**Comparison of Neural Network Training Algorithms used to perform Day Ahead Electricity Load Forecasting**

**Abstract**

Electricity is a need and a strategic asset for national economies. As a result, electric utilities strive to balance power generation and demand to provide a decent service at a reasonable cost. For this reliable forecasting of load is required to solve the unit commitment problem and schedule the energy sources. The efficient operation of a power system is dependent on the correct tracking of system load. The first stage in power system operation planning is accurately predicting system load. Accurate short-term load forecasting is critical for more effective load generation scheduling, decreasing the generation-demand gap, and lowering power losses. Load forecasting is a complex process that includes input from the system as well as the environment and customers. The correct prediction of generation is also a problem with solar and wind energy, especially with the broad integration of renewable sources. So far, several strategies have been used to anticipate electrical load. Meanwhile, neural-network-based approaches resulted in less prediction inaccuracies due to their capacity to adapt to the hidden feature of the consuming load. The NN based method performance is assessed by computing several parameters such as root mean square error, mean square error, mean absolute error, mean absolute percentage error, and so on. In addition, three distinct optimization techniques are used to find the optimum ANN training algorithm: Levenberg–Marquardt, Bayesian Regularization, and Scaled Conjugate Gradient. The effectiveness of these optimization algorithms is verified in terms of training, test, validation, and error analysis. The proposed system simulation is carried out using the MATLAB software. According to the results, the Bayesian Regularization optimization algorithm based ANN model provides the best electrical load forecasting results.

1. **Introduction**

The successful operation of a power system depends on the accurate tracking of the system load to match the generation. Predicting the load on the system accurately is the first step in planning of the power system operations. Load forecasting involves many details and requires inputs from the system as well as from the environment and customers. Especially with the widespread renewable sources integration, the accurate prediction of generation is also an issue with solar and wind energy. The load forecasting needs to be done for various time frames in a system, for smooth control and efficient operation of the power system:

• The automatic generation control (AGC) function ensures that the load-generation balance is maintained online.

• The economic load dispatch function ensures that the larger load variations over a few minutes are distributed among the generators which are available and most economic.

• When the time frames are larger, hours or days, the large load changes are met by the starting up or shutting down of generating plants or importing or exporting power from neighboring areas. The EMS functions used for this purpose include hydro scheduling, unit commitment, hydrothermal coordination, and interchange evaluation.

• The large load variations over weeks are met economically by functions such as hydro scheduling, thermal scheduling, and maintenance scheduling.

• Contingency analysis is utilized to rank the severity of contingencies (offline) and also requires accurate load prediction.

**1.1 Artificial Neural Networks (ANN)**

The concept of ANN was introduced several years ago for different applications because of its capacity to forecast the data and also to control the system response effectively. It has been demonstrated that ANN is one of the effective solutions for all forms of real‐time nonlinear issues. An artificial neural network (ANN) is designed based on the interconnection of processing elements that carries information.

Multi-Layer Feed Forward Network: This network consists of one input layer, one output layer, and single or multi hidden layers. The processing elements for the hidden layer are hidden neurons only. Before sending the inputs to the output layer, computation is performed by hidden neurons in the hidden layer.

The FNN is trained to approximate the nonlinear function Fs(.) between the hourly load and the input variables. The FNN comprises a layer of input units, one or more hidden layer(s) and a layer of output units. An FNN with one hidden layer is shown in Figure. The input layer consists of Ni inputs. The hidden layer in our NN model has 30 neurons.

Each ith input is connected to the each jth unit of the hidden layer by a weighting factor, Wij. Each unit in the hidden layer, called a neuron, performs a nonlinear transformation of its weighted input signals.

**1.2 Back Propagation**

Back propagation is the algorithm for determining how a single training example would like to nudge the weights and biases. Not just in terms of whether they should go up or down, but in terms of what relative proportions to those changes cause the most rapid decrease to the cost.

A true gradient descent step would involve doing this for all your training examples and averaging the desired changes that you get, but that is computationally slow.

So we randomly sub divide data into mini batches and compute each step with respect to a mini batch repeatedly going through all the mini batches and making adjustments, we can converge to a local minimum of the cost function which will make the network better on training examples.

**1.3 The Levenberg-Marquardt Algorithm**

The Levenberg-Marquardt algorithm, developed independently by Kenneth Levenberg and Donald Marquardt, is a combination of steepest descent and the Gauss-Newton algorithm. The variance of the network is presented as:

*E(xk) = Σmk=1[e(ek)]2 = Σmk=1[dk − zk]*

In which:

• *xk* is the input data vector.

• *dk* is the training data {dk; 1 ≤ k ≤ m}.

• *zk* is forecast data {zk; 1 ≤ k ≤ m}

• Error: e(*xk*) = *dk* − *zk*

- Taylor expansion of a function f(x) in the neighborhood of *∆xk*

*E(x k+1) = E(xk + ∆xk) ≈ E(xk) + G(xk)∆xk + 0.5∆xkH(xk)∆xk* (2)

In which:

• *∆xk = xk+1 − xk*

• G(*xk*): Gradient of function E(x)

• H(*xk*): Hessian matrix of E(x)

- The partial derivative of equation 2 with respect to *∆xk* will be:

*G(xk)∆xk + H(xk)∆xk = 0*.

