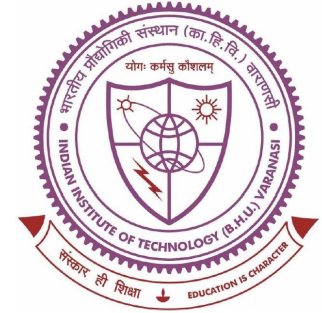


B.TECH. PROJECT

WEB APP FOR POWER-PLANTS



Submitted to : Prof. S.R. Mohanty,
Electrical Engineering Department,
IIT BHU, Varanasi

Submitted by : Gourav Sharma (18084008)
Raj Chaurasia(18085054)
Sagardeep Thakre(18085060)
Sahil Jain(18085061)
Sanidhya Singh(18085062)



Abstract

- Various electrical parameters in a power plant are calculated through smart energy meters. These meters upload the data to a central server (computer), which performs calculation and displays the required parameters to the user through a web-app.
- The various parameters calculated within the app include charges for each time block, the total charges for a day, and the additional penalties if there is sustained deviation for more than 12 consecutive time blocks.
- Finally, the app makes use of polynomial regression to predict future scheduled generation on the basis of past actual generation as a means to minimize the difference between the two, and consequently, to minimize costs.

Background

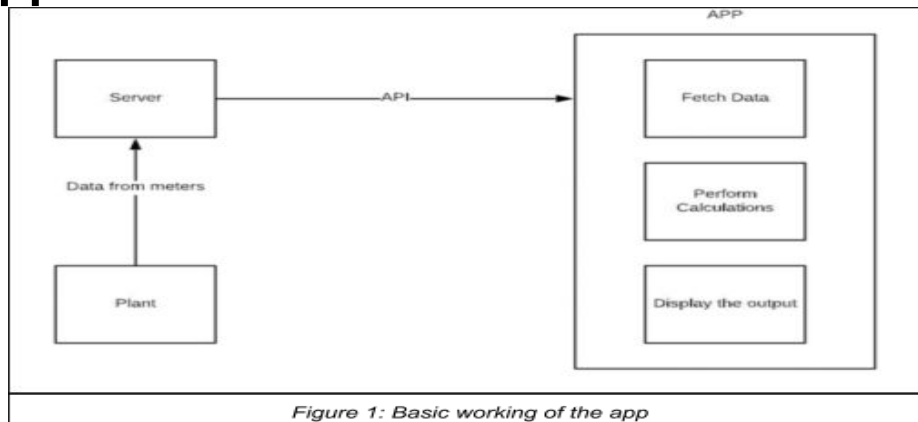
- Electrical power is supplied from sellers to buyers. This distribution of power is determined by a number of constraints based on price and other parameters. Two important parameters are the actual generation (AG) and the scheduled generation (SG).
- Sellers generate electricity as per a schedule dividing a day into 96 blocks of 15 minutes each. However, the buyer can demand more than the scheduled amount of power.
- This leads the seller to incur a loss, consequently slowing the generator blades as mechanical power is converted to electrical power, this loss must be compensated. Alternatively, since the consumer used more electricity than they were allotted, they should pay extra.

Terms Used

Terms	Meaning
Time block	Each day is divided into 96 time blocks of 15 minutes each
Scheduled Generation (SG)	It refers to the power which is scheduled to be provided by generators
Actual Generation (AG)	It refers to the power which is actually consumed by the buyers or provided by the sellers
Declared Capability	It is a measure of the contribution that a power station makes to the overall capacity of a distribution grid.
Android WebView	Built in browser in android SDK.
DOM	Document Object Model

