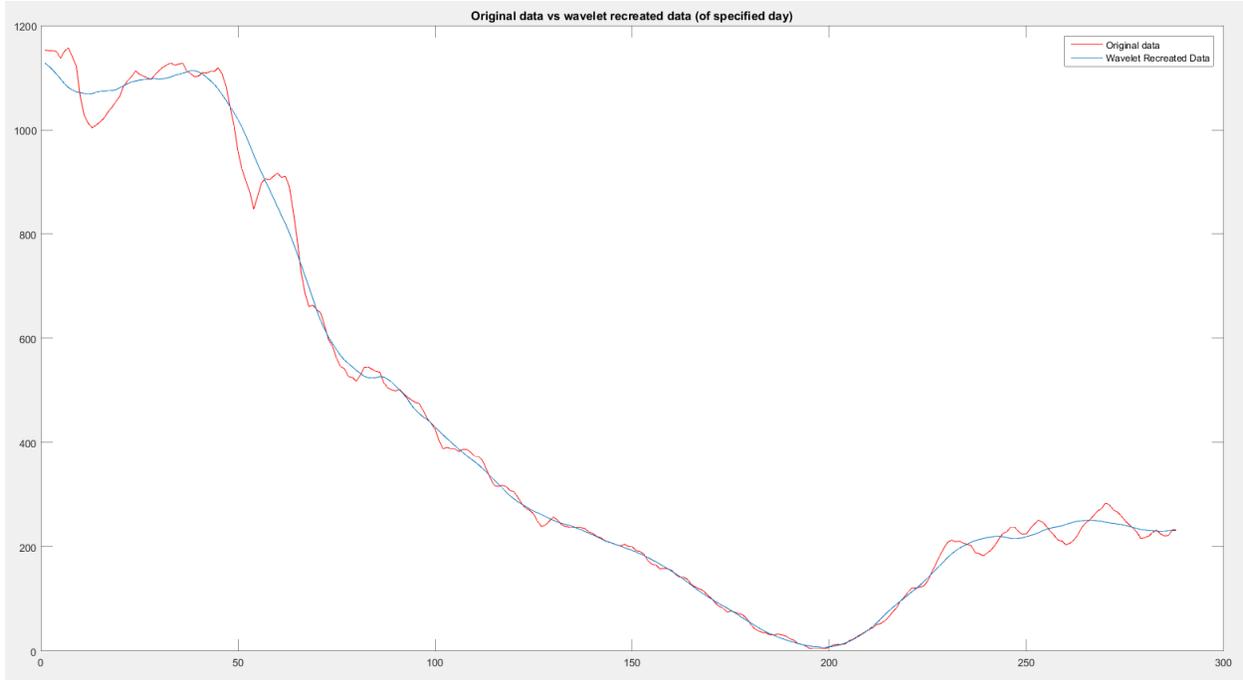


Figure 36: Wavelet Recreation at decomposition level 4 for wind power

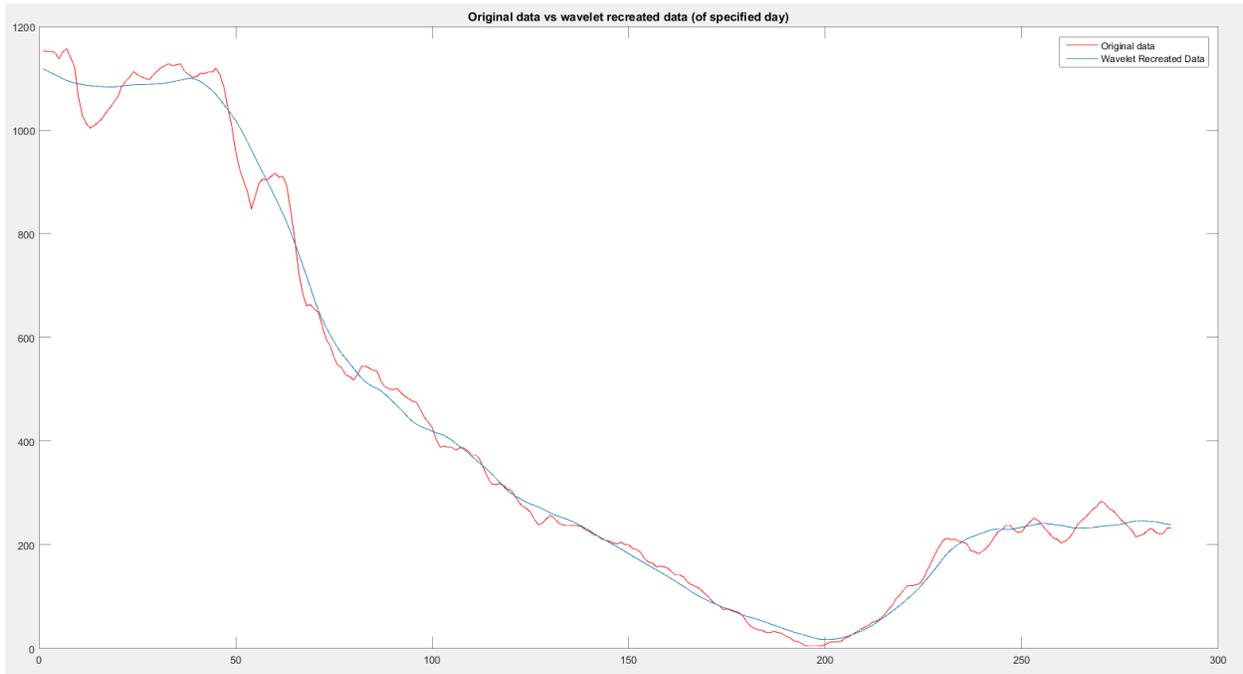


Figure 37: Wavelet Recreation at decomposition level 5 for wind power

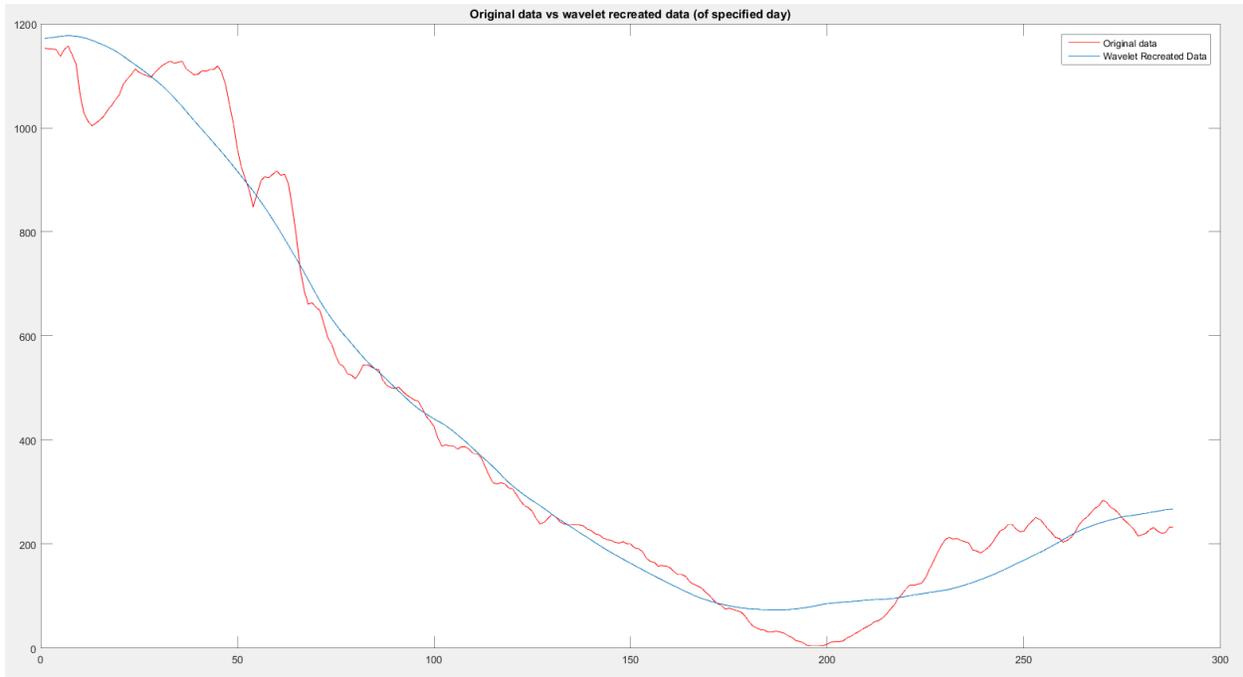


Figure 38: Wavelet Recreation at decomposition level 6 for wind power

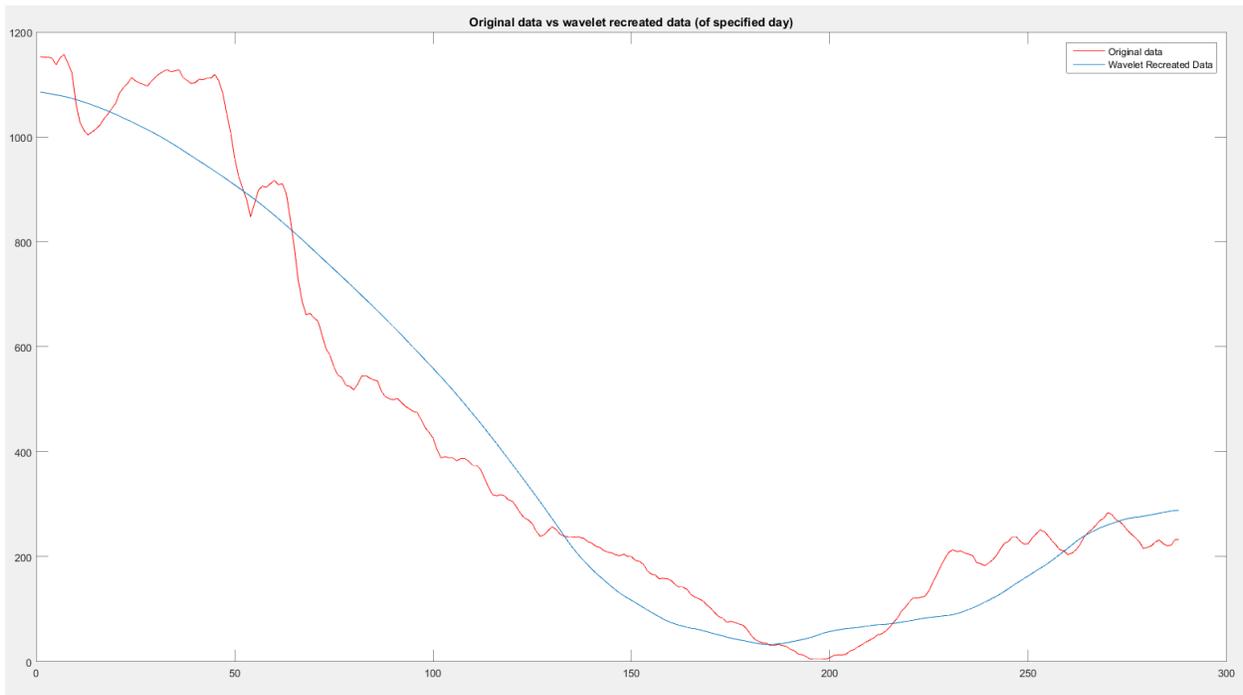


Figure 39: Wavelet Recreation at decomposition level 7 for wind power

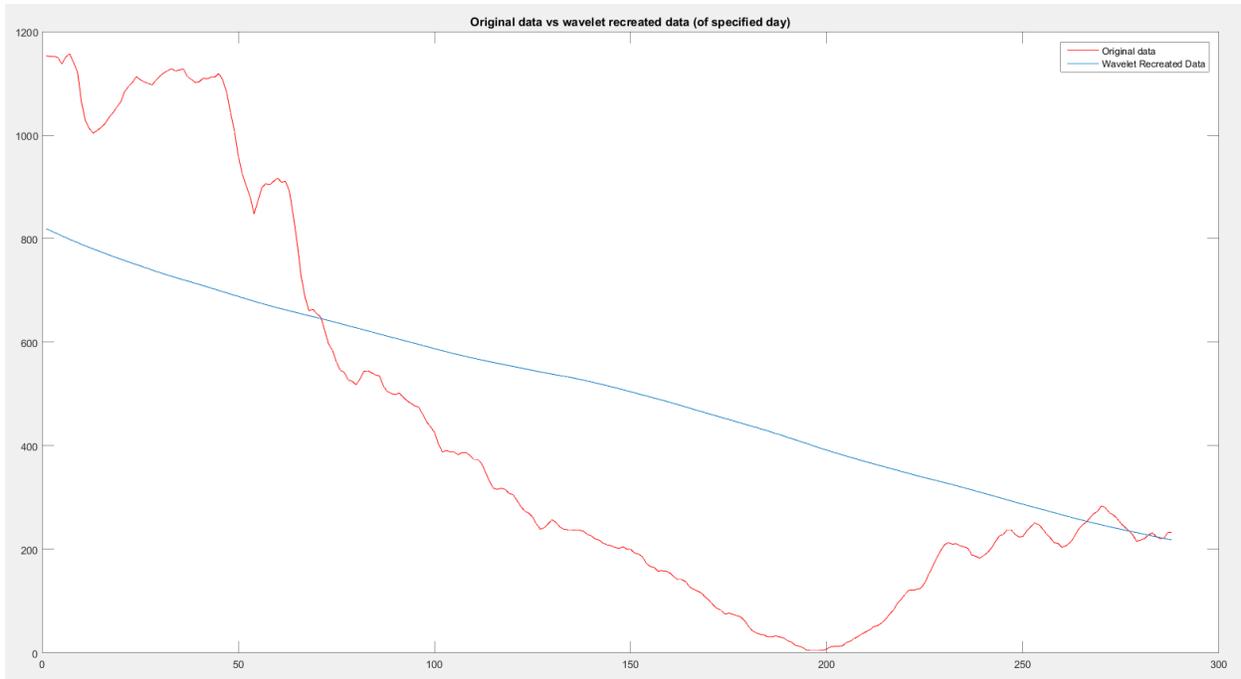


Figure 40: Wavelet Recreation at decomposition level 8 for wind power

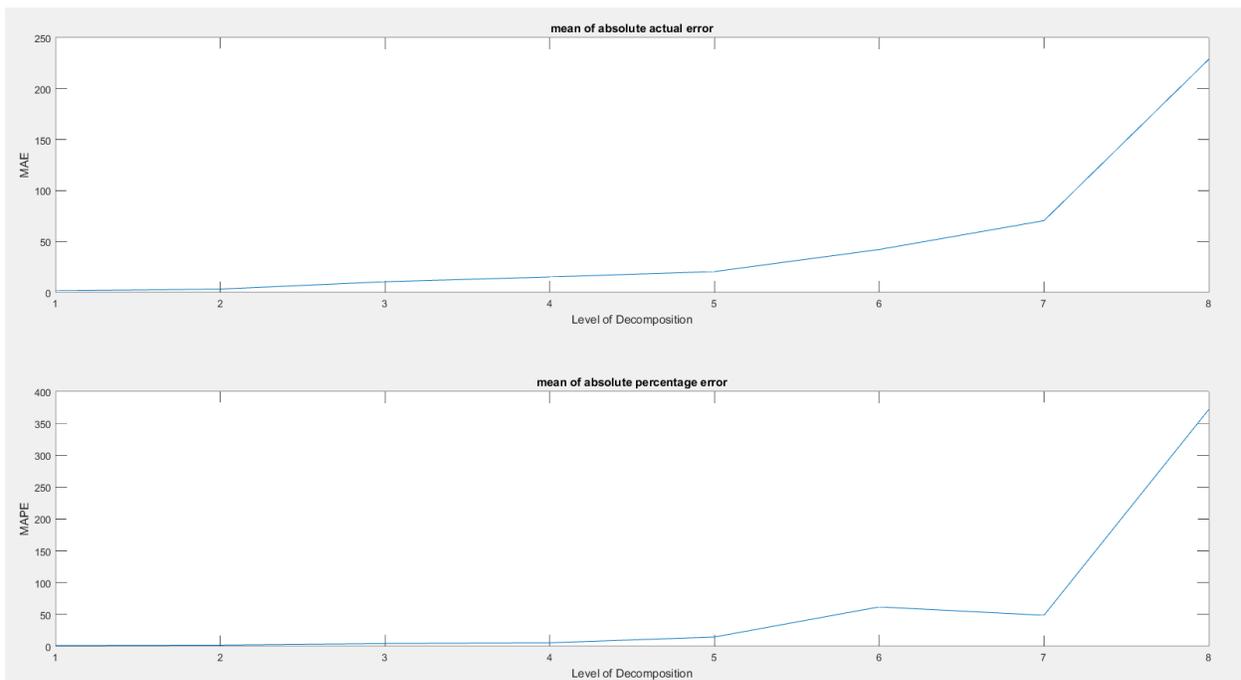


Figure 41: Mean of absolute errors and percentage errors at decomposition levels for wind power

Wavelet decomposition curves for wind power recreate the original data series with errors of around 2% till decomposition level 2. Level 3 onwards, the wavelet recreation starts to lose out information characteristic of abrupt changes in wind power profile with. The errors of

decomposition levels 3 and 4 are 4.4 and 5.5% respectively. The wavelet recreations at these decomposition levels thus capture the information to a broad extent with minimal losses at large inflexion points. Decomposition levels 5 to 8 show large percentage errors ranging from 14.65% to 371% in original data recreation implying significant loss of information. The percentage error at decomposition level 7 is lower than the error at decomposition level 6 though the absolute error is higher at level 7. The effect indicates the wavelet decomposition may present a positive as well as negative offset in the recreated curve though as a general trend, the errors increase with increasing decomposition levels.

Wavelet transformation curves till decomposition level 4 are therefore used in forecasting methods as they can provide a smoother curve devoid of “noise” to serve as a better base signal for forecasting. At a decomposition level, n , the recreated signal would have lost detail coefficients of time resolutions till $5 \text{ minutes} \times 2^n$. The first decomposition level would hence correspond to 10 minute time scale. Information contained in time resolutions lower than 10 minutes would have lost the characteristic details of the signal and instead would have been replaced by a base curve. Similarly, for decomposition levels 2 to 8, information would have been preserved for time resolutions 20, 40, 80 ... 1280 minutes.

An electric system operator requires proper scheduling of generators to ensure power supply at all time according to the physical ramping ability of systems. There are two main types of reserve systems available when the need arises: a) Spinning reserves for short term power requirements, b) Non-Spinning reserves for medium and long term requirements. Spinning reserves are generation units with oil, steam and combustion turbines that achieve an increase in power outputs through a turbine rotor. Non spinning reserves require a delay interval, like solar power and wind power, before the power can be connected to the main system of need. Spinning reserves in isolated grid systems may act as non-spinning generators too considering the substantial time required for the power to be delivered. Spinning reserves, with ramp rates of 20 minutes and smaller need to plan production according to the time scale the generator operates upon. A generator with a reserve ramp time of 20 minutes would not be concerned with information contained in a 10 minute resolution dataset. This extra information corresponds to noise that can interfere with forecasting method for scheduling operations. Thus, wavelet transformation eliminates this excess “noise” from the signal while still preserving the detail that is appropriate to the type of generating source.

Time resolutions of 5 to 20 minutes would be classified as fast ramping reserves with coal, natural gas and combustion turbine based sources serving the load through that profile curve, time resolutions of 40 to 160 minutes would be applicable to medium ramping generation sources such as solar power generators to plan timely delivery of power to the grid. Even higher ramping rate generators like wind, since, there are only a few dispersed hours when the wind blows, would only require correct load requirement information at 160 – 320 minutes time scales for onboarding a generator.