⇐⇒ *∆xk = −H(xk)( − 1)G(xk)* (3)

- Gradient and Hessian matrix of function E(x) can be rewritten as:

*G(xk) = 2JT(xk)E(xk).* (4)

*H(xk) = 2JT(xk)J(xk) + 2S(xk)* (5)

In which:

• J(xk): Jacobian matrix

• *S(xk) = Σmi=1ei(xk)∇2ei(xk)*

- Considering the value of S(xk) is so small, then we can approximate the Hessian matrix to:

*H(xk) ≈ 2JT(xk)J(xk)* (6)

- Substituting eq.3 and eq.5 into eq.3.2 we get:

*∆xk = −[2JT(xk)J(xk)]−1(2JT(xk)E(xk))* (7)

- However, the limitation of this algorithm is that the Hessian matrix can be difficult or impossible to inverse. To overcome this limitation, an approximation of the Hessian matrix is used:

*Hs(xk) ≈ H(xk) + µ.I* (8)

Where:

• I = Unit matrix

• µ = coherence coefficient

Thus, the Levenberg-Marquardt algorithm uses approximation and updates the weights for the Hessian matrix as follows:

*w k+1 = wk − [JT.J + µ.I]−1 JT E* (9)

Where:

• J : the Jacobian matrix that contains first derivatives of the network errors with respect to the weights w and biases b.

- When µ is zero, the Levenberg-Marquardt algorithm becomes Newton’s algorithm. When µ is large, this algorithm becomes a Gradient Descent algorithm with a small step size.

- Newton’s method is faster and more accurate near an error minimum, so the aim is to shift toward Newton’s method as quickly as possible. Thus, µ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function is always reduced at each iteration of the algorithm for ensuring the fast convergence.

**1.4 Scaled Conjugate Gradient Algorithm**

It is also possible to use another approach in estimating the step size than the line search technique. The idea is to combine the model trust region approach, known from the LM algorithm with the CG approach. This approach is known as SCG and introduced to literature by Møller (1993). SCG avoids the line search per learning iteration of the conventional gradient descent by using a Levenberg–Marquardt approach in order to scale the step size and contains no critical user-defined parameter.

SCG method searches along conjugate directions which usually lead to faster convergence. To train a neural network using SGC, the weights, net inputs, and transfer functions must have derivative functions. SCG viewed as a method in-between Newton’s method and gradient descent. In Newton’s method, some information such as the inversion of the Hessian matrix, storage, and evaluation is required, SCG avoids this information. This method also was implemented to accelerate the slow convergence linked with gradient descent method.

Consider having an initial parameter vector w0 and an initial training direction vector vo= -g0, the training directions can be constructed as:

vi+1 = gi+1 + viXi , i = 0, 1, 2, 3, ... (10)

Here v is the training direction vector, X is the conjugate parameter. In SCG, the training direction is always reset to the negative of the gradient. The next expression shows how the weights of the neural network can be updated.

wi+1 = wi + diri , i = 0, 1, 2, 3, ... (11)

**1.5 Bayesian regularization**

Regularization technique constrains the neural network to converge to a set of weights and biases having smaller values[27]. This causes the network response to be smoother and less likely to over fit to training patterns. In neural network regularization technique, the cost function F is defined as

F = γED + (1 − γ)EW (9)

where, ED is the sum of squared errors, EW = |w|2/2 is the sum of squares of the network parameters, and γ (<1.0) is the performance ratio parameter. One technique of automatically determining optimum regularization parameter is the Bayesian framework. This method allows the parameter to be selected using only the training data, without having to use separate training and validation data. Bayesian framework considers a probability distribution over the weight space, representing the relative degrees of belief in different values for the weights. The weight space is initially assigned some prior distribution. Let D = {xm, tm} be the training data set of the input-target pair. After the data is taken, the posterior probability distribution for the weight p(w|D,γ) is given according to the Bayesian rule as:

p(w|D, γ) = { p(D|w, γ)p(w|γ)} / p(D|γ) (10)

where, p(w|γ) is the prior distribution, p(D|w,γ) is the likelihood function and p(D|γ) is a normalization factor, which guarantees that the total probability is 1. In Bayesian framework, the optimal weight should maximize the posterior probability p(w|D,γ),

which is equivalent to minimizing the function in (9). The performance ratio parameter γ is optimized by applying the Bayes’ rule

p(γ|D) = { p(D|γ)p(γ) } / p(D). (11)

If a uniform prior density p(γ) is assumed for the regularization parameter γ, then maximizing the posterior probability is achieved by maximizing the likelihood function p(D|γ). Since all probabilities have a Gaussian form it can be expressed as

p(D|γ) = (π/γ)−N/2.{ π /(1 – γ) }−L/2 .ZF(γ) (12)

where, L is the total number of parameters in the NN. Supposing that F has a single minimum as a function of w at w\* and has shape of a quadratic function in a small area surrounding that point, ZF is approximated as

ZF = (2π)L/2 det−1/2H∗ exp(−F(w∗)) (13)

where, H = γ∇2ED + (1 − γ)∇2EW is the Hessian matrix of the objective function. Substituting value of ZF from (13) into (12), the optimum value of γ at the minimum point can be determined.

1. **Simulation**

The simulation has been performed using Neural Network Fitting toolbox of MATLAB R2021a. The historical load and temperature data were obtained from the region of New England, US.

**2.1 Training and Testing**

Training was done on the data from 1st Jan, 2018 to 30th June, 2021, and testing was done on the data from 1st July, 2021 to 30th June, 2022.

They were done using 3 different algorithms provided by MATLAB namely,

1. Levenberg Marquardt (LM) Algorithm
2. Bayesian Regularization (BR) Algorithm
3. Scaled Conjugate Gradient (SCG) Algorithm

Their load forecasts and errors were compared using different visualization graphs.

The accuracy of the models was computed using the Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE) metrics. These are defined as,

Where,  = kth actual load, = kth forecasted load

And n = number of load data points.

After the testing was done for each of the 3 algorithms, these results were obtained,

Using LM – 1. MAPE: 2.14%

2. MAE: 285.70 MW

3. Daily Peak MAPE: 1.97%

Using BR – 1. MAPE: 2.10%

2. MAE: 280.69 MW

3. Daily Peak MAPE: 1.93%

Using SCG – 1. MAPE: 2.32%

2. MAE: 310.19 MW

3. Daily Peak MAPE: 2.21%

The corresponding graphs for these results were plotted.