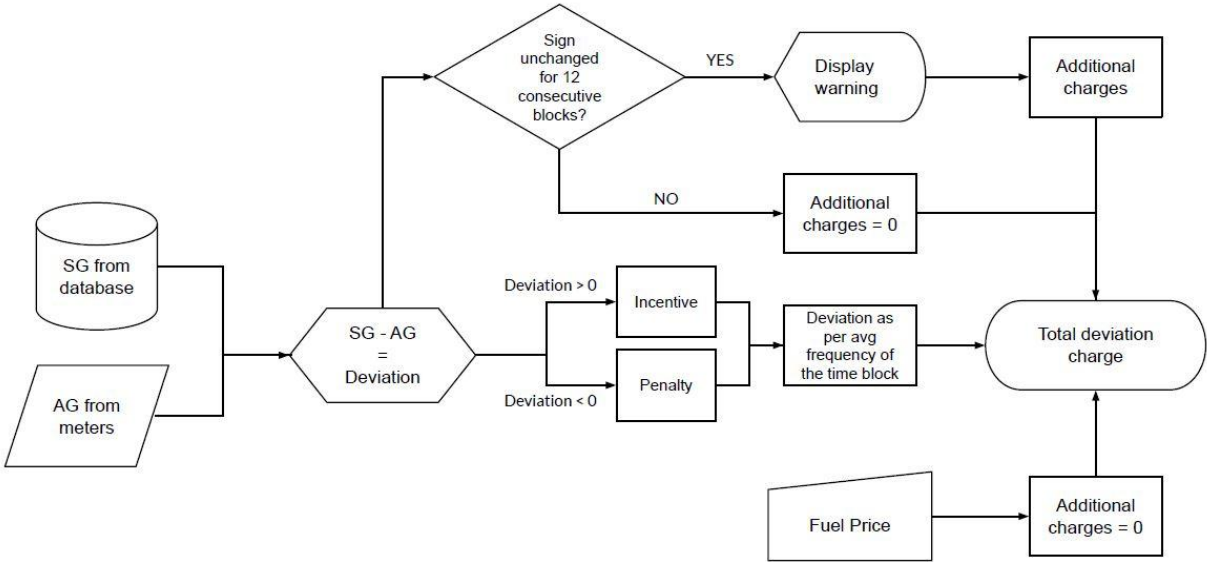
Table 1: Terms used

Working of the App



The server features a bunch of APIs (Application Programming Interfaces) which are used to exchange data between the web-app and master server. The data exchange takes place through these APIs, and the web-app displays the data when requested.

Calculations



Hosting the Data on Server

Server Setup

- Since the application is completely dependent on server side REST APIs for data so we have managed to replicate the present power-plant system.
- A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources respectively.
- We've hosted the data on an EC2 instance on AWS (Amazon Web Services). An EC2 instance is nothing but a virtual server in Amazon Web services terminology. It stands for Elastic Compute Cloud.
- We used Python and Django REST framework to build the server side application. The Django REST framework is a powerful and flexible toolkit for building Web APIs, and has a bunch of REST APIs responsible for healthy interaction of data between server and Web Application.

Generating the Data

- Ranges for various parameters were known and pseudorandom number generators (present in every programming language) were used to generate the data.
- The objective is to update variables in real time for each time block. This is trivial if this code is run constantly as we can just use a while loop to do so. However, due to the limits of the server, the entire code is run every time the data is accessed, which makes it complex.

Idea

- The random function present in every programming language is actually a pseudorandom number generator. What this means is that if we give it a certain input, it can only return a certain output. As an example, it can be thought of as the function $f(x) = x+1$: if we give 1 as an input it can only give 2 as an output.
- Along similar lines, if we give an input to the random function that stays the same for a given time block and changes for another time block, we will be able to generate the required variables. This idea is utilized in the following approach.

Approach

Every time someone accesses the data on the server, the following process takes place:

- The only input taken by the server is the current date and time which is taken from an internal clock present on every server (and on every device). On the basis of the current time, we find out the current block number and the current block time. Let's call them the parent variables.
- On the basis of the parent variables a function calculates the previous or next block time as needed. These functions can calculate the previous or next block time given a block time as an input. By using this function multiple times we can calculate the block time for a block that is an arbitrary number of time blocks ahead or behind.

The seed value or the input value for the random function is calculated using functions which have the inputs block no., day and month. The day and month are directly accessed using the server clock and the block no. has been calculated using the method discussed above.

- The function which returns the seed value for the variables that are updated in every time block takes input as block no., day and month. Now, all three of these values cannot be the same for a block. We require a function which can map these three values to a single value which is unique.
- The seed function is inspired from the **Rabin-Karp string matching algorithm**. Since the maximum values of the three variables block no., day and month are 96, 31 and 12, the highest of these is chosen. Now, we're using this value as a 'base' and exponentiating, which is exactly how we calculate values in the common number system. The function is as follows:

```
def function_seed(blk_no, day, mon):  
    return (96 * 96) * blk_no + 96 * day + mon
```

We can imagine that we are using a number system of base 96, and if we just input any number in the first, second and third place we'll obtain a unique number.

Developing the app

Fetching the data

For fetching the data from the apis, we have used the Streamlit Client Library. This is a built-in library for making network calls over the web and getting data in the form of JSON format. We only needed to add the dependency for the Library similar to import statement in programming.