Wind power ramp errors represent the accuracy with which the wavelet recreated curve can represent the data pertaining to that time scale resolution. High resolution time scales (10 to 20 minutes) can provide substantial information that may not be required from 5 minute resolution data, without losing characteristics of the data itself while low resolution time scales (320 – 640 minutes) can analyze the hours-ahead requirement of wind energy within the day.

Decomposition Level	Corresponding time resolutions (min)	MAPEs – Load	MAPEs – Wind Power
1	10	0.20	1.27
2	20	0.29	1.91
3	40	0.34	4.40
4	80	0.44	5.58
5	160	0.85	14.65
6	320	3.57	61.96
7	640	5.86	48.75
8	1280	7.60	371.84

Time Resolution Manipulation

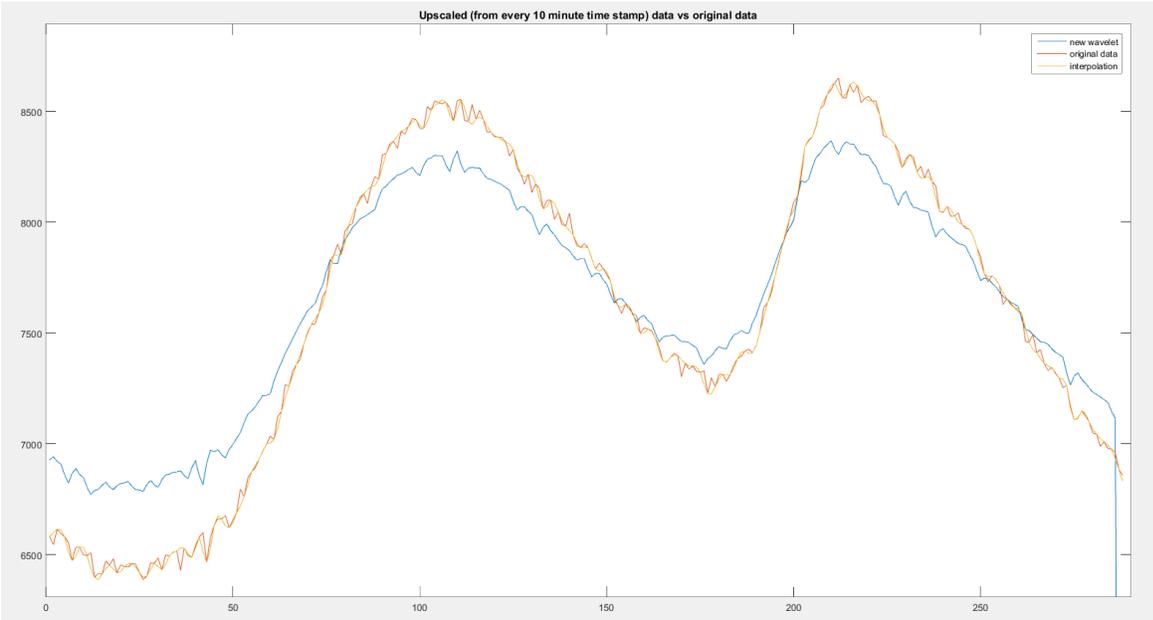


Figure 42: Wavelet recreation vs interpolation on load

Upscaling of load data from 10 minute resolution to 5 minute resolution through wavelet transform technique produces a data series with only 0.76 MAPE error. Observing the plot of the original data and wavelet reconstructed data, wavelet recreation works during the times of transition, with the errors largely during peaks and dips in the data. Interpolation still works better for recreating the original data series but the information lost in the wavelet recreation would be isolated through the detail coefficient, implying wavelet transform as a valuable method of upscaling data.

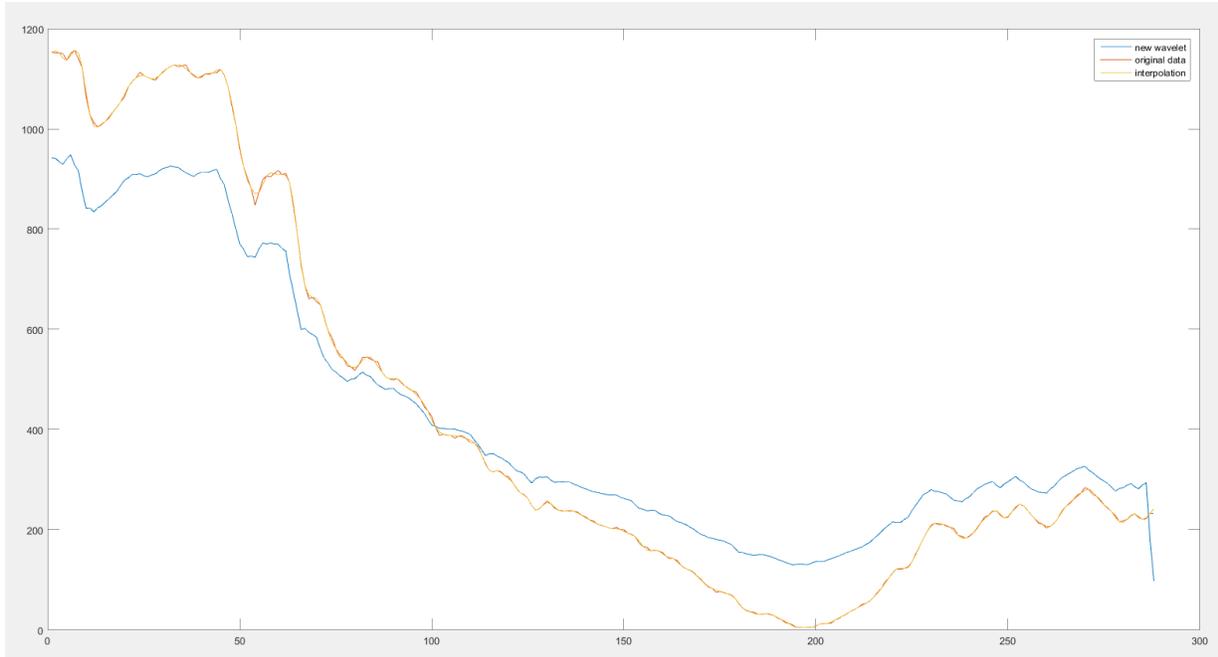


Figure 43: Wavelet Recreation and interpolation on wind power

Upscaling wind power data from 10 minute resolution to 5 minute resolution produces MAPE error of 1.172. From the plot, it can be observed that wavelet recreated wind profile captures the minute characteristics of the original data but overcompensates for large changes in the data. Wavelet recreated data does not accurately model the large dips and rises in the wind data. Since the wavelet curve is the base signal of the original data, the detail coefficients are the actual indicators of successful recreation. While wind power is zero at some points of time in the day, wavelet recreation does not signify it as zero, instead showing data values of the base curve generated.

Upscaling of wind data can thus be conducted using the wavelet transform technique but only for short terms and when the wind power does not approach zero in that course of time. The wavelet transforms grossly over predicts if the target is a very low or zero value.

Forecasting of time series

Artificial Neural Network (ANN)

Forecasting Errors for Load

Forecasting Horizon	Range of MAPEs (%)	Mean MAPEs (%)	Mean Gradients	Variance of Gradients
Year ahead	7.4 – 11.4	9.36	-1.63e-07	5.16e-13
Six month ahead	11.3 – 20.1	14.35	-2.46e-07	2.01e-13
Month ahead	0 – 66.3	1.25	-6.03e-06	3.46e-08
Day ahead	1.9– 13.43	8.66	-6.38e-06	3.60e-08
Hour ahead	0.4 – 3.8	1.38	-2.64e-05	3.08e-07

Forecasting Errors for Wind Power

Forecasting Horizon	Range of MAPEs (%)	Mean MAPEs (%)	Mean Gradients	Variance of Gradients
Day ahead	0– 99.27	23.69	7.16e-05	7.39e-07
Hour ahead	0 – 10	60.86	5.40e-05	7.09e-07

Forecasting errors are heavily dependent on the variability of data within that time scale and do not follow a set trend. Errors of six month ahead prediction are higher than year ahead predictions as well as month ahead predictions. Data volatility within those time scales is the determining factor of prediction values. While month-ahead forecasting has the lowest mean percentage, the range of mean percentage error is lowest in case of hour ahead forecasting.

The mean gradients follow an increasing trend as the time scales under consideration increases. Gradients follow the pattern Year > Six months > Month > Day > Hour. Thus, year ahead prediction are smoother than day ahead or hour ahead predictions.