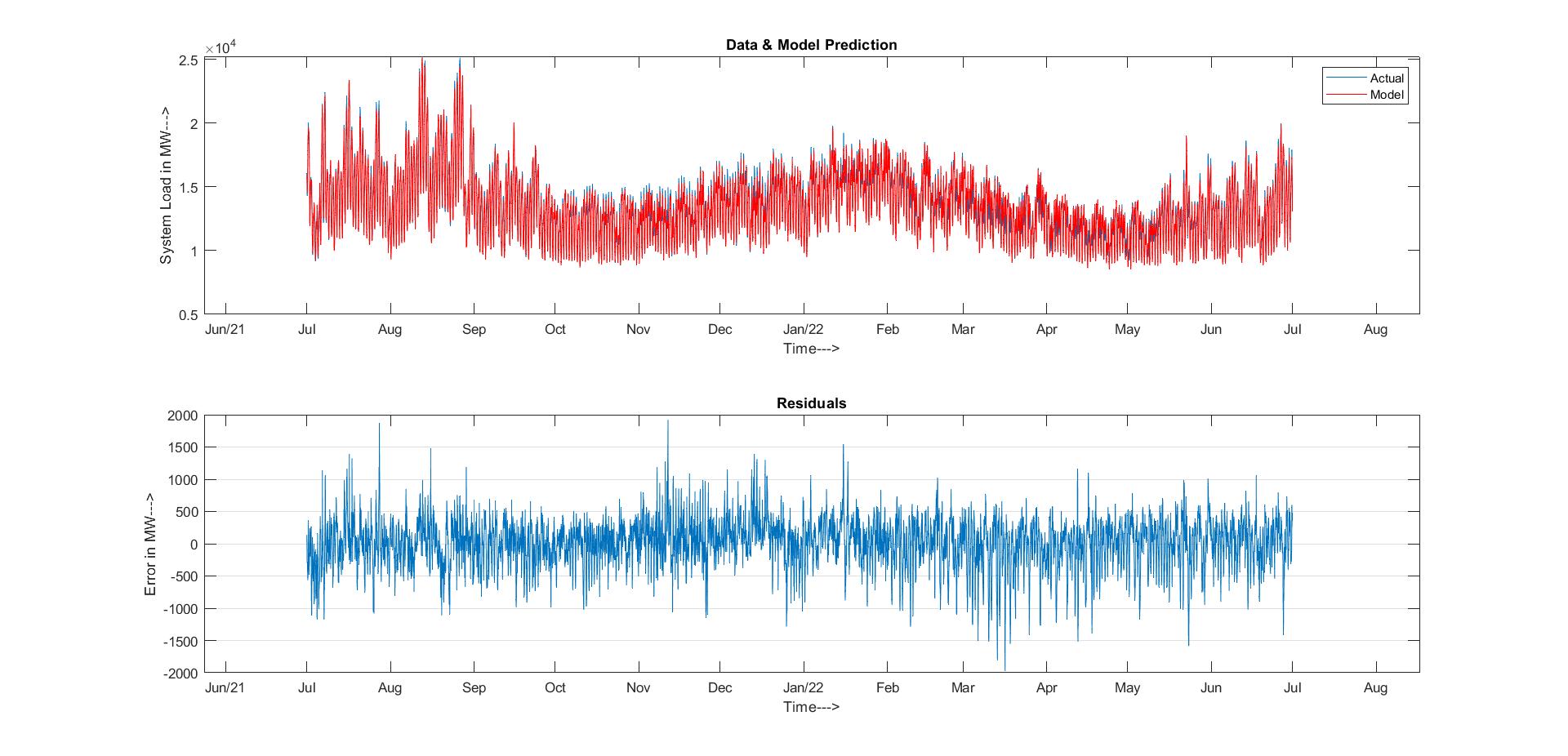


Fig 1. Actual load vs Forecasted Load and their residuals using LM algorithm.