Plotting the graph

For plotting graph, we used the Plot results obtained from the Polynomial Regression python notebook (Prediction Model).

Additional Features

The following additional features were also implemented in the app:

- **Alarm:** An alert will be shown if there is sustained deviation in one direction for more than 11 time blocks.
- **Prediction model:** A machine learning model to predict the actual generation is also implemented

Model to predict Actual Generation (AG)

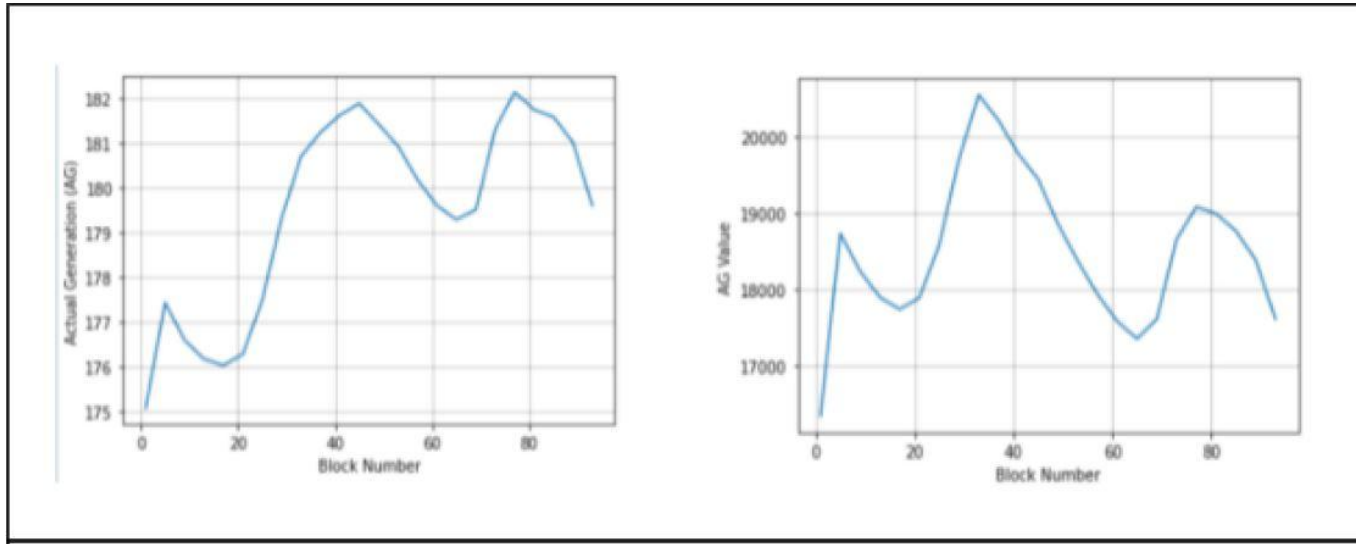
Data

Hourly electricity consumption data of a regional transmission organization (RTO in the US was used ([link](#)). The data was downloaded as a CSV file with each row having two values: datetime and the consumption in MW. The data was cleaned (by removing the days that didn't have 24 data points), resulting in a total data of 209 days in 5022 data points. Since the value of energy consumption (as per our needs) should lie between 150MW to 200MW. The data was scaled to lie within this range.

Datetime	AEP_MW
2004-12-31 1:00:00	13478
2004-12-31 2:00:00	12865
2004-12-31 3:00:00	12577
2004-12-31 4:00:00	12517
2004-12-31 5:00:00	12670
2004-12-31 6:00:00	13038
2004-12-31 7:00:00	13692
2004-12-31 8:00:00	14297
2004-12-31 9:00:00	14719
2004-12-31 10:00:00	14941
2004-12-31 11:00:00	15184
2004-12-31 12:00:00	15009
2004-12-31 13:00:00	14808
2004-12-31 14:00:00	14522
2004-12-31 15:00:00	14349

Observations (from data)

First, we plotted the energy consumption for a couple of days to check if there's a pattern. We noticed that the data resembles a polynomial function with peaks around the 40th and the 80th block. Consequently, we decided to implement a polynomial regression model for prediction.



Applying Polynomial Regression

We have data in the form of x_i (input data) and y_i (output data). We are required to predict a relationship between them. Suppose we have n such points (x_i, y_i) .