Wavelet fed Artificial Neural Network (WANN)

Forecasting Errors for Load

Forecasting Horizon (Decomposition Level)	Range of MAPEs (%)	Mean MAPEs (%)	Mean Gradients	Variance of Gradients
Year ahead (1)	7.41 - 11.48	9.40	-1.68e-07	4.94e-13
Year ahead (2)	9.12 - 12.36	10.46	-1.39e-07	5.27e-13
Year ahead (3)	7.51 - 11.47	9.38	-1.49e-07	5.28e-13
Year ahead (4)	7.44 - 11.68	9.40	-1.17e-07	8.62e-13
Six month ahead (1)	11.84 - 21.85	14.4	-1.60e-07	1.82e-13
Six month ahead (2)	11.83 - 21.76	14.3	-1.15e-07	1.83e-13
Six month ahead (3)	11.79 - 21.68	14.5	-1.14e-07	2.07e-13
Six month ahead (4)	11.79 - 22.35	14.4	-7.67e-08	2.38e-13
Month ahead (1)	0 - 41.9	1.24	4.52e-06	4.15e-08
Month ahead (2)	0 - 23.4	0.58	-1.34e-06	3.77e-08
Month ahead (3)	0 - 31.0	0.83	2.15e-05	4.26e-08
Month ahead (4)	0 - 31.0	0.83	3.53e-05	3.16e-08
Day ahead (1)	1.8 - 43.9	5.18	2.86e-06	4.35e-08
Day ahead (2)	1.79 - 12.8	4.63	-3.90e-06	3.64e-08
Day ahead (3)	1.79 - 11.3	4.43	2.09e-05	4.35e-08
Day ahead (4)	1.72 - 14.6	4.90	3.18e-05	3.22e-08
Hour ahead (1)	0.38 to 2.76	0.84	-1.34e-06	1.27e-07

Hour ahead (2)	0.32 to 3.06	1.05	-6.33e-06	1.09e-07
Hour ahead (3)	0.385 to 2.82	1.18	1.14e-05	7.60e-07
Hour ahead (4)	0.512 to 2.21	1.15	2.55e-05	9.35e-08

Mean errors remain approximately the same for every decomposition levels with most variations in $1e-02$ magnitude. Different decomposition levels are preferable for different data time scales (by considering mean MAPEs); year ahead predictions are most suitable with decomposition level 3; six month predictions with decomposition level 2; month ahead predictions with decomposition level 2; day ahead predictions with decomposition level 3; hour ahead predictions with decomposition level 1. Also, as noted previously, six month ahead errors are higher than year ahead as well as month ahead errors.

Wavelet fed Artificial Neural Network performs has greater errors when analyzing large time scale periods. For time scales of year ahead and six month ahead forecasting, errors at every decomposition level are greater than the errors from conventional Artificial Neural Network model. Errors increase by a magnitude of $1e-02$. Wavelet transformed curves thus hold no advantage over conventional machine learning forecasting when large time scales are under observation. In the case of short scale forecasting, wavelet inputs produce much better results. For month ahead time scale, decomposition level 2 produces mean errors that are half of conventional forecasting. Day ahead forecasting at decomposition level 3 also halves the errors, and decomposition level 1 for hour ahead forecasting reduces the forecast error by 0.54%, i.e. a 40% reduction.

Forecasting Errors for Wind Power

Forecasting Horizon (Decomposition Level)	Range of MAPEs (%)	Mean MAPEs (%)	Mean Gradients	Variance of Gradients
Day ahead (1)	0– 99.2	31.02	7.16e-05	8.39e-07

Day ahead (2)	0 – 98.9	35.21	5.40e-05	7.75e-07
Day ahead (3)	0 – 98.29	39.41	1.50e-05	5.98e-07
Day ahead (4)	0 – 99.65	41.31	2.57e-05	6.26e-07
Day ahead (5)	8 – 99.67	39.96	6.73e-05	5.26e-07
Day ahead (6)	1.68 – 98.44	40.26	6.95e-05	7.59e-07
Day ahead (7)	0.52 – 98.37	40.12	4.16e-05	8.60e-07
Day ahead (8)	0.22 – 97.02	31.93	1.09e-05	8.02e-08
Hour ahead (1)	0 – 100	65.21	1.84e-05	8.39e-07
Hour ahead (2)	0 – 100	82.60	6.91e-05	7.75e-07
Hour ahead (3)	0 – 100	91.30	1.50e-05	5.98e-07
Hour ahead (4)	49.60 – 99.83	74.47	2.57e-05	6.26e-07
Hour ahead (5)	10.40 – 97.35	33.63	6.73e-05	5.26e-07
Hour ahead (6)	15.55 – 77.18	30.97	6.95e-05	7.59e-07
Hour ahead (7)	1.44 – 65.80	17.86	4.16e-05	8.60e-07
Hour ahead (8)	1.55 – 86.18	20.13	1.09e-05	8.02e-08

Mean errors of day ahead forecasting from WANN forecasting model are much larger than mean errors from conventional ANN model. Wavelet recreation of data is ineffective in capturing the characteristics of daily wind power profiles as was inferred from wavelet recreation curves not overestimating zero values and underestimating peaks. But, in the case of hour ahead forecasting, WANN forecasting model dramatically improves forecasting ability. Mean percentage error drops from 60.8% from conventional ANN model to 17.86% at decomposition level 7 WANN model, a 70% reduction in error.

Results from the WANN model can be inferred in two ways according to the requirement from the forecasting model. If a least error model is required, the WANN decomposition is selected such that the mean of MAPEs is lowest. Or, if the smoothest curve is required (smooth curves may be required in cases where uncertainty of data needs to be eliminated), decomposition level is chosen such that the mean gradient is closest to the gradient of the presample data (data being used to forecast). Decomposition levels 3 and 4 of WANN model have lower variance of gradients than ANN forecasts for day ahead forecasting while decomposition levels 3, 4 and 5 have lower variances than ANN forecasts for hour ahead. It should also be noted that the mean errors are lowest for high decomposition levels for hour ahead forecasting. This is owing to the fact that the Wavelet Transform algorithm produces a stable basis curve that can convolve better to the neural network and produce more accurate results. Higher the decomposition level used, more stable is the generated wavelet transform curve and result in a better forecasting performance.

Auto Regressive Integrated Moving Averaging (ARIMA)

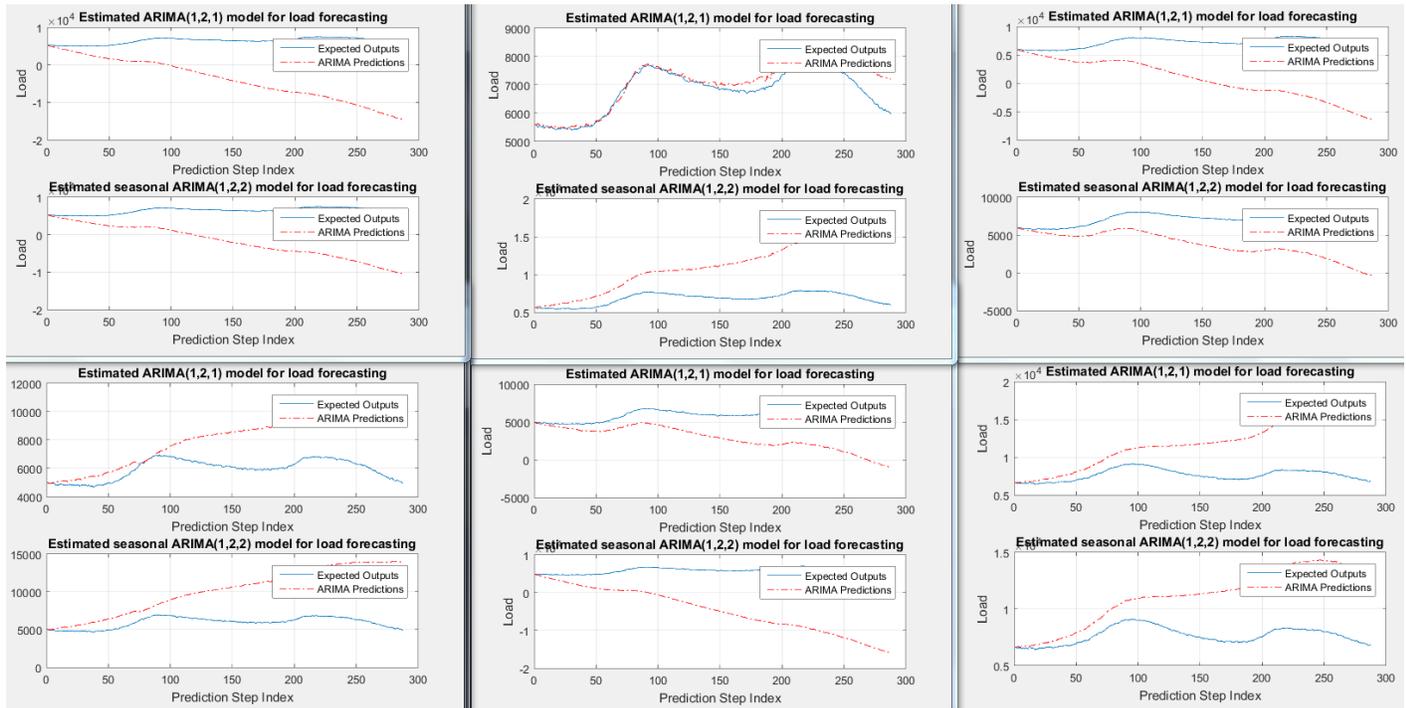


Figure 44: ARIMA simulations of load forecasting

The range of MAPEs for the first (1, 2, 1) model is 0% to 155% while the mean of MAPES is 57.17% and the range of MAPEs for the second (1, 2, 2) model is 0% to 171% while the mean of MAPES is 71.43%.

ARIMA simulations of load data depicts that using too many MA terms in the model can prove counter-beneficial while predicting. A single moving average term is optimal in this 5 minute resolution scenario. The ARIMA (1, 2, 1) is the best ARIMA model that is possible for the data series but still the MAPEs are very high at 57% while the ANN forecasted MAPE was 8.66% and WANN forecasted MAPEs were even lower at around 5%. Thus, while ARIMA is good for predicting short term values early in the data series, the errors grow as the data series progresses. Hence, ARIMA model would be suitable for forecasting of load on a monthly/yearly time resolution data but is far inferior to ANN or WANN based forecasting models.

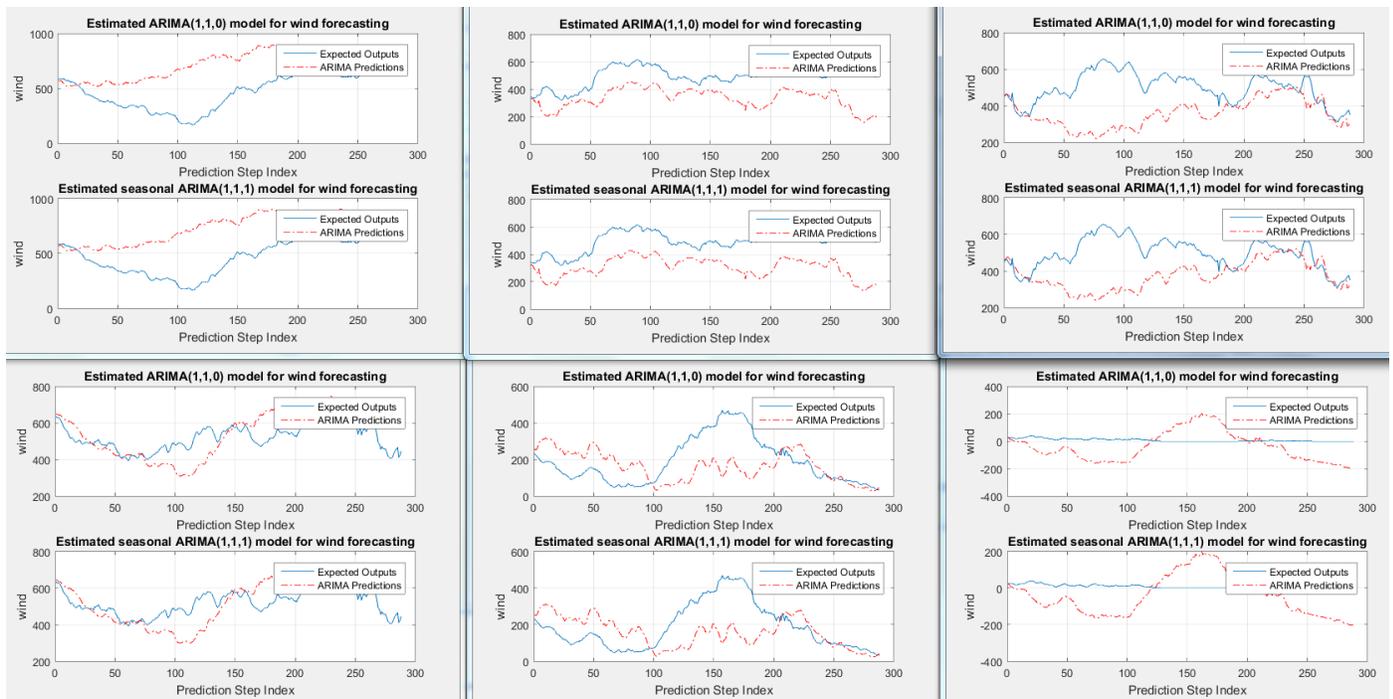


Figure 45: ARIMA simulations of wind power forecasting

The range of MAPEs for the first (1, 1, 0) model is 0 to 309% while the mean of MAPES is 85.2% and the range of MAPEs for the second (1, 1, 1) model is 0 to 313% while the mean of MAPES is 85.9%.

ARIMA models used for predicting high resolution data are highly variable in producing results. ARIMA outputs are close to the target outputs only during some times of the day. ARIMA outputs do not perform well capturing variability of data values. These tend to be much less responsive than the targeted values. ARIMA forecasts work well for short term predictions, accuracy is higher in the early part of the time series and tend to lose out on details as the time series progresses. Zero values in the presample response data also makes the ARIMA predictions non-usable in cases as the ARIMA output tends not to stay at stationary levels due to inherently assigned trend to the model.

Conclusions

The current study explores wavelet transforms as a potential algorithmic approach to analyze high resolution energy data. High resolution data is difficult to manipulate and analyze due to the presence of greater volatility. Wavelet transforms hold the advantage of working in the time domain while manipulating frequency information by decomposing the data into a base data component and a detail component to isolate the information present in the series. Wavelet recreated series are smoother than the conventional moving averaging method. While moving averaging is a simple mathematical operation, wavelet algorithm requires more steps and computation power to implement but results in a smoother time series that defines the data better than any polynomial fitting function can.