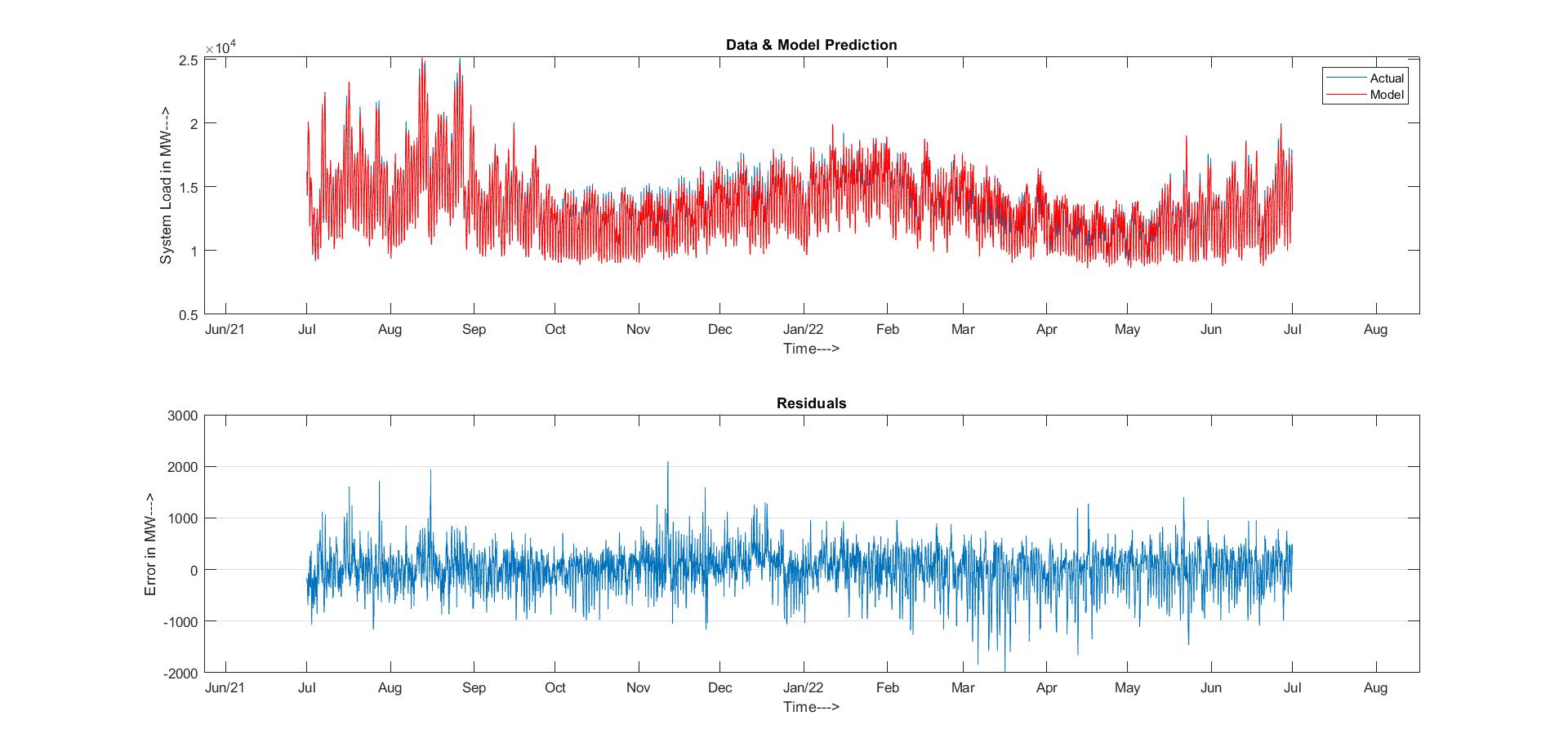


Fig 2. Actual load vs Forecasted Load and their residuals using BR algorithm.

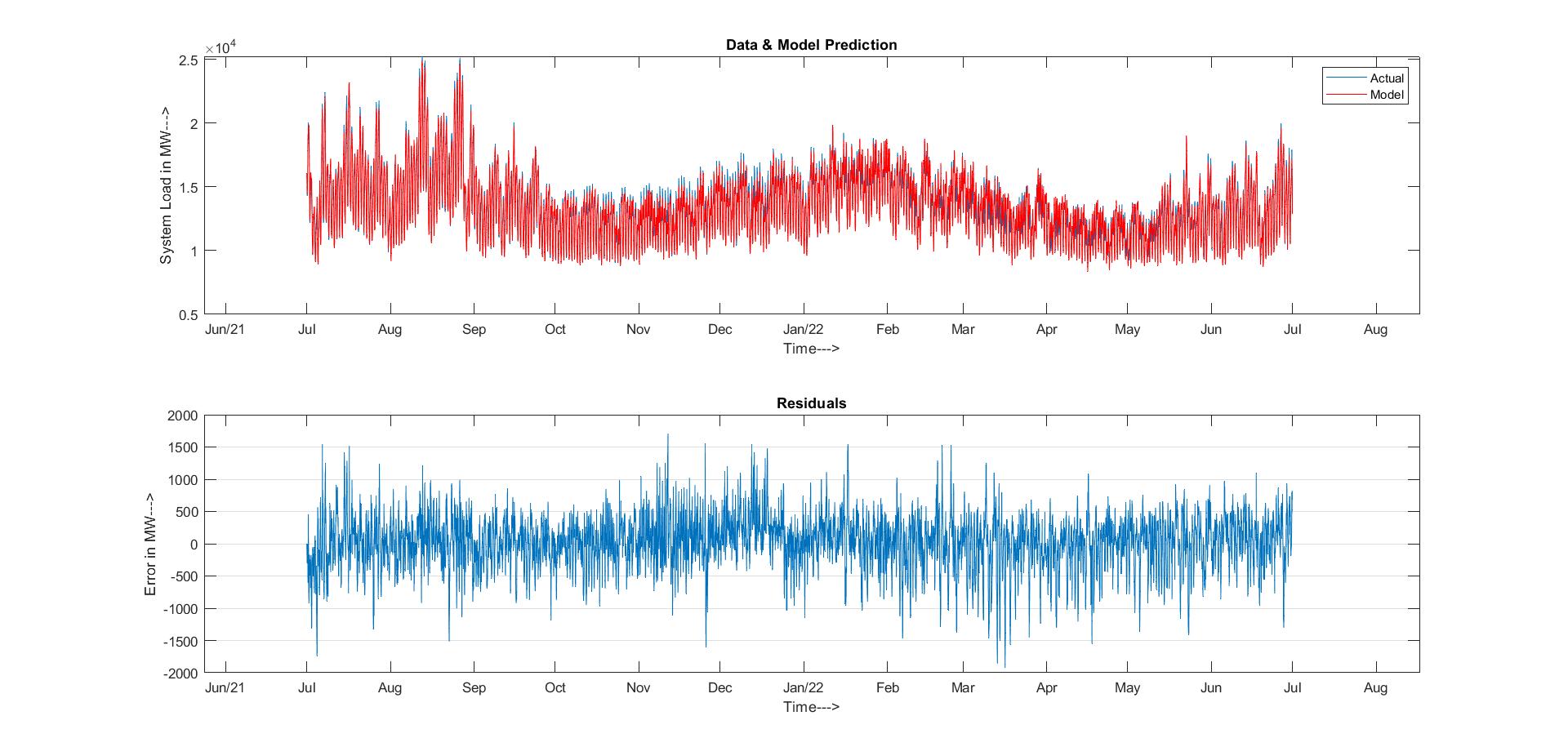


Fig 3. Actual load vs Forecasted Load and their residuals using SCG algorithm

The error distribution histograms for these testing results were also plotted.