To predict a first-degree polynomial using these points, we'll consider the predicted polynomial as follows:

Here, β_0, β_1 are the parameters to be determined. To determine these parameters, we first get the cost function J as follows:

$$\bar{y} = (\beta_0 + \beta_1 x_i)$$
$$J = (\bar{y} - y_i)^2 = \sum_{i=1}^n (\beta_0 + \beta_1 x_i)^2$$

The cost function simply takes the squared error of our predicted value from the actual values. In order to have good predictions, it is required that the value of the cost function is minimum. For this, we partially differentiate the cost function with respect to each of the unknowns as follows:

$$\frac{\delta J}{\delta \beta_0} = 0 \quad \frac{\delta J}{\delta \beta_1} = 0 \quad \frac{\delta J}{\delta \beta_2} = 0$$

$$\beta_0 n + \beta_1 \sum x_i = \sum y_i \quad \dots(1)$$

$$\beta_0 \sum x_i + \beta_1 \sum x_i^2 = \sum y_i x_i \quad \dots(2)$$

Combining equations (1) and (2) in matrix form, we get

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix}$$

$$\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix}$$

Along similar lines, if we want to predict using a quadratic polynomial of the following form:

$$\bar{y} = (\beta_0 + \beta_1 x + \beta_2 x^2)$$

We will partially differentiate with respect to β_0 , β_1 , and β_2 . We'll obtain three equations which will then be combined into the following matrix form:

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix}$$

$$\boxed{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix}}$$

Based on the previous discussion, we can form the following generalization for a k-degree polynomial prediction function:

1. The cost function is obtained by taking the square of the difference of the predicted value from the actual value.
2. The cost function is differentiated with respect to $\beta_0, \beta_1 \dots \beta_k$ to obtain $(k+1)$ equations.
3. These equations are then combined into a single matrix equation and the value of the β matrix is found by solving the equation by taking inverse.

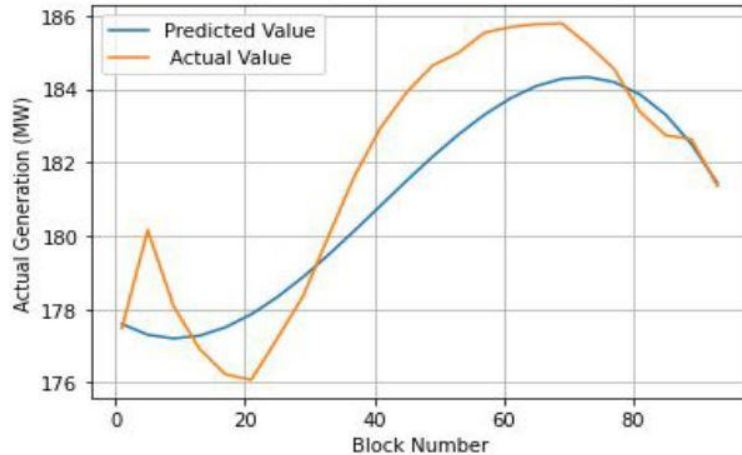
In general, we have the following:

$$\begin{bmatrix} \beta_0 \\ \dots \\ \beta_k \end{bmatrix} = \begin{bmatrix} n & \dots & \sum x_i^k \\ \dots & \dots & \dots \\ \sum x_i^k & \dots & \sum x_i^{2k} \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \dots \\ \sum x_i^k y_i \end{bmatrix}$$

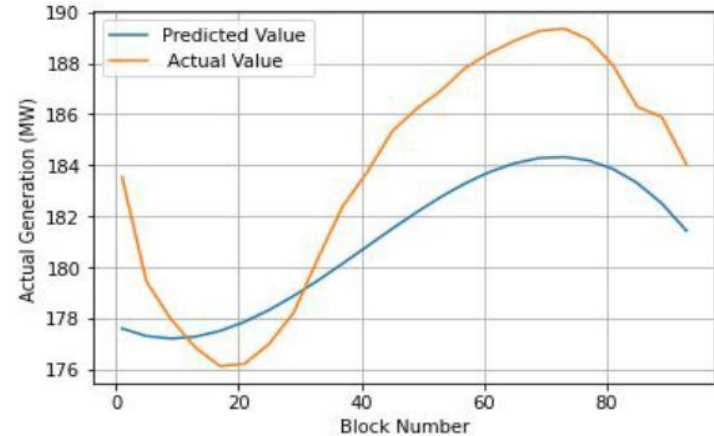
Predictions

We splitted the data into training and testing sets. The function for polynomial regression was trained using the training set. A split of 40% training data to 60% testing data gave the most desirable results. On further increasing the size of training data, the function lost the ability to generalize (due to overfitting). Finally, the prediction was done on the testing set.