The decomposition levels associated with the wavelet analysis are all applicable under different uses. For short scale forecasting, higher decomposition levels work best though over a large time scale, higher decomposition levels also result in the most loss of the detail component. Thus, different decomposition levels need to be implemented under different applications. Data analysis would be most relevant within decomposition levels 1 to 4 while short term forecasting of highly variable wind data benefits most from higher decomposition levels. The selection of decomposition levels thus require careful consideration to choose applicability to the objective at hand.

For forecasting methods of high resolution data, while wavelet transforms do not contribute much towards large scale forecasting, considerable improvement in short term forecasting can be achieved. Since very long term, year ahead or six month ahead forecasts, are anyways impractical to be implemented in real life owing to the changing nature of consumer behavior and energy

production cycles, the WANN forecasting technique can find implementation in forecasting reliable load and wind power future data. ARIMA models while conventionally used as forecasting models for low resolution data, are far inferior to Artificial Neural Networks when processing high temporal resolution data. ARIMA models fail to capture the complex variability of energy data. WANN model can therefore be considered an improvement over conventional ANN models of prediction. Tweaking the models further may produce better results.

Further Work

While this project analyzes the use of Wavelet Transform algorithm for time scale manipulation and machine learning forecasting of load and wind power, the scope of the project could be expanded by analyzing the “noise” signal, i.e. the detail coefficients obtained at each decomposition level of the wavelet transform, which gets isolated through the wavelet technique. The “noise” separated from the base curve obtained through the wavelet recreation curves in the energy data context would contain valuable information characteristic of the energy scenario. The shape of the noise curves could provide important insights about consumer power use and also study variability of wind power generation in greater detail. Analyzing load profiles through wavelet transforms can yield valuable information through isolating the spikes, dips and other structures within the data series. A situation where Wavelet transforms can work really well would be analyzing domestic energy usage through smart meter data. Sudden variations in the base load would isolate appliance-level energy usage information. Wavelet transforms for wind power generation could capture abrupt changes in power generation profiles that may stem through variability of wind speeds or mechanical conditions of the turbine machinery. Time resolution aspects of Wavelet transforms can be implemented to provide greater uniformity of energy data dissemination on part of utility operators. Wavelet transforms can thus be exploited in the energy industry as an advanced data analysis tool not limited to machine learning forecasting application.

References

- [1] Meyer, Yves (1992). *Wavelets and Operators*. Cambridge: Cambridge University Press. ISBN 0-521-42000-8.
- [2] Chui, Charles K. (1992). *An Introduction to Wavelets*. San Diego: Academic Press. ISBN 0-12-174584-8.

- [3] Akansu, Ali N.; Haddad, Richard A. (1992). *Multiresolution Signal Decomposition: Transforms, Subbands, Wavelets*. San Diego: Academic Press. ISBN 978-0-12-047141-6.
- [4] T.H. Le, D.A. Nguyen, “Orthogonal-based wavelet analysis of wind turbulence and correlation between turbulence and forces”, *Journal of Mechanics*, VAST, vol. 29(2), pp. 73-82, 2007.
- [5] A. Woyte, R. Belmans, and J. Nijs. Fluctuations in instantaneous clearness index: Analysis and statistics. *Solar Energy*, 81(2):195–206, February 2007.
- [6] Margarete Oliveira Domingues, Odim Mendes Jr., and Aracy Mendes da Costa. On wavelet techniques in atmospheric sciences. *Advances in Space Research*, 35(5):831–842, 2005.
- [7] Katie Coughlin, Aditya Murthi and Joseph Eto, Multi-scale Analysis of Wind Power and Load Time Series Data, *Renewable Energy* 68, no. August 2014 (2014): 494-504.
- [8] K. Coughlin and J. Eto. Analysis of wind power and load data at multiple time scales. Technical Report LBNL-4147E, Lawrence Berkeley National Laboratory, December 2010.
- [9] H. Holttinen. Hourly wind power variations in the nordic countries. *Wind Energy*, 8(2):173–195, April 2005.
- [10] L. Soder, H. Abildgaard, A. Estanqueiro, C. Hamon, H. Holttinen, E. Lannoye, E. Gomez-Lazaro, M. O’Malley, and U. Zimmermann. Experience and challenges with short-term balancing in european systems with large share of wind power. *IEEE Transactions on Sustainable Energy*, 3(4):853 –861, October 2012.
- [11] Y. Wan. Wind power plant behaviors: Analyses of Long-Term wind power data. Technical Report NREL/TP-500-36551, National Renewable Energy Laboratory, September 2004.
- [12] Y. Wan. Primer on wind power for utility applications. Technical Report NREL/TP-500-36230, National Renewable Energy Laboratory, December 2005.
- [13] Y. Wan and J. R Liao. Analyses of wind energy impact on WFEC system operations: Preprint. Technical Report NREL/JA-500-39583, National Renewable Energy Laboratory, March 2006.
- [14] GE Energy. Western wind and solar integration study. Technical report, National Renewable Energy Laboratory, May 2010.
- [15] Enernex. Minnesota statewide wind integration study. Technical report, November 2006.
- [16] M. Milligan, D. Lew, D. Corbus, R. Piwko, N. Miller, K. Clark, G. Jordan, L. Freeman, B. Zavadil, and M. Schuerger. Large-scale wind integration studies in the United States: Preliminary results; preprint. Technical Report NREL/CP-550-46527, NREL, September 2009.)

- [17] Rana, Koprinska et al., Forecasting electricity load with advanced wavelet neural networks, *Neurocomputing*, Volume 182, 19 March 2016, Pages 118–132
- [18] Benoit B. Mandelbrot and John W. Van Ness. Fractional brownian motions, fractional noises and applications. *SIAM Review*, 10(4):422–437, October 1968.
- [19] W. Willinger, M. S. Taqqu, W. E. Leland, and D. V. Wilson. Self-Similarity in High-Speed packet traffic: Analysis and modeling of ethernet traffic measurements. *Statistical Science*, 10(1):67–85, February 1995.
- [20] P. Abry. Scaling and wavelets: An introductory walk. In Govindan Rangarajan and Mingzhou Ding, editors, *Processes with Long-Range Correlations*, volume 621 of *Lecture Notes in Physics*, pages 34–60. Springer Berlin / Heidelberg, 2003.
10.1007/3-540-44832-2_3.
- [21] I. Daubechies. Ten lectures on wavelets. SIAM, June 1992.
- [22] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.
- [23] J. Pando and L. Fang. Discrete wavelet transform power spectrum estimator. *Physical Review E*, 57(3):3593, March 1998.
- [24] William H. Press, Saul A. Teukolsky, Brian P. Flannery, and William T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, 1992.
- [25] <http://www.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html>, Mathworks, Neural Network Training Function. Accessed: 05/08/2016
- [26] Electric Load Forecasting Using An Artificial Neural Network, D.C. Park, M.A. El-Sharkawi, R.J. Marks 11, L.E. Atlas and M.J. Damborg, *IEEE Transactions on Power Systems*, Vol.1.6, No. 2, May 1991.
- [27] Neural Networks for Short-Term Load Forecasting: A Review and Evaluation, Henrique Steinherz Hippert, Carlos Eduardo Pedreira, and Reinaldo Castro Souza, *IEEE Transactions on Power Systems* Vol. 16, No. 1, February 2001.
- [28] R. Lamedica, A. Prudenzi, M. Sforza, M. Caciotta, and V. O. Cencelli, “A neural network based technique for short-term forecasting of anomalous load periods,” *IEEE Trans. Power Systems*, vol. 11, no. 4, pp. 1749–1756, 1996.

- [29] G. Papaefthymiou and B. Klöckl, “MCMC for wind power simulation,” *IEEE Trans. Energy Convers.*, vol. 23, no. 1, pp. 234–240, Mar. 2008.
- [30] R. Billinton, H. Chen, and R. Ghajar, “Time-series models for reliability evaluation of power systems including wind energy,” *Microelectron Reliab.*, vol. 36, pp. 1253–1261, 1996.
- [31] B. G. Brown, R. W. Katz, and A. H. Murphy, “Time series models to simulate and forecast wind speed and wind power,” *J. Climate Appl. Meteorol.*, vol. 23, pp. 1184–1195, May 1984.
- [32] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes.*, 4th ed. New York: McGraw-Hill, 2002.
- [33] W. W. S. Wei, *Time Series Analysis: Univariate and Multivariate Methods*. Redwood City, CA: Addison-Wesley, 1990.
- [34] <https://onlinecourses.science.psu.edu/stat510/node/67>. Penn State University. Accessed: 05/08/2016
- [35] <https://www.nwcouncil.org/history/BPAElectricity>. Bonneville Power Association. Accessed: 05/08/2016.
- [36] <http://www.forbes.com/sites/jamesconca/2014/01/18/wind-energy-of-no-use-in-the-pacific-northwest/2/#30d5a76c345b>. Forbes. Accessed: 05/08/2016.

Appendix

Nan_correction

```
load('main project.mat')
load_07(isnan(load_07))==0;
load_08(isnan(load_08))==0;
load_09(isnan(load_09))==0;
load_10(isnan(load_10))==0;
load_11(isnan(load_11))==0;
load_12(isnan(load_12))==0;
load_13(isnan(load_13))==0;
load_14(isnan(load_14))==0;
load_15(isnan(load_15))==0;
wind_07(isnan(wind_07))==100000;
wind_08(isnan(wind_08))==100000;
wind_09(isnan(wind_09))==100000;
wind_10(isnan(wind_10))==100000;
wind_11(isnan(wind_11))==100000;
wind_12(isnan(wind_12))==100000;
wind_13(isnan(wind_13))==100000;
wind_14(isnan(wind_14))==100000;
wind_15(isnan(wind_15))==100000;
for i=1:length(load_07)
if load_07(i) == 0
    load_07(i)= ( load_07(i-1) + load_07 (i+1) )/2;
end
end
```

```

for i=1:length(load_08)
if load_08(i) == 0
load_08(i)= ( load_08(i-1) + load_08 (i+1) )/2;
end
end
for i=1:length(load_09)
if load_09(i) == 0
load_09(i)= ( load_09(i-1) + load_09 (i+1) )/2;
end
end
for i=1:length(load_10)
if load_10(i) == 0
load_10(i)= ( load_10(i-1) + load_10 (i+1) )/2;
end
end
for i=1:length(load_11)
if load_11(i) == 0
load_11(i)= ( load_11(i-1) + load_11 (i+1) )/2;
end
end
for i=1:length(load_12)
if load_12(i) == 0
load_12(i)= ( load_12(i-1) + load_12 (i+1) )/2;
end
end
for i=1:length(load_13)
if load_13(i) == 0
load_13(i)= ( load_13(i-1) + load_13 (i+1) )/2;
end
end
for i=1:length(load_14)
if load_14(i) == 0
load_14(i)= ( load_14(i-1) + load_14 (i+1) )/2;
end
end
for i=1:length(load_15)
if load_15(i) == 0
load_15(i)= ( load_15(i-1) + load_15 (i+1) )/2;
end
end
for i=1:length(wind_07)
if wind_07(i) == 100000
wind_07(i)= ( wind_07(i-1) + wind_07 (i+1) )/2;
end
end
for i=1:length(wind_08)
if wind_08(i) == 100000
wind_08(i)= ( wind_08(i-1) + wind_08 (i+1) )/2;
end
end
for i=1:length(wind_09)
if wind_09(i) == 0
wind_09(i)= ( wind_09(i-1) + wind_09 (i+1) )/2;
end
end
for i=1:length(wind_10)
if wind_10(i) == 100000

```

```

        wind_10(i)= ( wind_10(i-1) + wind_10 (i+1) )/2;
end
end
for i=1:length(wind_11)
if wind_11(i) == 100000
    wind_11(i)= ( wind_11(i-1) + wind_11 (i+1) )/2;
end
end
for i=1:length(wind_12)
if wind_12(i) == 100000
    wind_12(i)= ( wind_12(i-1) + wind_12 (i+1) )/2;
end
end
for i=1:length(wind_13)
if wind_13(i) == 100000
    wind_13(i)= ( wind_13(i-1) + wind_13 (i+1) )/2;
end
end
for i=1:length(wind_14)
if wind_14(i) == 100000
    wind_14(i)= ( wind_14(i-1) + wind_14 (i+1) )/2;
end
end
for i=1:length(wind_15)
if wind_15(i) == 100000
    wind_15(i)= ( wind_15(i-1) + wind_15 (i+1) )/2;
end
end
end

```

To evaluate performance of Discrete Wavelet Transforms against other Smoothing and Polynomial Fitting Algorithms through statistical metrics

```

clc;
clear all;
close all;
load('main project.mat')
Nan_correction
count = 1;
arranged_load(:, :)=0;
yr=07
if yr==07
input_doy = doym_07;
input_load = load_07;
elseif yr==08
input_doy = doym_08;
input_load = load_08;
elseif yr==09
input_doy = doym_09;
input_load = load_09;
elseif yr==10
input_doy = doym_10;
input_load = load_10;
elseif yr==11

```

```

input_doy = doym_11;
input_load = load_11;
elseif yr==12
input_doy = doym_12;
input_load = load_12;
elseif yr==13
input_doy = doym_13;
input_load = load_13;
elseif yr==14
input_doy = doym_14;
input_load = load_14;
elseif yr==15
input_doy = doym_15;
input_load = load_15;
end
for i=1:length(input_doy)

    arranged_load(input_doy(i),count) = input_load(i);

    count = count +1;

    if((i~=length(input_doy)) && (input_doy(i) ~= input_doy(i+1)))

        count = 1;

    end
end

[m,n]=size(arranged_load);
day=input('Enter day');
lvl_d=input('Enter level for day');
wavename='db4';

%day
for day =1:m
[C_day,L_day]=wavedec(arranged_load(day,:),lvl_d,wavename);% C -
Decomposition vector, L - Book keeping vector
A_day=appcoef(C_day,L_day,wavename,lvl_d);
C_day_rec=A_day;
for i=lvl_d:-1:1
    D_day=detcoef(C_day,L_day,i);
    D_day(:)=0;
    C_day_rec=[C_day_rec,D_day];
end
load_rec(day,:)=waverec(C_day_rec,L_day,wavename);
end
figure
x=1 : length(load_rec(day,:));
plot(x,arranged_load(day,:), 'r',x,load_rec(day,:))
title('Original data vs wavelet recreated data (of specified day)')
legend('Original data', 'Wavelet Recreated Data')
%mean percentage error
mpe=(arranged_load(day,:)-load_rec(day,:))./arranged_load(day,:);

%Smoothing