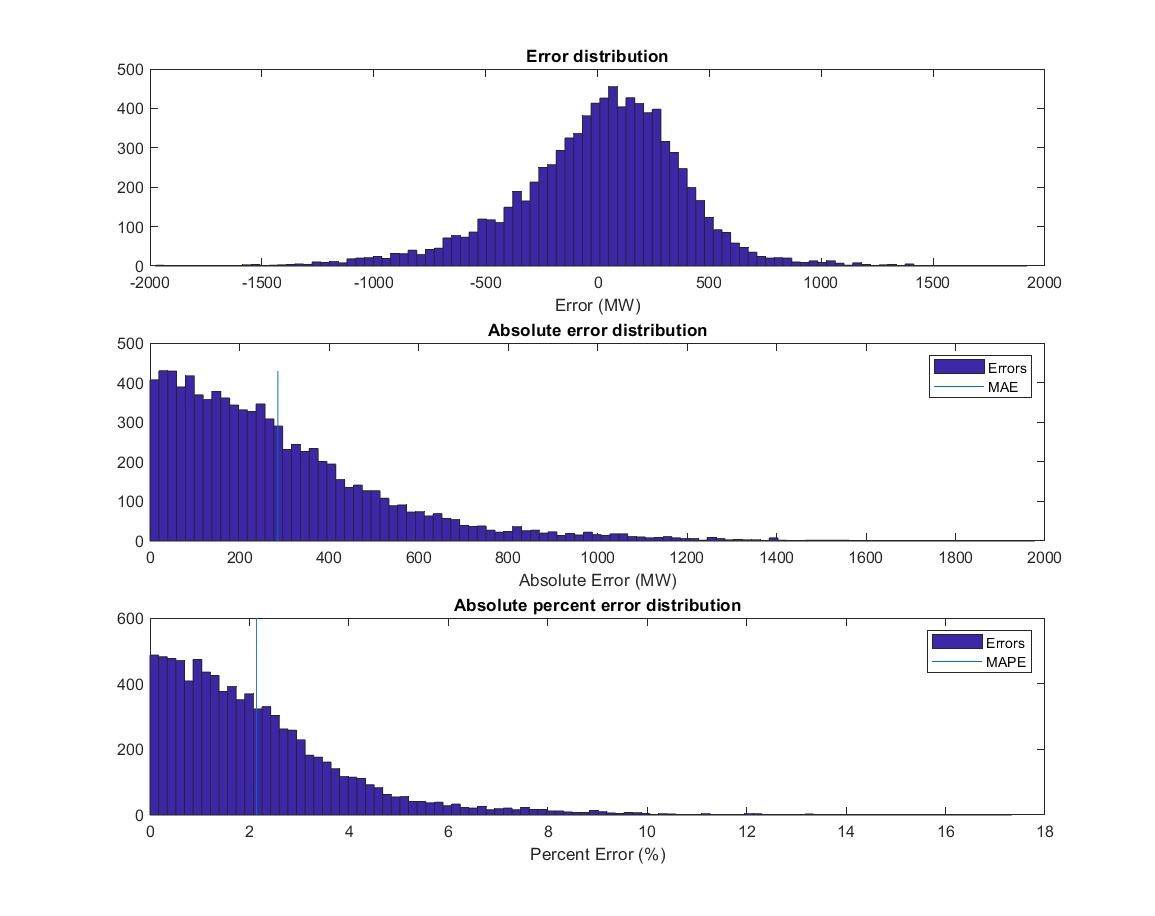
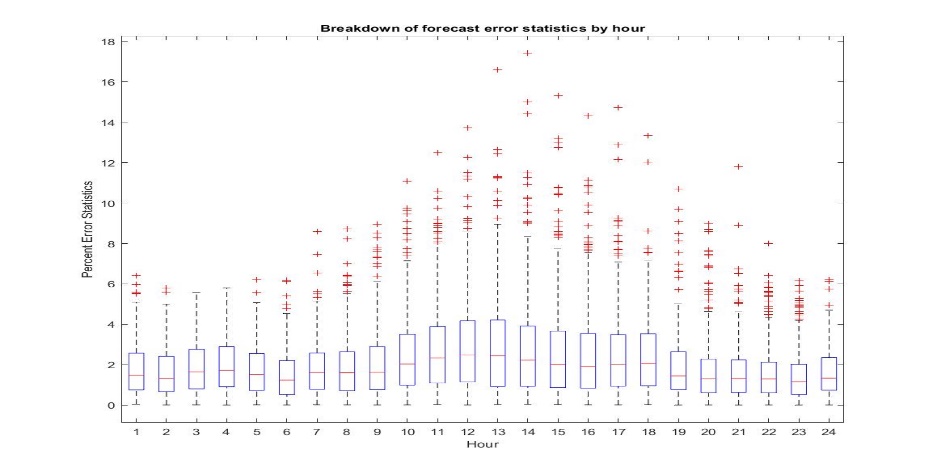


Fig 4. Error distribution, absolute error distribution & absolute percent error distribution histograms using LM algorithm