Day 1 Prediction



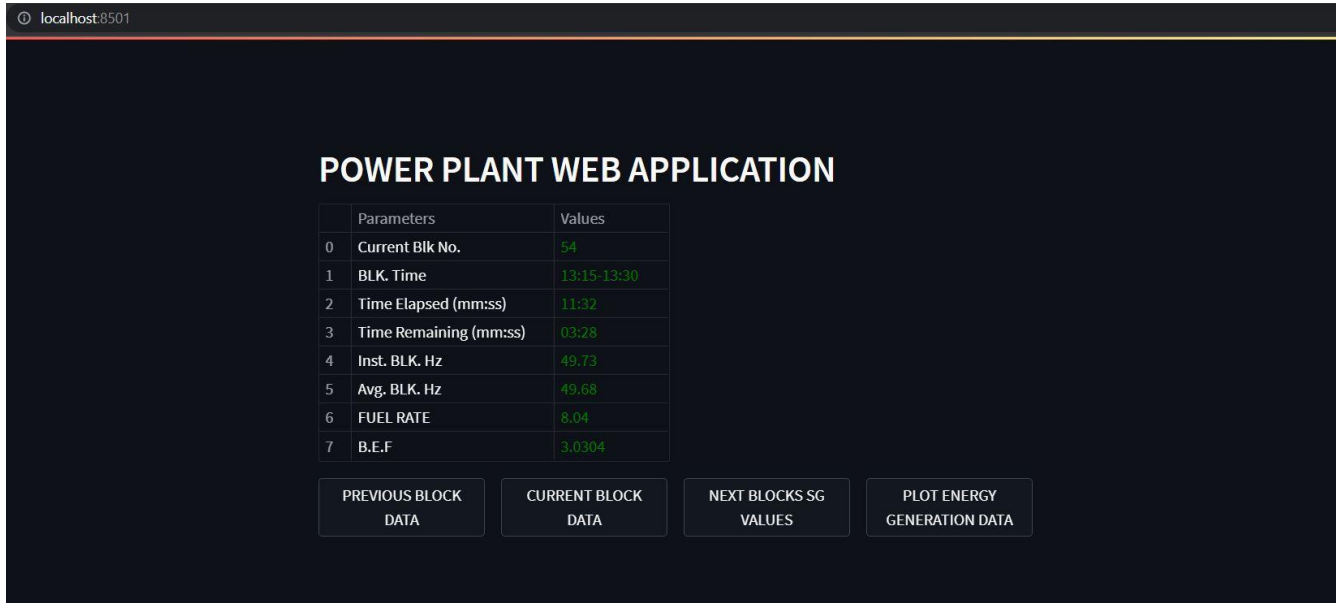
Day 2 Prediction



Demonstration of the App ([Github Repo link here](#))

As the user enters into the Powerplant webapp, the previously discussed Powerplant parameters are displayed along with the buttons to view:

- **Previous Block Data**
- **Current Block Data**
- **Next Blocks SG values**
- **Plot of Actual and Predicted Power Generations.**



The screenshot displays the 'POWER PLANT WEB APPLICATION' interface. At the top, the browser address bar shows 'localhost:8501'. The main content area features a table with 8 rows of parameters and their values, followed by four buttons for data navigation.

	Parameters	Values
0	Current Blk No.	54
1	BLK. Time	13:15-13:30
2	Time Elapsed (mm:ss)	11:32
3	Time Remaining (mm:ss)	03:28
4	Inst. BLK. Hz	49.73
5	Avg. BLK. Hz	49.68
6	FUEL RATE	8.04
7	B.E.F	3.0304

Below the table, there are four buttons:

- PREVIOUS BLOCK DATA
- CURRENT BLOCK DATA
- NEXT BLOCKS SG VALUES
- PLOT ENERGY GENERATION DATA

Previous Block Data

This button shows the Powerplant parameters corresponding to the time block previous of the current block.