```

```

smoothed_load1=[];
smoothed_load2=[];
smoothed_load3=[];
smoothed_load4=[];
for d=1:m
    %j-1:288,i-1:365
    smoothed_load1(d,:)=smooth(arranged_load(d,:), 'moving');
    smoothed_load2(d,:)=smooth(arranged_load(d,:), 'lowess');
    smoothed_load3(d,:)=smooth(arranged_load(d,:), 'loess');
    smoothed_load4(d,:)=smooth(arranged_load(d,:), 'sgolay');
end
figure
plot(x, arranged_load(day,:), 'r', x, load_rec(day,:), 'k', x, smoothed_load1(day,:),
'-', x, smoothed_load2(day,:), '-', x, smoothed_load3(day,:), '-
', x, smoothed_load4(day,:), '-');
title('arranged vs actual load for day compared with wavelet recreation and
smoothing algorithms on original data (of specified day)')
legend('original data', 'wavelet recreated', 'curve smoothing(moving)', 'curve
smoothing(lowess)', 'curve smoothing(loess)', 'curve smoothing(sgolay)')
xlabel('Time')
%Fit algorithms

for ii = 1:m
    fitVar1{ii} = fit(x', arranged_load(ii,:), 'poly9'); %Ninth degree
polynomial fitting
    fitVar2{ii} = fit(x', arranged_load(ii,:), 'poly8'); %Eighth degree
polynomial fitting
end

iii = 1;
while(iii<m+1)
    fit1(iii,:)=fitVar1{iii}(x);
    fit2(iii,:)=fitVar2{iii}(x);
    iii = iii+1;
end

figure
plot(fit1(day,:))
hold on
plot(fit2(day,:))
hold on
plot(x, arranged_load(day,:), '-', x, load_rec(day,:), '-k')
hold off
title('Original data vs wavelet recreated vs fitting curves at a particular
day(of specified day)')
legend('Ninth degree polynomial', 'Eighth degree fourier', 'Original
data', 'wavelet recreation')
xlabel('Time')

%RMSE of curves(||||| column wise mean, Mean of all days)
rmse1=rms(arranged_load-load_rec);
rmse2=rms(arranged_load-smoothed_load1);
rmse3=rms(arranged_load-smoothed_load2);
rmse4=rms(arranged_load-smoothed_load3);
rmse5=rms(arranged_load-smoothed_load4);
rmse6=rms(arranged_load-fit1);

```

```

rmse7=rms(arranged_load-fit2);
figure
plot(x,rmse1,'k',x,rmse2,x,rmse3,x,rmse4,x,rmse5,x,rmse6,x,rmse7);
title('Root mean square error for wavelet recreation, fitting and smoothing
algorithms on original data(all days on every particular time)')
legend('wavelet recreated','curve smoothing(moving)','curve
smoothing(lowess)','curve smoothing(loess)','curve
smoothing(sgolay)','fitting (ninth degree poly)','fitting(Eighth degree
fourier)')
xlabel('Time')
%Standard deviation(||||||| column wise mean, Mean of all days)
std1=std(arranged_load-load_rec,0,1);
std2=std(arranged_load-smoothed_load1,0,1);
std3=std(arranged_load-smoothed_load2,0,1);
std4=std(arranged_load-smoothed_load3,0,1);
std5=std(arranged_load-smoothed_load4,0,1);
std6=std(arranged_load-fit1,0,1);
std7=std(arranged_load-fit2,0,1);
figure
plot(x,std1,'k',x,std2,x,std3,x,std4,x,std5,x,std6,x,std7)
title('Standard Deviation error for recreated, fitting and smoothing
curves(all days on every particular time)')
legend('wavelet recreated','curve smoothing(moving)','curve
smoothing(lowess)','curve smoothing(loess)','curve
smoothing(sgolay)','fitting(ninth degree polynomial)','fitting(Eighth degree
fourier)')
xlabel('Time')
%Gradient test (||||||| column wise mean, Mean of all days)
[FX1]=mean-gradient(arranged_load),1);
[FX2]=mean-gradient(load_rec),1);
[FX3]=mean-gradient(smoothed_load1),1);
[FX4]=mean-gradient(smoothed_load2),1);
[FX5]=mean-gradient(smoothed_load3),1);
[FX6]=mean-gradient(smoothed_load4),1);
[FX7]=mean-gradient(fit1),1);
[FX8]=mean-gradient(fit2),1);
figure
plot(x,FX1,'r',x,FX2,'k',x,FX3,x,FX4,x,FX5,x,FX6,x,FX7,x,FX8)
title('Mean gradients of curves(all days on every particular time)')
legend('Original data','Wavelet Recreated','curve smoothing(moving)','curve
smoothing(lowess)','curve smoothing(loess)','curve
smoothing(sgolay)','fitting(ninth degree polynomial)','fitting(Eighth degree
fourier)')
xlabel('Time')
figure
surf(load_rec)
title('Recreated load')

```

wind_all_wavonly

```

yr = 07; %Change to 10 and 13 for further use
load_all;
[m,n]=size(arranged_load);

wavename='db4';
for lvl=1:8

```

```

%day
for day =1:m
[C_day,L_day]=wavedec(arranged_load(day,:),lvl,wavename);
A_day=appcoef(C_day,L_day,wavename,lvl);
C_day_rec=A_day;
for i=lvl:-1:1
    D_day=detcoef(C_day,L_day,i);
    D_day(:)=0;
    C_day_rec=[C_day_rec,D_day];
end
load_rec(day,:)=waverec(C_day_rec,L_day,wavename);
end

%Convert recreated load to one day (whole year) profile to be used in ANN
%where columns represent the decomposition levels (105120*8 matrix will be
formed)

wav_out_07(lvl,:) = reshape(load_rec',1,numel(load_rec));

end

```

Time Resolution Manipulation

```

%Load the data set for wavelet analysis on wind
wind_all_wavonly;
%Convert the one dimensional time series to two dimensional data (Rows =
%days, Columns = Hour of day)
for i=1:length(input_doy)

    arranged_wind(input_doy(i),count) = input_wind(i);

    count = count +1;

    if((i~=length(input_doy)) && (input_doy(i) ~= input_doy(i+1)))

        count = 1;

    end
end

%Input the day to analyze and the decomposition level to analyze
[m,n]=size(arranged_wind);
day=input('Enter day');
lvl_d=input('Enter level for day');
wavename='db4';

%Analyzing daily profiles
% Zero padding wavelet
for day =1:m
[C_day,L_day]=wavedec(arranged_wind(day,:),lvl_d,wavename); %Decompose the
daily data using wavedec at specified level
A_day=appcoef(C_day,L_day,wavename,lvl_d); %Extract approximation coefficient
from the decomposition vector

```

```

C_day_rec=A_day; % Create a decomposition vector equal to the approximation
coefficient
for i=lvl_d:-1:1 %Iteratively set the detail coefficients at corresponding
level equal to and lower than the user specified level
    D_day=detcoef(C_day,L_day,i); %Extract detail coefficient from the new
decomposition vector and using the same bookkeeping vector (L)
    D_day(:)=0; %Set detail coefficient to zero
    C_day_rec=[C_day_rec,D_day]; %Recreate new decomposition vector with zero
detail components till the specified level
end
wind_rec(day,:)=waverec(C_day_rec,L_day,wavename); % Recreate the whole time
series using the new decomposition vector and the original bookkeeping vector
end
figure
x=1 : length(wind_rec(day,:));
plot(x,arranged_wind(day,:), 'r',x,wind_rec(day,:))
title('Original data vs wavelet recreated data (of specified day)')
legend('Original data', 'Wavelet Recreated Data')

%Zero padding new wavelet
wavename='db2';
xxx=1; %Define a level for decomposition
[C_n,S_new]=wavedec(arranged_wind(day,:),xxx,wavename); %for level one, can
be changed
a_new=appcoef(C_n,S_new,wavename); %Approximation coefficient
d_new=detcoef(C_n,S_new,wavename); % Detail coefficient
a=arranged_wind(day,:);
approx=a(1:2:length(a));%odd values of the time series are used as
approximation coefficients (Length = 1/2 of original data length)
det=zeros(1,length(C_n)-length(approx)); %Adjust length of detail coefficient
vector and set it to zero
C_new=[approx det]; %Create new decomposition vector using the approximation
and the detail ceofficeints
wav_new=waverec(C_new,S_new,wavename); % Recreate the wavelet series using
the new
interpolation=interp(approx,2); %Interpolate the approximation to double the
data length
figure
%Plot normalized data
plot(x, (wav_new-min(wav_new))/(max(wav_new)-min(wav_new)),x, (a-
min(a))/(max(a)-min(a)),x, (interpolation-
min(interpolation))/(max(interpolation)-min(interpolation)))
title('Approx coef = odd values for user specified day')
legend('Recreated','Original data','interpolation')
factor=mean(a-wav_new); % A factor needs to added to shift the entire series
figure
plot(x,wav_new+factor,x,a,x,interpolation)
legend('new wavelet','original data','interpolation')

```

Artificial Neural Network - Load

```

clc
clear all
diary('ANN_load'); diary on;
%% Load data

```

```

yr = 07; %Base year for input_load
load_all; %Load all data

%% Making the ANN model
% Yearly forecasting
%Inputs and target data need to be row vector always
x=input_load'; %x is the input that the model is trained on, year 2007 data
t=load_08'; %t is the target data that the ANN uses to set weights and
initialize the network

%Normalize all data to be used in the network
x=x/max(x);
t = t / max(t);
net = feedforwardnet(40); %Feed forward network with 40 neurons is designed

%Divide the data set in 3 equal parts for training/validation/testing
net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3; net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

%Maximum allowable validation failure = 6
net.trainParam.max_fail = 6;
% Transfer function between layers = logsig to limit output in the [0,1]
% range
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t); % Train the network based on inout and target data
view(net)
y = net(x);
mape = mean((abs(y-t))./t); %Mean of errors between forecasted values and
target data values
grad_1 = mean(gradient(y)); %The mean of gradients at all points on the
forecasted value curve

%% Year ahead predictions (2009-10,11-12,13-14)
l1 = load_09'/max(load_09);
y1 = net(l1);
grad_2 = mean(gradient(y1));
l11 = load_10'/max(load_10);
mape1 = mean((abs(y1-l11))./l11);
l2 = load_11'/max(load_11);
y2 = net(l2);
grad_3 = mean(gradient(y2));
l22 = load_12'/max(load_12);
mape2 = mean((abs(y2-l22))./l22);
l2 = load_13'/max(load_13);
y2 = net(l2);
grad_4 = mean(gradient(y2));
l22 = load_14'/max(load_14);
mape3 = mean((abs(y2-l22))./l22);

mape_yearly = [mape, mape1, mape2, mape3]; %Concatenate MAPEs to find the
range of MAPEs and mean of all MAPEs

```

```

m1 = max(mape_yearly); n1 = min(mape_yearly); mmape1 = mean(mape_yearly);
fprintf('Range of mapes %d to %d and the mean of MAPEs of yearly forecasting
is %d\n',n1,m1,mmape1);
grad_y=mean([grad_1 grad_2 grad_3 grad_4]); %Mean of all gradients
var_grad_y = var([grad_1 grad_2 grad_3 grad_4]);
fprintf('Yearly mean gradient is %d and variance of gradients is
%d\n',grad_y,var_grad_y);

%% Change timescales to short-term and medium term forecasting -
%% Error for 6 month ahead prediction (2007,08,09,10,11,12,13)

x=input_load(1:end/2)'; %First half of year
t=input_load(end/2 + 1 : end)'; % Second half of year

x=x/max(x);
t = t / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3; net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);
mape = mean((abs(y-t))./t);

%% Error for half-year ahead prediction
%Year 2008
l1 = load_08(1:end/2)'/max(load_08(1:end/2));
y1 = net(l1);
grad_1 = mean(gradient(y1));
l10 = load_08(end/2 + 1 : end)'/max(load_08(end/2 + 1 : end));
mape1 = mean((abs(y1-l10))./l10);
%Year 2009
l1 = load_09(1:end/2)'/max(load_09(1:end/2));
y1 = net(l1);
grad_2 = mean(gradient(y1));
l10 = load_09(end/2 + 1 : end)'/max(load_09(end/2 + 1 : end));
mape2 = mean((abs(y1-l10))./l10);
%Year 2010
l1 = load_10(1:end/2)'/max(load_10(1:end/2));
y1 = net(l1);
grad_3 = mean(gradient(y1));
l10 = load_10(end/2 + 1 : end)'/max(load_10(end/2 + 1 : end));
mape3 = mean((abs(y1-l10))./l10);
%Year 2011
l1 = load_11(1:end/2)'/max(load_11(1:end/2));
y1 = net(l1);
grad_4 = mean(gradient(y1));

```

```

l10 = load_11(end/2 + 1 : end)'/max(load_11(end/2 + 1 : end));
mape4 = mean((abs(y1-l10))./l10);
%Year 2012
l1 = load_12(1:end/2)'/max(load_12(1:end/2));
y1 = net(l1);
grad_5 = mean(gradient(y1));
l10 = load_12(end/2 + 1 : end)'/max(load_12(end/2 + 1 : end));
mape5 = mean((abs(y1-l10))./l10);
% Year 2013
l1 = load_13(1:end/2)'/max(load_13(1:end/2));
y1 = net(l1);
grad_6 = mean(gradient(y1));
l10 = load_13(end/2 + 1 : end)'/max(load_13(end/2 + 1 : end));
mape6 = mean((abs(y1-l10))./l10);
%Year 2014
l1 = load_14(1:end/2)'/max(load_14(1:end/2));
y1 = net(l1);
grad_7 = mean(gradient(y1));
l10 = load_14(end/2 + 1 : end)'/max(load_14(end/2 + 1 : end));
mape7 = mean((abs(y1-l10))./l10);

mape_halfyear = [mape,mape1,mape2,mape3,mape4,mape5,mape6,mape7];
m1 = max(mape_halfyear); n1 = min(mape_halfyear); mmape1 =
mean(mape_halfyear); fprintf('Range of mapes %d to %d and the mean of MAPEs
of sixth month forecasting is %d\n',n1,m1,mmape1);

grad_hy=mean([grad_1 grad_2 grad_3 grad_4 grad_5 grad_6 grad_7]);
var_grad_hy = var([grad_1 grad_2 grad_3 grad_4 grad_5 grad_6 grad_7]);
fprintf('Half Yearly mean gradient is %d and variance of gradients is
%d\n',grad_hy, var_grad_hy);
%% Extract two day pairs for consideration
%%There are 24*(60/5) samples in a day. 288 samples make a day. (i*288)+1 :
%((i+1)*288) samples define a day
x=input_load(1 : 288)'; % First day of year
t=input_load(288+1 : 2*288)'; % Second day of year

x=x/max(x);
t = t / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3; net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);
% Iteratively running the analysis on all days
for i = 0:1:363
l1 = input_load((i*288)+1 : ((i+1)*288))' / max(input_load((i*288)+1 :
((i+1)*288)));
y1 = net(l1);