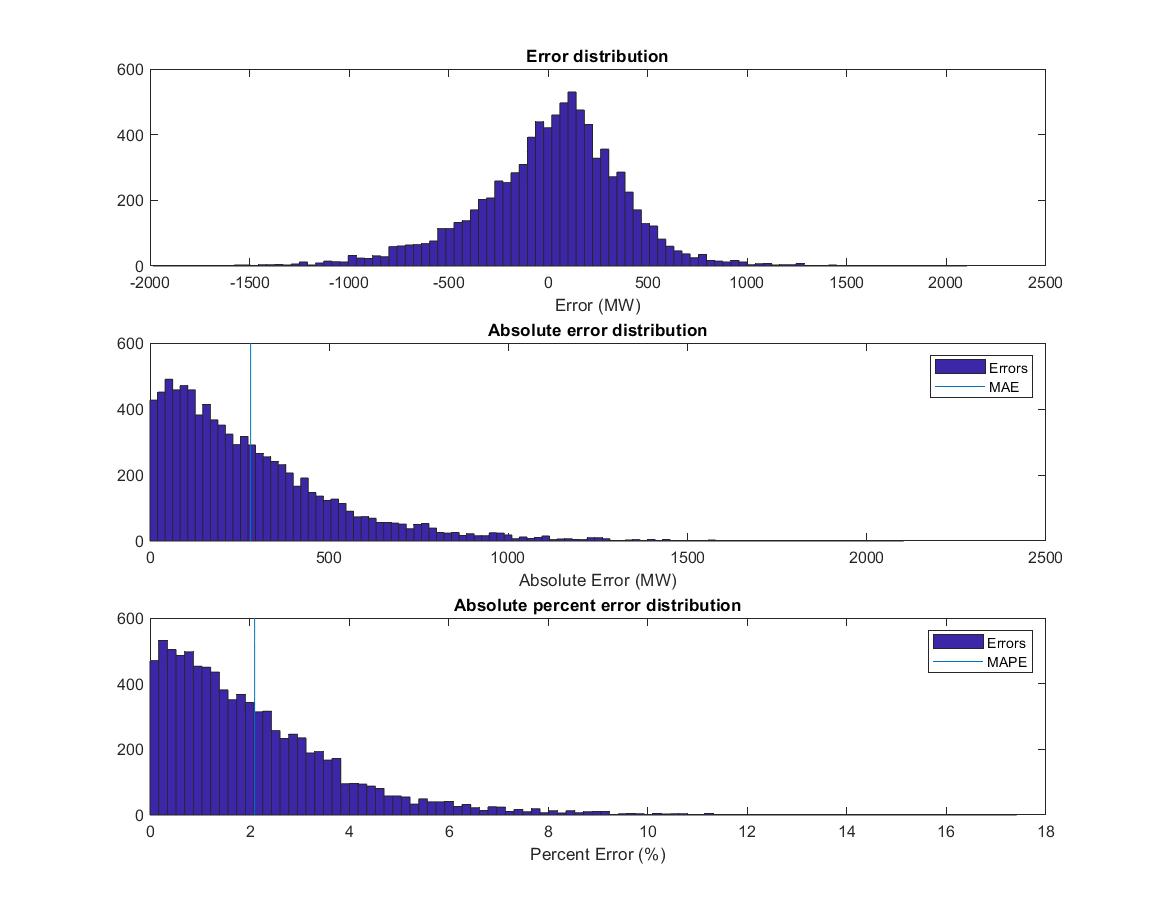


Fig 5. Error distribution, absolute error distribution & absolute percent error distribution histograms using BR algorithm

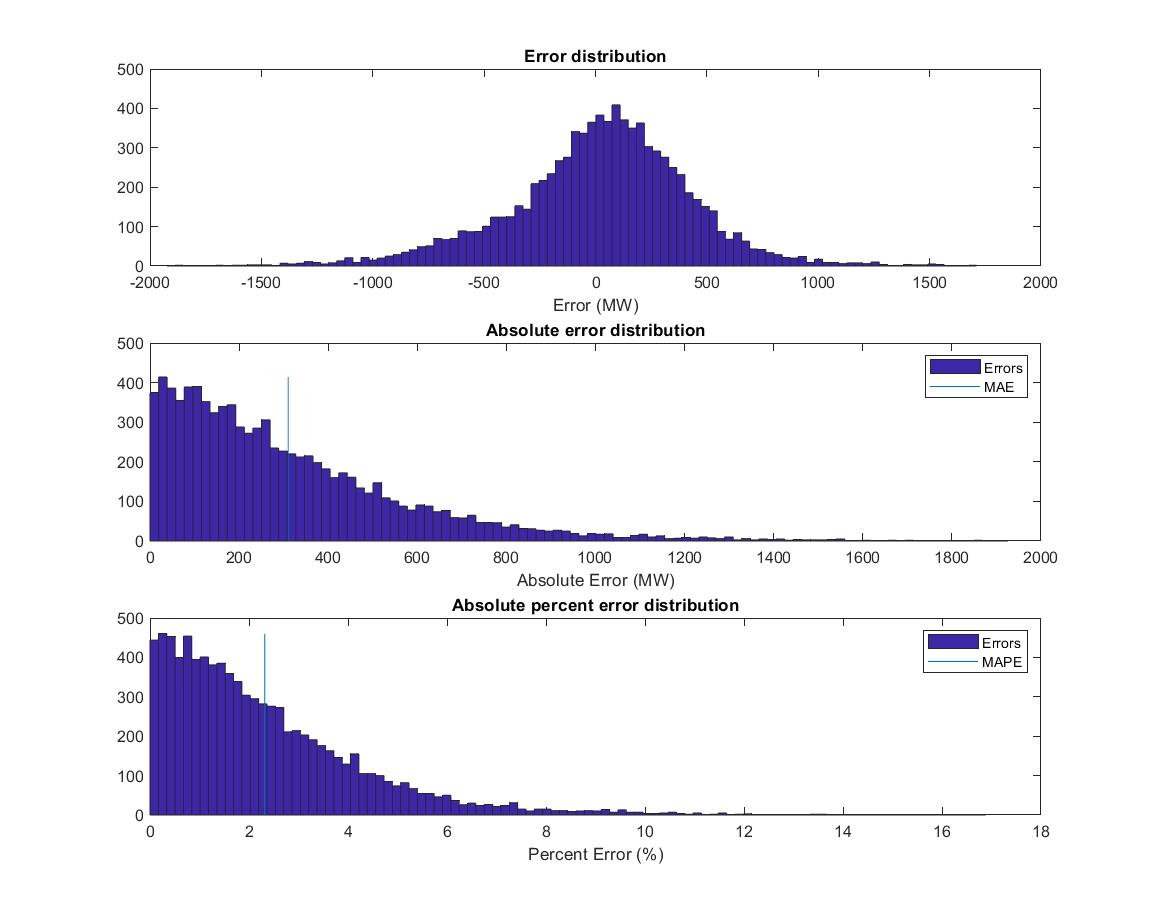


Fig 6. Error distribution, absolute error distribution & absolute percent error distribution histograms using SCG algorithm.

To get an idea of how predictable is the hourly load data, box plots were plotted for each of the 3 algorithms. They show the percentage error variations at different hours of the day for the whole testing data set.

Fig 7. Box Plot of Percent Error of forecasted load using LM algorithm as a function of hour of the day

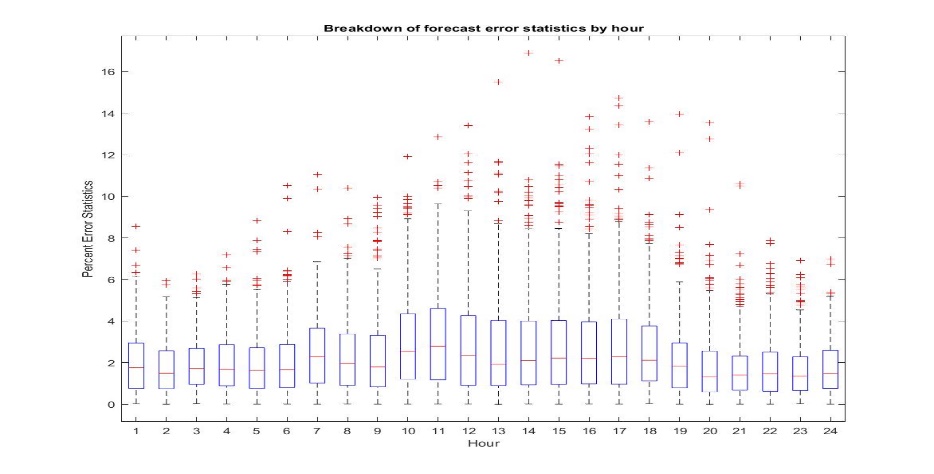


Fig 8. Box Plot of Percent Error of forecasted load using BR algorithm as a function of hour of the day

Fig 9. Box Plot of Percent Error of forecasted load using SCG algorithm as a function of hour of the day

On comparing and analyzing the results of the 3 algorithms, it was observed that the Bayesian Regularization algorithm gave the least MAPE error over the testing data. The error distribution histograms for BR algorithm had the most instances near 0 error. Therefore, the model trained using Bayesian Regularization algorithm was chosen to be used for performing Day-Ahead Load Forecasting.

1. **Results**

**3.1 Day-Ahead Load Forecasting in MS Excel**

The model trained and tested using BR algorithm was deployed as an Excel Add-In using the Library Compiler toolbox in MATLAB. The generated files were imported into a Macro-Enabled Excel file. Load data from a week prior, dry bulb and dew point temperature forecast for the day to be forecasted, date and a holiday indicator were used as inputs to the model.

Load for 30th September, 2022 was forecasted using the Excel App, after giving the respective inputs.