```

```

l10 = input_load((i+1)*288+1 : (i+2)*288)'/max(input_load((i+1)*288+1 :
(i+2)*288));
mape_day(i+1) = mean((abs(y1-l10))./l10);
%A value of the wavelet recreation is negative at i = 86 which is an
%isolated incident of a negative value being produced by wavelet
%transformation. Thus it is eliminated.
if mape_day(i+1) < 0
    mape_day(i+1) = [];
end
grad(i+1) = mean(gradient(y1));
end
m1 = max(mape_day); n1 = min(mape_day); mmape1 = mean(mape_day);
fprintf('Range of mapes %d to %d and the mean of MAPEs of day ahead
forecasting is %d\n',n1,m1,mmape1);
grad_day = mean(grad);
var_grad_d = var(grad);
fprintf('Daily mean gradient is %d and variance of gradients is
%d\n',grad_day,var_grad_d);
%% Extract two month pairs for consideration
% Assuming each month to be 30 days long and ignoring all the left over
% values after 12*30 days i.e. 12 months are considered with 30 days each
% Trained on January - February pair
%((i*288) + 1) : ((i+30)*288) define a month

x=input_load(1 : 30*288)';
t=input_load(30*288 + 1 : (30+30)*288)';

x=x/max(x);
t = t / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3;    net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);
plot(y)
% Iteratively running the analysis on all months
for i = 0:30:30*10
l1 = input_load(((i*288) + 1) : ((i+30)*288))'/max(input_load((i*288) + 1) :
((i+30)*288));
y1 = net(l1);

l10 = input_load(((i+30)*288) + 1 :
(i+30+30)*288)'/max(input_load(((i+30)*288) + 1 : (i+30+30)*288));
mape_month(i+1) = mean((abs(y1-l10))./l10);
grad(i+1) = mean(gradient(y1));
end

```

```

m1 = max(mape_month); n1 = min(mape_month); mmape1 = mean(mape_month);
fprintf('Range of mapes %d to %d and the mean of MAPEs of month ahead
forecasting is %d\n',n1,m1,mmape1);
grad_m = mean(grad);
var_grad_m = var(grad);
fprintf('Monthly mean gradient is %d and variance of gradients is
%d\n',grad_m,var_grad_m);
%% Hour ahead prediction ; January -> Training: 12 a.m. to 1 a.m. and 1 to 2
a.m.
x=input_load(1:12)';
t=input_load(12 + 1 : 12+12)';

x=x/max(x);
t = t / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3; net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);

% Iteratively running the analysis on all hours of the first day
for i = 0:1:22
l1 = input_load((i*12) + 1 : (i+1)*12)'/max(input_load((i*12) + 1 :
(i+1)*12));
y1 = net(l1);
l10 = input_load(((i+1)*12) + 1 : (i+2)*12)'/max(input_load(((i+1)*12) + 1 :
(i+2)*12));
mape_hour(i+1) = mean((abs(y1-l10))./l10);
grad(i+1) = mean(gradient(y1));
end
m1 = max(mape_hour); n1 = min(mape_hour); mmape1 = mean(mape_hour);
fprintf('Range of mapes %d to %d and the mean of MAPEs of hour ahead
forecasting is %d\n',n1,m1,mmape1);
grad_h = mean(grad);
var_grad_h = var(grad);
fprintf('Hourly mean gradient is %d and variance of gradients is
%d\n',grad_h,var_grad_h);
diary off

```

Artificial Neural Network – Wind

```

clc
clear all
diary('ANN_wind'); diary on;
%% wind data
yr = 07; %Base year for input_wind
wind_all; %Load all data

```

```

%% Extract two day pairs for consideration
%There are 24*(60/5) samples in a day. 288 samples make a day. (i*288)+1 :
%((i+1)*288) samples define a day
x=input_wind(1 : 288)';% First day of year
t=input_wind(288+1 : 2*288)'; % Second day of year

x=x/max(x);
t = t / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3; net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);

% Iteratively running the analysis on all days
for i = 0:1:360
l1 = input_wind((i*288)+1 : ((i+1)*288))' / max(input_wind((i*288)+1 :
((i+1)*288)));
y1 = net(l1);
grad(i+1) = mean(gradient(y1));
l10 = input_wind((i+1)*288+1 : (i+2)*288)'/max(input_wind((i+1)*288+1 :
(i+2)*288));
mape_day(i+1) = mean((abs(y1-l10))./l10);
if mape_day(i+1) == Inf %Remove all infinity error values
    mape_day(i+1) = [];
elseif isnan(mape_day(i+1)) == 1 %Remove all errors values that correspond to
100% error as error ranges lie in the 0-1% range, 100% error can throw off
the mean error calculations
    mape_day(i+1) = [];
end
end
m1 = max(mape_day); n1 = min(mape_day); mmape1 = mean(mape_day);
fprintf('Range of mapes %d to %d and the mean of MAPEs of day ahead
forecasting is %d\n',n1,m1,mmape1);
grad_day = mean(grad);
var_grad_d = var(grad);
fprintf('Daily mean gradient is %d and variance of gradients is
%d\n',grad_day,var_grad_d);
%% Hour ahead predictions ; January -> Training: 12 a.m. to 1 a.m. and 1 to 2
a.m. ; 8-9/9-10 am; 7-8/8-9 pm
x=input_wind(1:12)';
t=input_wind(12 + 1 : 12+12)';

x=x/max(x);
t = t / max(t);
net = feedforwardnet(40);

```

```

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3;    net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);

% Iteratively running the analysis on all hours of the first day
for i = 0:1:22
l1 = input_wind((i*12) + 1 : (i+1)*12)'/max(input_wind((i*12) + 1 :
(i+1)*12));
y1 = net(l1);
grad(i+1) = mean(gradient(y1));
l10 = input_wind(((i+1)*12) + 1 : (i+2)*12)'/max(input_wind(((i+1)*12) + 1 :
(i+2)*12));
mape_hour(i+1) = mean((abs(y1-l10))./l10);
if mape_hour(i+1) == Inf
    mape_hour(i+1) = [];
elseif isnan(mape_hour(i+1)) == 1
    mape_hour(i+1) = [];
end
end
m1 = max(mape_hour); n1 = min(mape_hour); mmape1 = mean(mape_hour);
fprintf('Range of mapes %d to %d and the mean of MAPEs of hour ahead
forecasting is %d\n',n1,m1,mmape1);
grad_h = mean(grad);
var_grad_h = var(grad);
fprintf('Hourly mean gradient is %d and variance of gradients is
%d\n',grad_h, var_grad_h);
diary off

```

wav_wind_07 (applies wavelet algorithm to year 2007)

```

yr = 07; %Change to 10 and 13 for further use
wind_all_wavonly;
[m,n]=size(arranged_wind);

wavename='db4';
for lvl=1:8
%day
for day =1:m
[C_day,L_day]=wavedec(arranged_wind(day,:),lvl,wavename);
A_day=appcoef(C_day,L_day,wavename,lvl);
C_day_rec=A_day;
for i=lvl:-1:1
    D_day=detcoef(C_day,L_day,i);
    D_day(:)=0;
    C_day_rec=[C_day_rec,D_day];
end

```

```
wind_rec(day,:)=waverec(C_day_rec,L_day,wavename);
end
```

```
%Convert recreated load to one day (whole year) profile to be used in ANN
%where columns represent the decomposition levels (105120*8 matrix will be formed)
```

```
wav_out_07(lvl,:)= reshape(wind_rec',1,numel(wind_rec));
```

```
end
```

WANN model – wind

```
clc
clear all
diary('WANN_wind'); diary on;
%Base year for input_wind = 2007

%error in wavelet recreations
wav_wind_07;

for lvl = 1:8
input_wind = wav_out_07(lvl,:);
%% Extract two day pairs for consideration
%Works with 20 - 25% error!
%January 20-21 ; May 20-21; October 20-21
%There are 24*(60/5) samples in a day. 288 samples make a day. (19*288+1 :
%20*288) - Jan 20
x=input_wind(1 : 288)';
t=input_wind(288+1 : 2*288)';

x=x'/max(x);
t = t' / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3; net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);

for i = 0:1:360
l1 = input_wind((i*288)+1 : ((i+1)*288)) / max(input_wind((i*288)+1 :
((i+1)*288)));
y1 = abs(net(l1));
grad(i+1) = mean(gradient(y1));
l10 = abs(input_wind((i+1)*288+1 : (i+2)*288)/max(input_wind((i+1)*288+1 :
(i+2)*288)));
mape_day(i+1) = mean((abs(y1-l10))./l10);
```

```

if mape_day(i+1) == Inf
    mape_day(i+1) = [];
elseif isnan(mape_day(i+1)) == 1
    mape_day(i+1) = [];
end
end
m1 = max(mape_day); n1 = min(mape_day); mmape1 = mean(mape_day);
fprintf('Range of mapes %d to %d and the mean of MAPEs of day ahead
forecasting is %d at decomposition level %d\n',n1,m1,mmape1,lvl);
grad_day = mean(grad);
grad_var_d = var(grad);
fprintf('Daily mean gradient is %d and variance of gradients is
%d\n',grad_day, grad_var_d);
%% Error for hour ahead prediction ; January -> Training: 12 a.m. to 1 a.m.
and 1 to 2 a.m. ; 8-9/9-10 am; 7-8/8-9 pm
% Works!
x=input_wind(1:12)';
t=input_wind(12 + 1 : 12+12)';

x = x' / max(x);
t = t' / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3;    net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);

for i = 0:1:22
l1 = input_wind((i*12) + 1 : (i+1)*12)/max(input_wind((i*12) + 1 :
(i+1)*12));
y1 = abs(net(l1));
l10 = abs(input_wind(((i+1)*12) + 1 : (i+2)*12)/max(input_wind(((i+1)*12) + 1
: (i+2)*12)));
mape_hour(i+1) = mean((abs(y1-l10))./l10);
if mape_hour(i+1) == Inf
    mape_hour(i+1) = [];
elseif isnan(mape_hour(i+1)) == 1
    mape_hour(i+1) = [];
end
end
m1 = max(mape_hour); n1 = min(mape_hour); mmape1 = mean(mape_hour);
fprintf('Range of mapes %d to %d and the mean of MAPEs of hour ahead
forecasting is %d\n',n1,m1,mmape1);
grad_h = mean(grad);
grad_var_h = var(grad);
fprintf('Hourly mean gradient is %d and the variance of gradients is
%d\n',grad_h,grad_var_h);
end

```

```
diary off
```

WANN model - load

```
clc
clear all
diary('WANN_load'); diary on
%Base year for input_load = 2007

%error in wavelet recreations
wav_load_07;
wav_load_08;
wav_load_09;
wav_load_10;
wav_load_11;
wav_load_12;
wav_load_13;
wav_load_14;

for lvl = 1:4
    lvl

    input_load = wav_out_07(lvl,:);

    x=input_load;
    t=load_08';

    x=x/max(x);
    t = t / max(t);
    net = feedforwardnet(40);

    net.divideFcn = 'divideint';
    net.divideParam.trainRatio = 1/3;    net.divideParam.valRatio = 1/3;
    net.divideParam.testRatio = 1/3;

    net.trainParam.max_fail = 6;
    net.layers{1}.transferFcn = 'logsig';

    %% Training the model
    [net,tr] = train(net,x,t);
    view(net)
    y = net(x);
    grad_1 = mean(gradient(y));
    mape = mean((abs(y-t))./t);

    %% Error for year ahead prediction (2009-10,11-12,13-14)
    l1 = wav_out_09(lvl,:)/max(wav_out_09(lvl,:));
    y1 = net(l1);
    grad_2 = mean(gradient(y1));
    l11 = load_10'/max(load_10);
    mapel = mean((abs(y1-l11))./l11);    % Results in 14% error! Not bad for 5
    min time resolution and whole year data!
    l2 = wav_out_11(lvl,:)/max(wav_out_11(lvl,:));
    y2 = net(l2);
    grad_3 = mean(gradient(y2));
```

```

l22 = load_12'/max(load_12);
mape2 = mean((abs(y2-l22))./l22);    %10% error!
l2 = wav_out_13(lvl,+)/max(wav_out_13(lvl,+)/);
y2 = net(l2);
grad_4 = mean(gradient(y2));
l22 = load_14'/max(load_14);
mape3 = mean((abs(y2-l22))./l22);    %10% error!

mape_yearly = [mape, mape1, mape2, mape3];
m1 = max(mape_yearly); n1 = min(mape_yearly); mmape1 = mean(mape_yearly);
fprintf('Range of mapes %d to %d and the mean of MAPEs of yearly forecasting
is %d\n',n1,m1,mmape1);
grad_y=mean([grad_1 grad_2 grad_3 grad_4]);
grad_var_y = var([grad_1 grad_2 grad_3 grad_4]);
fprintf('Yearly mean gradient is %d and variance of gradients is
%d\n',grad_y,grad_var_y);
%% Change timescales to short-term and medium term forecasting -
% 105120
%% Error for 6 month ahead prediction (2010, 2013)
x=input_load(1:end/2)';
t=input_load(end/2 + 1 : end)';

x= x' / max(x);
t = t' / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3;    net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);

mape = mean((abs(y-t))./t);

%% Error for half-year ahead prediction
%Year 2008
wav_08 = wav_out_08(lvl,+)/;
wav_09 = wav_out_09(lvl,+)/;
wav_10 = wav_out_10(lvl,+)/;
wav_11 = wav_out_11(lvl,+)/;
wav_12 = wav_out_12(lvl,+)/;
wav_13 = wav_out_13(lvl,+)/;
wav_14 = wav_out_14(lvl,+)/;
l1 = wav_08(1:end/2)/max(wav_08(1:end/2));
y1 = net(l1);
grad_1 = mean(gradient(y1));
l10 = load_08(end/2 + 1 : end)'/max(load_08(end/2 + 1 : end));
mape1 = mean((abs(y1-l10))./l10);
%Year 2009

```

```

l1 = wav_09(1:end/2)/max(wav_09(1:end/2));
y1 = net(l1);
grad_2 = mean(gradient(y1));
l10 = load_09(end/2 + 1 : end)'/max(load_09(end/2 + 1 : end));
mape2 = mean((abs(y1-l10))./l10);
%Year 2010
l1 = wav_10(1:end/2)/max(wav_10(1:end/2));
y1 = net(l1);
grad_3 = mean(gradient(y1));
l10 = load_10(end/2 + 1 : end)'/max(load_10(end/2 + 1 : end));
mape3 = mean((abs(y1-l10))./l10);
%Year 2011
l1 = wav_11(1:end/2)/max(wav_11(1:end/2));
y1 = net(l1);
grad_4 = mean(gradient(y1));
l10 = load_11(end/2 + 1 : end)'/max(load_11(end/2 + 1 : end));
mape4 = mean((abs(y1-l10))./l10);
%Year 2012
l1 = wav_12(1:end/2)/max(wav_12(1:end/2));
y1 = net(l1);
grad_5 = mean(gradient(y1));
l10 = load_12(end/2 + 1 : end)'/max(load_12(end/2 + 1 : end));
mape5 = mean((abs(y1-l10))./l10);
% Year 2013
l2 = wav_13(1:end/2)/max(wav_13(1:end/2));
y2 = net(l2);
grad_6 = mean(gradient(y1));
l20 = load_13(end/2 + 1 : end)'/max(load_13(end/2 + 1 : end));
mape6 = mean((abs(y2-l20))./l20);
%Year 2014
l1 = wav_14(1:end/2)/max(wav_14(1:end/2));
y1 = net(l1);
grad_7 = mean(gradient(y1));
l10 = load_14(end/2 + 1 : end)'/max(load_14(end/2 + 1 : end));
mape7 = mean((abs(y1-l10))./l10);

mape_halfyear = [mape,mape1,mape2,mape3,mape4,mape5,mape6,mape7];
m1 = max(mape_halfyear); n1 = min(mape_halfyear); mmape1 =
mean(mape_halfyear); fprintf('Range of mapes %d to %d and the mean of MAPEs
of sixth month forecasting is %d\n',n1,m1,mmape1);

grad_hy=mean([grad_1 grad_2 grad_3 grad_4 grad_5 grad_6 grad_7]);
grad_var_hy = var([grad_1 grad_2 grad_3 grad_4 grad_5 grad_6 grad_7]);
fprintf('Half Yearly mean gradient is %d and the variance of gradients is
%d\n',grad_hy, grad_var_hy);
%% Extract two day pairs for consideration
%January 20-21 ; May 20-21; October 20-21
%There are 24*(60/5) samples in a day. 288 samples make a day. (19*288+1 :
%20*288) - Jan 20
x=input_load(1 : 288)';
t=input_load(288+1 : 2*288)';

x=x'/max(x);
t = t' / max(t);
net = feedforwardnet(40);

```

```

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3;    net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);

for i = 0:1:363
l1 = input_load((i*288)+1 : ((i+1)*288)) / max(input_load((i*288)+1 :
((i+1)*288)));
y1 = net(l1);
l10 = input_load((i+1)*288+1 : (i+2)*288)/max(input_load((i+1)*288+1 :
(i+2)*288));
mape_day(i+1) = mean((abs(y1-l10))./l10);
grad(i+1) = mean(gradient(y1));
end
m1 = max(mape_day); n1 = min(mape_day); mmape1 = mean(mape_day);
fprintf('Range of mapes %d to %d and the mean of MAPEs of day ahead
forecasting is %d\n',n1,m1,mmape1);
grad_day = mean(grad);
grad_var_d = var(grad);
fprintf('Daily mean gradient is %d and variance of gradients is
%d\n',grad_day,grad_var_d);
%% Extract two month pairs for consideration
% November - December (Winter) ; March - April(Summer) ; Training -> July -
August
x=input_load(1 : 30*288)';
t=input_load(30*288 + 1 : (30+30)*288)';

x=x'/max(x);
t = t' / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3;    net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);
for i = 0:30:30*10
l1 = input_load(((i*288) + 1) : ((i+30)*288))/max(input_load((i*288) + 1) :
((i+30)*288));
y1 = net(l1);
l10 = input_load(((i+30)*288) + 1 :
(i+30+30)*288)/max(input_load(((i+30)*288) + 1 : (i+30+30)*288));

```

```

mape_month(i+1) = mean((abs(y1-l10))./l10);
grad(i+1) = mean(gradient(y1));
end
m1 = max(mape_month); n1 = min(mape_month); mmape1 = mean(mape_month);
fprintf('Range of mapes %d to %d and the mean of MAPEs of month ahead
forecasting is %d\n',n1,m1,mmape1);
grad_m = mean(grad);
grad_var_m = var(grad);
fprintf('Monthly mean gradient is %d and variance of gradients is
%d\n',grad_m, grad_var_m);
%% Error for hour ahead prediction ; January -> Training: 12 a.m. to 1 a.m.
and 1 to 2 a.m. ; 8-9/9-10 am; 7-8/8-9 pm
x=input_load(1:12)';
t=input_load(12 + 1 : 12+12)';

x=x'/max(x);
t = t' / max(t);
net = feedforwardnet(40);

net.divideFcn = 'divideint';
net.divideParam.trainRatio = 1/3; net.divideParam.valRatio = 1/3;
net.divideParam.testRatio = 1/3;

net.trainParam.max_fail = 6;
net.layers{1}.transferFcn = 'logsig';

%% Training the model
[net,tr] = train(net,x,t);
view(net)
y = net(x);

for i = 0:1:22
l1 = input_load((i*12) + 1 : (i+1)*12)/max(input_load((i*12) + 1 :
(i+1)*12));
y1 = net(l1);
l10 = input_load(((i+1)*12) + 1 : (i+2)*12)/max(input_load(((i+1)*12) + 1 :
(i+2)*12));
mape_hour(i+1) = mean((abs(y1-l10))./l10);
grad(i+1) = mean(gradient(y1));
end
m1 = max(mape_hour); n1 = min(mape_hour); mmape1 = mean(mape_hour);
fprintf('Range of mapes %d to %d and the mean of MAPEs of hour ahead
forecasting is %d\n',n1,m1,mmape1);
grad_h = mean(grad);
grad_var_h = var(grad);
fprintf('Hourly mean gradient is %d and variance of gradients is
%d\n',grad_h,grad_var_h);
end

diary off

ARIMA – load
clc;
clear all;
close all;

```

```

yr=07
load_all;

%Try removing AR lags
%Check if the distribution is stationary or non-stationary. Since the ACF
%plot does not converge to zero, we conclude it is non stationary and needs
%differencing

%Subset input load to the load that needs to be analyzed
input_load = load_07(1:288*30);
figure()
plot(input_load)
title('Load series')
%The time series is differentiated to remove seasonality of data
diff_load = diff(input_load);
figure()
plot(diff_load)
title('First order differentiated load profile')
diff2_load = diff(diff_load); % Makes the series stationary on variance
figure()
plot(diff2_load)
title('Second order differentiated load profile')
%acf measures the correlation between x and x+k values, acf plot shows
correlation between lags 1:last lag ; hour ahead - 60/5 = 12, 1 day -
24*(60/5) = 288 , 1 month - 288 * 30 , 6 month ahead - 288*30*6
lag_h = 288; %Check correlation of 12,288 values for see if the 1st and the
12/288th values are correlated
[acf,lags,bounds] = autocorr(input_load,lag_h);
% The series has positive correlations to a high number of lags hence the
% series requires differencing.
Feedback_delays = lags((acf > bounds(1,1)) | (acf < bounds(2,1)));
q = Feedback_delays;

[PACF, Plags, Pbounds] = parcorr(input_load,lag_h);
%The PACF cuts off after the 8th lag implying an AR(8) model
P = lags((PACF > Pbounds(1,1)) | (PACF < Pbounds(2,1)));
figure();
subplot(2,1,1)
lineHandles = stem(lags,acf,'filled','r-o');
set(lineHandles(1),'MarkerSize',4)
grid('on')
xlabel('Lag')
ylabel('Sample Autocorrelation')
title('Sample Autocorrelation Function')
hold('on')
plot(lags,acf);hold on; plot(lags,repmat(bounds(1,1),1,max(lags)+1));hold
on;plot(lags,repmat(bounds(2,1),1,max(lags)+1))
subplot(2,1,2)
lineHandles = stem(Plags,PACF,'filled','r-o');
set(lineHandles(1),'MarkerSize',4)
grid('on')
xlabel('Lag')
ylabel('Sample Partial Autocorrelations')
title('Sample Partial Autocorrelation Function')
hold('on')

```

```

    plot(Plags,PACF);hold on;
plot(Plags, repmat(Pbounds(1,1),1,max(Plags)+1));hold
on;plot(Plags, repmat(Pbounds(2,1),1,max(Plags)+1))

% if we have a linearly decaying sample ACF indicates a nonstationary
% process (time-serie) for that a differencing should be done using the diff
% function in matlab as follows:

%1 degree differencing
[ACF_D, lags_D, bounds_D] = autocorr(diff_load,lag_h);
% The autocorrelations are positive till the thirty third lag, this is
% rather high too. The series require further differencing.
Feedback_delays_D = lags_D((ACF_D > bounds_D(1,1)) | (ACF_D <
bounds_D(2,1)));
q_D = Feedback_delays_D;
[PACF_D, Plags_D, Pbounds_D] = parcorr(diff_load,lag_h);
P_D = lags((PACF_D > Pbounds_D(1,1)) | (PACF_D < Pbounds_D(2,1)));
figure();
subplot(2,1,1)
lineHandles = stem(lags_D,ACF_D, 'filled', 'r-o');
set(lineHandles(1), 'MarkerSize', 4)
grid('on')
xlabel('Lag')
ylabel('Sample Autocorrelation')
title('Sample Autocorrelation Function')
hold('on')
plot(lags_D,ACF_D);hold on;
plot(lags_D, repmat(bounds_D(1,1),1,max(lags_D)+1));hold
on;plot(lags_D, repmat(bounds_D(2,1),1,max(lags_D)+1))
subplot(2,1,2)
lineHandles = stem(lags_D,PACF_D, 'filled', 'r-o');
set(lineHandles(1), 'MarkerSize', 4)
grid('on')
xlabel('Lag')
ylabel('Sample Partial Autocorrelations')
title('Sample Partial Autocorrelation Function')
hold('on')
plot(Plags_D,PACF_D);hold on;
plot(Plags_D, repmat(Pbounds_D(1,1),1,max(Plags_D)+1));hold
on;plot(Plags_D, repmat(Pbounds_D(2,1),1,max(Plags_D)+1))

%2 degree differencing
[ACF_DD, lags_DD, bounds_DD] = autocorr(diff2_load,lag_h);
% The autocorrelations are positive till the thirty third lag, this is
% rather high too. The series require further differencing.
Feedback_delays_DD = lags_DD((ACF_DD > bounds_DD(1,1)) | (ACF_DD <
bounds_DD(2,1)));
q_DD = Feedback_delays_DD;
[PACF_DD, Plags_DD, Pbounds_DD] = parcorr(diff2_load,lag_h);
P_DD = lags((PACF_DD > Pbounds_DD(1,1)) | (PACF_DD < Pbounds_DD(2,1)));
figure();
subplot(2,1,1)
lineHandles = stem(lags_DD,ACF_DD, 'filled', 'r-o');
set(lineHandles(1), 'MarkerSize', 4)
grid('on')
xlabel('Lag')

```

```

    ylabel('Sample Autocorrelation')
    title('Sample Autocorrelation Function')
    hold('on')
    plot(lags_DD,ACF_DD);hold on;
plot(lags_DD, repmat(bounds_DD(1,1),1,max(lags_DD)+1));hold
on;plot(lags_DD, repmat(bounds_DD(2,1),1,max(lags_DD)+1))
    subplot(2,1,2)
    lineHandles = stem(lags_DD,PACF_DD, 'filled', 'r-o');
    set(lineHandles(1), 'MarkerSize', 4)
    grid('on')
    xlabel('Lag')
    ylabel('Sample Partial Autocorrelations')
    title('Sample Partial Autocorrelation Function')
    hold('on')
    plot(Plags_DD,PACF_DD);hold on;
plot(Plags_DD, repmat(Pbounds_DD(1,1),1,max(Plags_DD)+1));hold
on;plot(Plags_DD, repmat(Pbounds_DD(2,1),1,max(Plags_DD)+1))

% The Auto Correlation is negative and near -0.5 at the first lag and
% the function at further lags are small and all patternless. Thus, the
% second order differential suits the best on the series. Since the time
% required two orders of differencing, it can be concluded
% that the load time series has a time-varying trend

%% Choosing AR and MA terms
% The lag beyond which the ACF cuts off is the indicated number of MA
terms.
% The lag beyond which the PACF cuts off is the indicated number of AR
terms.

% Choosing MA term
% Since the PACF at the first lag is negative, an MA term is required
% for the series. The lag beyond which the ACF cuts off indicates the
% number of MA terms. i.e. MA = 2. MA term rectifies overdifferencing of
% time series

% Choosing AR term
% An AR term is chosen if the PACF of the differenced time series cuts
% off sharply or the lag 1 value is positive indicating underdifferncng.
Since the third order
% differentiated time series PACF cuts off after lag 8
% Chosen terms: AR = 8; I = 2 ; MA = 2

% Choosing seasonal lags - The series has a seasonal pattern that
% represents the electricity usage over a day repeated throughout the
% month. A day corresponds to 288 units of time steps in the case of
% load profiles. Therefore, the auto correlation at lag 288 is observed
% to determine the seasonal terms of the ARIMA model. The 288th term is
% positive, hence a SAR term is used in the model.
%% creating the ARIMA-mode
% Change seasonality - Order of seasonal differencing - use 1
% 5 min resolution, (60/5) * 24 = 6912 points in one day. Declare seasonality
= length(input_load/6912) to capture seasonality change in daily values
% Creating ARIMA(1,2,1) model