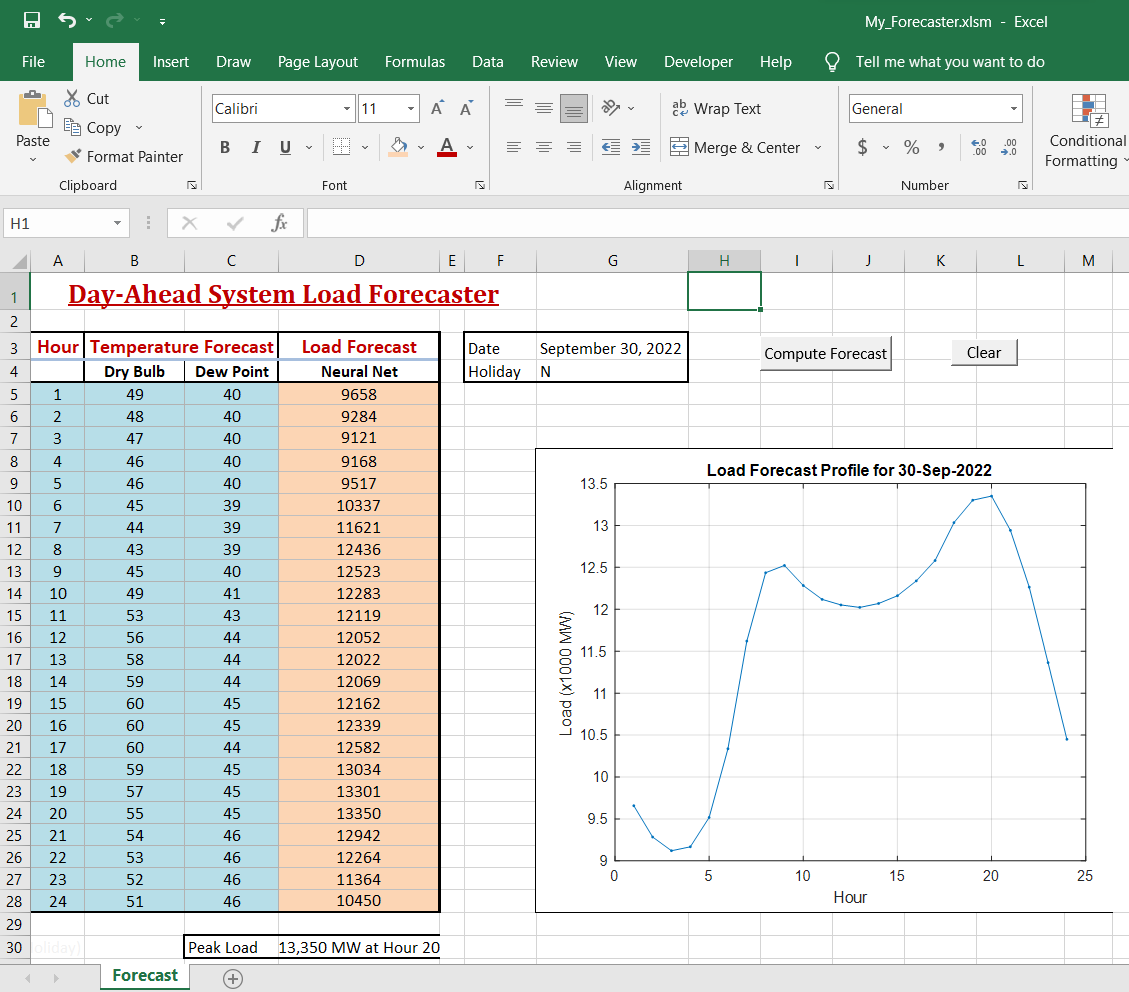


Fig 10. Day-Ahead Load Forecasting done on 30th September, 2022 using a Macro-Enabled MS Excel file, showing hourly load forecast and its corresponding graph

Error analysis was also done on the next day by comparing the forecasted load with the actual load. The results were a MAPE of 1.34% thereby, an accuracy of 98.66%.

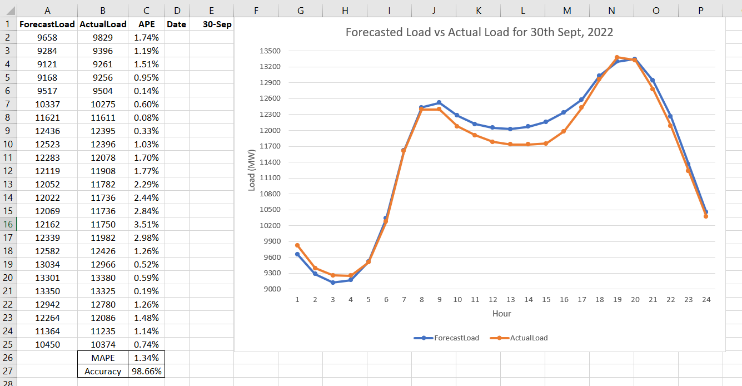


Fig 11. Comparison of Forecasted Load and Actual Load on 30th September, 2022 along with the MAPE and accuracy values

**Conclusion**

For the purpose of forecasting day-ahead electric load, this study examined the performance of various ANN algorithms. The input features chosen during data preparation showed to be successful because they produced low and acceptable MAPE. The optimal number of neurons in a load forecasting ANN model was discovered to be 30, and it was also learned that the Bayesian Regularization (BR) Algorithm works effectively, while the Scaled Conjugate Gradient (SCG) Algorithm does not in all trained neural network models. Bayesian Regularization (BR) Algorithm has the lowest MAPE of 2.10%, followed by Levenberg Marquardt (LM) Algorithm with a MAPE of 2.14%, and Scaled Conjugate Gradient of 2.32%. The day-ahead forecasting done on 30th of September, 2022 resulted in a MAPE of 1.34%.

**References**

[1] T. Hong and P. Wang, "Artificial Intelligence for Load Forecasting: History, Illusions, and Opportunities," in IEEE Power and Energy Magazine, vol. 20, no. 3, pp. 14-23, May-June 2022, doi: 10.1109/MPE.2022.3150808.

[2] M. S. Thomas and J. D. McDonald, Power System SCADA and Smart Grids, Boca Raton, Florida, USA: CRC PRESS, 2015.

[3] S. Sharif and J. H. Taylor, “Short-Term Load Forecasting by Feed-forward Neural Networks”, Proc. IEEE/ASME First International Energy Conference (IEC 2000), Al Ain, United Arab Emirates.

[4] “Chapter 4 What is backpropagation really doing?” ,Published Nov 3, 2017,Lesson by Grant Sanderson, text adaptation by Josh Pullen

[5] A. A. Bataineh and D. Kaur, "A Comparative Study of Different Curve Fitting Algorithms in Artificial Neural Network using Housing Dataset," NAECON 2018 - IEEE National Aerospace and Electronics Conference, 2018, pp. 174-178, doi: 10.1109/NAECON.2018.8556738.

[6] Giap, QH, DL Nguyen, TTQ Nguyen, and TMD Tran. “Applying Neural Network And Levenberg - Marquardt Algorithm for Load Forecasting in IA-Grai District, Gia Lai Province”. Journal of Science and Technology - University of Danang , vol 20, no. 6.2, June 2022, pp. 13-18, doi:10.31130/ud-jst.2022.240ICT.

[7] H. Okut, "Bayesian Regularized Neural Networks for Small n Big p Data", in Artificial Neural Networks - Models and Applications. London, United Kingdom: IntechOpen, 2016 [Online]. Available: https://www.intechopen.com/chapters/50570 doi: 10.5772/63256

[8] Zahra Shafiei Chafi, Hossein Afrakhte, "Short-Term Load Forecasting Using Neural Network and Particle Swarm Optimization (PSO) Algorithm", Mathematical Problems in Engineering, vol. 2021, Article ID 5598267, 10 pages, 2021.

[9] Lalit Mohan Saini ,”Peak load forecasting using Bayesian regularization, Resilient and adaptive backpropagation learning based artificial neural networks” , Electric Power Systems Research 78 (2008) 1302–1310