```

```

ARIMA = arima('SMALags', 288,
'SARLags',288,'MALags',1,'ARLags',1,'Seasonality',288,'D',2); % ARIMA(1,2,1)
with seasonal AR(288) and MA(288)
% Creating the ARIMA(1,2,2) model
%'SMALags',288,
ARIMA_1 =
arima('D',2,'SMALags',288,'SARLags',288,'MALags',2,'ARLags',1,'Seasonality',2
88); % ARIMA(1,2,2) with seasonal AR(288)
%% tuning (training or estimating) stage of the built ARIMA model
% for tuning our ARIMA model we will need to use the estimate function in
% matlab with a training time series. as following
Est_ARIMA = estimate(ARIMA,input_load,'Display','full');
Est_ARIMA_1 = estimate(ARIMA_1,input_load,'Display','full'); %Report Z
statistic (Mean - estimated mean / standard deviation of sample size)

%% check goodness of fit
% now we will try to check of the residuals are normally distributed and
% uncorrelated by using the infer function in matlab and finally plot the
% results by the following code:
res = infer(Est_ARIMA,input_load);
res_1 = infer(Est_ARIMA_1,input_load);
% for plotting the results How to infer ARIMA estimates
http://www.itl.nist.gov/div898/handbook/pmc/section6/pmc624.htm
figure();
subplot(2,2,1) %PACF decays faster than ACF plot
plot(res./sqrt(Est_ARIMA.Variance));grid on
title('Standardized Residuals')
subplot(2,2,2)%Weibull distribution works best. Limited points at the highest
quantile of Weibull distribution suggest abnormal recordings of load data
qqplot(res);grid on %Significant departure at from normal at early and later
Standard Quantiles suggest distribution is not normal
subplot(2,2,3)
autocorr(res)
subplot(2,2,4)
parcorr(res)
figure();
subplot(2,2,1)
plot(res_1./sqrt(Est_ARIMA_1.Variance));grid on
title('Standardized Residuals')
subplot(2,2,2)
qqplot(res_1);grid on % Demonstrate without pd too
subplot(2,2,3)
autocorr(res_1)
subplot(2,2,4)
parcorr(res_1)

%% Forecasting stage using the Est_STSFMs_ARIMA (where the coefficients
are now known)
% for foreacasting use the forecast matlab function as following:
for i = 2:1:10 % Simulating 8 days
Target_load = load_07((288*(i+1)+1):288*(i+2));
Y=load_07((288*(i-2))+1:288*(i+1));
N = length(Target_load);% forecast horizon - 1 day
[Yc,YcMSE,U] = forecast (Est_ARIMA,N,'Y0',Y);% the second input is the
forecast horizon for more details refer to the created report by the author
and matlab documentation.

```

```

[Yc_1,YcMSE_1,U_1] = forecast (Est_ARIMA_1,N,'Y0',Y);
% for showing the forecasted results use the plot function in matlab as
% following:
figure();
subplot(2,1,1); %Target_wind - Wind to measure performance against
plot(Target_load);grid on;hold on;plot(Yc','-r');xlabel('Prediction Step
Index');ylabel('Load');title('Estimated ARIMA(1,2,1) model for load
forecasting');legend('Expected Outputs','ARIMA Predictions');hold off
subplot(2,1,2);
plot(Target_load);grid on;hold on;plot(Yc_1','-r');xlabel('Prediction Step
Index');ylabel('Load');title('Estimated seasonal ARIMA(1,2,2) model for load
forecasting');legend('Expected Outputs','ARIMA Predictions');hold off

mape(i) = mean(abs((Yc - Target_load) ./ Target_load));
mape_1(i) = mean(abs((Yc_1 - Target_load) ./ Target_load));
end

%Removing Infinity values
mape(mape == Inf) = [];
mape_1(mape_1 == Inf) = [];
mmape = mean(mape);
mmape_1 = mean(mape_1);
fprintf('The range of MAPEs for the first model is %d to %d while the mean of
MAPES is %d and the range of MAPEs for the second model is %d to %d while the
mean of MAPES is %d\n',min(mape),max(mape), mmape, min(mape_1), max(mape_1),
mmape_1)

```

ARIMA – wind power

```

clc;
clear all;
close all;
yr=07
wind_all;

%Try removing AR lags
%Check if the distribution is stationary or non-stationary. Since the ACF
%plot does not converge to zero, we conclude it is non stationary and needs
%differencing

%Subset input wind to the wind that needs to be analyzed
input_wind = wind_07(1:288*30);
figure()
plot(input_wind)
title('wind series')
%The time series is differentiated to remove seasonality of data
diff_wind = diff(input_wind);
figure()
plot(diff_wind)
title('First order differentiated wind profile')
diff2_wind = diff(diff_wind); % Makes the series stationary on variance
figure()
plot(diff2_wind)
title('Second order differentiated wind profile')

```

```

%acf measures the correlation between x and x+k values, acf plot shows
correlation between lags 1:last lag ; hour ahead - 60/5 = 12, 1 day -
24*(60/5) = 288 , 1 month - 288 * 30 , 6 month ahead - 288*30*6
lag_h = 288; %Check correlation of 12,288 values for see if the 1st and the
12/288th values are correlated
[acf,lags,bounds] = autocorr(input_wind,lag_h);
% The series has positive correlations to a high number of lags hence the
% series requires differencing.
Feedback_delays = lags((acf > bounds(1,1)) | (acf < bounds(2,1)));
q = Feedback_delays;

[PACF, Plags, Pbounds] = parcorr(input_wind,lag_h);
%The PACF cuts off after the 8th lag implying an AR(8) model
P = lags((PACF > Pbounds(1,1)) | (PACF < Pbounds(2,1)));
figure();
subplot(2,1,1)
lineHandles = stem(lags,acf,'filled','r-o');
set(lineHandles(1),'MarkerSize',4)
grid('on')
xlabel('Lag')
ylabel('Sample Autocorrelation')
title('Sample Autocorrelation Function')
hold('on')
plot(lags,acf);hold on; plot(lags, repmat(bounds(1,1),1,max(lags)+1));hold
on;plot(lags, repmat(bounds(2,1),1,max(lags)+1))
subplot(2,1,2)
lineHandles = stem(Plags,PACF,'filled','r-o');
set(lineHandles(1),'MarkerSize',4)
grid('on')
xlabel('Lag')
ylabel('Sample Partial Autocorrelations')
title('Sample Partial Autocorrelation Function')
hold('on')
plot(Plags,PACF);hold on;
plot(Plags, repmat(Pbounds(1,1),1,max(Plags)+1));hold
on;plot(Plags, repmat(Pbounds(2,1),1,max(Plags)+1))

% if we have a linearly decaying sample ACF indicates a nonstationary
% process (time-serie) for that a differencing should be done using the diff
% function in matlab as follows:

%1 degree differencing
[ACF_D, lags_D, bounds_D] = autocorr(diff_wind,lag_h);
% The autocorrelations are positive till the thirty third lag, this is
% rather high too. The series require further differencing.
Feedback_delays_D = lags_D((ACF_D > bounds_D(1,1)) | (ACF_D <
bounds_D(2,1)));
q_D = Feedback_delays_D;
[PACF_D, Plags_D, Pbounds_D] = parcorr(diff_wind,lag_h);
P_D = lags_D((PACF_D > Pbounds_D(1,1)) | (PACF_D < Pbounds_D(2,1)));
figure();
subplot(2,1,1)
lineHandles = stem(lags_D,ACF_D,'filled','r-o');
set(lineHandles(1),'MarkerSize',4)
grid('on')
xlabel('Lag')

```

```

    ylabel('Sample Autocorrelation')
    title('Sample Autocorrelation Function')
    hold('on')
    plot(lags_D,ACF_D);hold on;
plot(lags_D, repmat(bounds_D(1,1),1,max(lags_D)+1));hold
on;plot(lags_D, repmat(bounds_D(2,1),1,max(lags_D)+1))
    subplot(2,1,2)
    lineHandles = stem(lags_D,PACF_D, 'filled', 'r-o');
    set(lineHandles(1), 'MarkerSize',4)
    grid('on')
    xlabel('Lag')
    ylabel('Sample Partial Autocorrelations')
    title('Sample Partial Autocorrelation Function')
    hold('on')
    plot(Plags_D,PACF_D);hold on;
plot(Plags_D, repmat(Pbounds_D(1,1),1,max(Plags_D)+1));hold
on;plot(Plags_D, repmat(Pbounds_D(2,1),1,max(Plags_D)+1))

% The Auto Correlation is negative and near -0.5 at the first lag and
% the function at further lags are small and all patternless. Thus, the
% second order differential suits the best on the series. Since the time
% required two orders of differencing, it can be concluded
% that the wind time series has a time-varying trend

%% Choosing AR and MA terms
% The lag beyond which the ACF cuts off is the indicated number of MA
terms.
% The lag beyond which the PACF cuts off is the indicated number of AR
terms.

% Choosing MA term
% Since the PACF at the first lag is negative, an MA term is required
% for the series. The lag beyond which the ACF cuts off indicates the
% number of MA terms. i.e. MA = 5. MA term rectifies overdifferencing of
% time series

% Choosing AR term
% An AR term is chosen if the PACF of the differenced time series cuts
% off sharply or the lag 1 value is positive indicating underdifferncng.
Since the third order
% differentiated time series PACF cuts off after lag 3
% Chosen terms: AR = 3; I = 1 ; MA = 5

% Choosing seasonal lags - The series has a seasonal pattern that
% represents the electricity usage over a day repeated throughout the
% month. A day corresponds to 288 units of time steps in the case of
% wind profiles. Therefore, the auto correlation at lag 288 is observed
% to determine the seasonal terms of the ARIMA model. The 288th term is
% positive, hence a SAR term is used in the model.

%No effect of changing MA terms

%% creating the ARIMA-mode
% Change seasonality - Order of seasonal differencing - use 1

```

```

% 5 min resolution, (60/5) * 24 = 6912 points in one day. Declare seasonality
= length(input_wind/6912) to capture seasonality change in daily values
% Creating ARIMA(1,1,0) model
ARIMA = arima('ARLags',1,'SARLags',288,'SMALags',
288,'Seasonality',288,'D',1); % ARIMA(1,1,0) with seasonal AR(288) and
MA(288)
% Creating the ARIMA(1,1,1) model
%'SMALags',288,
ARIMA_1 =
arima('D',1,'ARLags',1,'MALags',1,'SMALags',288,'SARLags',288,'Seasonality',2
88); % ARIMA(1,1,1) with seasonal AR(288)
%% tuning (training or estimating) stage of the built ARIMA model
% for tuning our ARIMA model we will need to use the estimate function in
% matlab with a training time series as following
Est_ARIMA = estimate(ARIMA,input_wind,'Display','full');
Est_ARIMA_1 = estimate(ARIMA_1,input_wind,'Display','full'); %Report Z
statistic (Mean - estimated mean / standard deviation of sample size)

%% check goodness of fit
% now we will try to check of the residuals are normally distributed and
% uncorrelated by using the infer function in matlab and finally plot the
% results by the following code:
res = infer(Est_ARIMA,input_wind);
res_1 = infer(Est_ARIMA_1,input_wind);
% for plotting the results How to infer ARIMA estimates
http://www.itl.nist.gov/div898/handbook/pmc/section6/pmc624.htm
figure();
subplot(2,2,1) %PACF decays faster than ACF plot
plot(res./sqrt(Est_ARIMA.Variance));grid on
title('Standardized Residuals')
subplot(2,2,2)
qqplot(res);grid on %Significant departure at from normal at early and later
Standard Quantiles suggest distribution is not normal
subplot(2,2,3)
autocorr(res)
subplot(2,2,4)
parcorr(res)
figure();
subplot(2,2,1)
plot(res_1./sqrt(Est_ARIMA_1.Variance));grid on
title('Standardized Residuals')
subplot(2,2,2)
qqplot(res_1);grid on
subplot(2,2,3)
autocorr(res_1)
subplot(2,2,4)
parcorr(res_1)

%% Forecasting stage using the Est_STSFMs_ARIMA (where the coefficients
are now known)
% for forecasting use the forecast matlab function as following:
for i = 2:1:10 % Simulating 8 days
Target_wind = wind_07((288*(i+1)+1):288*(i+2));
Y=wind_07((288*(i-2))+1:288*(i+1));
N = length(Target_wind);% forecast horizon - 1 day

```

```

[Yc,YcMSE,U] = forecast (Est_ARIMA,N,'Y0',Y);% the second input is the
forecast horizon for more details refer to the created report by the author
and matlab documentation.
[Yc_1,YcMSE_1,U_1] = forecast (Est_ARIMA_1,N,'Y0',Y);
% for showing the forecasted results use the plot function in matlab as
% following:
figure();
subplot(2,1,1); %Target_wind - Wind to measure performance against
plot(Target_wind);grid on;hold on;plot(Yc,'-r');xlabel('Prediction Step
Index');ylabel('wind');title('Estimated ARIMA(1,1,0) model for wind
forecasting');legend('Expected Outputs','ARIMA Predictions');hold off
subplot(2,1,2);
plot(Target_wind);grid on;hold on;plot(Yc_1,'-r');xlabel('Prediction Step
Index');ylabel('wind');title('Estimated seasonal ARIMA(1,1,1) model for wind
forecasting');legend('Expected Outputs','ARIMA Predictions');hold off

mape(i) = mean(abs(((Yc - Target_wind)./ Target_wind)));
mape_1(i) = mean(abs(((Yc_1 - Target_wind)./Target_wind)));
end

%Removing Infinity values
mape(mape == Inf) = [];
mape_1(mape_1 == Inf) = [];
mmape = mean(mape);
mmape_1 = mean(mape_1);
fprintf('The range of MAPES for the first model is %d to %d while the mean of
MAPES is %d and the range of MAPES for the second model is %d to %d while the
mean of MAPES is %d\n',min(mape),max(mape), mmape, min(mape_1), max(mape_1),
mmape_1)

```