POWER PLANT WEB APPLICATION

	Parameters	Values
0	Current Blk No.	55
1	BLK. Time	13:30-13:45
2	Time Elapsed (mm:ss)	04:22
3	Time Remaining (mm:ss)	10:38
4	Inst. BLK. Hz	50.92
5	Avg. BLK. Hz	50.870000000000005
6	FUEL RATE	8.04
7	B.E.F	0

PREVIOUS BLOCK
DATA

CURRENT BLOCK
DATA

NEXT BLOCKS SG
VALUES

PLOT ENERGY
GENERATION DATA

PREVIOUS BLOCK DATA

	Parameters	Values
0	Block No.	54
1	Block Time	13:15-13:30
2	DC (MW)	177.16
3	SG (MW)	193.5
4	AG/SG %	84.78
5	Avg. Hz	49.73
6	Dev. MW	-29.46
7	Dev. Rate	0
8	Dev. (Rs.)	3.0304
9	Addi. Dev.(Rs.)	0

	Parameters	Values
0	Block No.	54
1	Block Time	13:15-13:30
2	DC (MW)	177.16
3	SG (MW)	193.5
4	AG/SG %	84.78
5	Avg. Hz	49.73
6	Dev. MW	-29.46
7	Dev. Rate	0
8	Dev. (Rs.)	3.0304
9	Addi. Dev.(Rs.)	0
10	Total Dev.	-44637.8
11	Fuel Cost(Rs.)	11150.61
12	Net Gain(Rs.)	-33487.19

Current Block Data

This button shows the Powerplant parameters corresponding to the current time block.

POWER PLANT WEB APPLICATION		
	Parameters	Values
0	Current Blk No.	55
1	BLK. Time	13:30-13:45
2	Time Elapsed (mm:ss)	11:48
3	Time Remaining (mm:ss)	03:12
4	Inst. BLK. Hz	50.92
5	Avg. BLK. Hz	50.870000000000005
6	FUEL RATE	8.04
7	B.E.F	0

PREVIOUS BLOCK DATA

CURRENT BLOCK DATA

NEXT BLOCKS SG VALUES

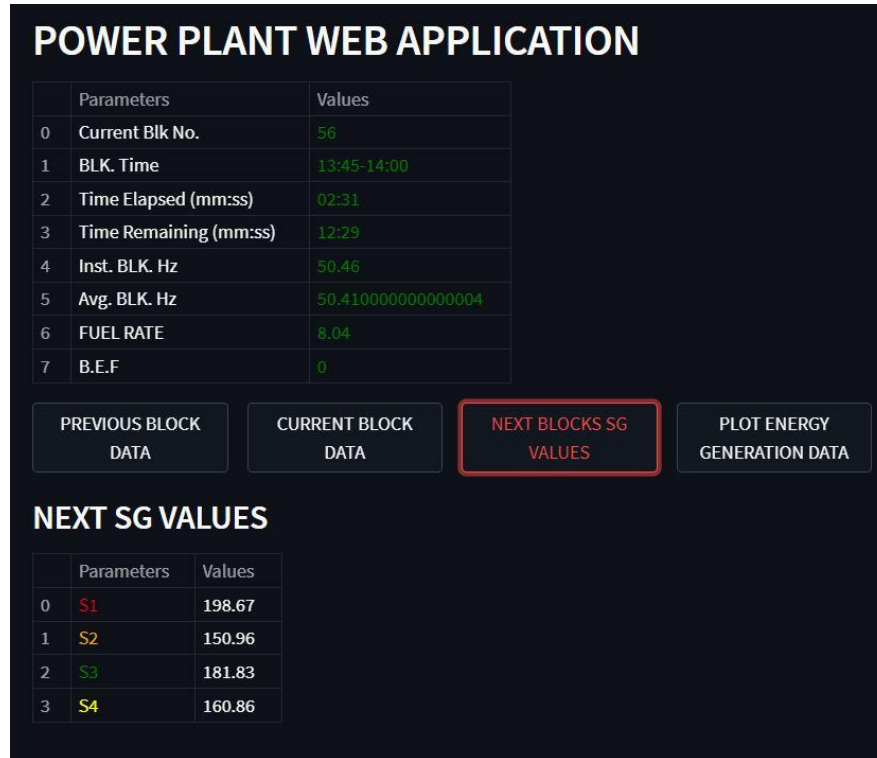
PLOT ENERGY GENERATION DATA

CURRENT BLOCK DATA		
	Parameters	Values
0	Block No.	55
1	Block Time	13:30-13:45
2	DC (MW)	181.43
3	SG (MW)	176.62
4	AG/SG %	109.31
5	Avg. Hz	50.92
6	Dev. MW	16.45
7	Dev. Rate	0
8	Dev. (Rs.)	0
9	Addi. Dev.(Rs.)	0

	Parameters	Values
0	Block No.	55
1	Block Time	13:30-13:45
2	DC (MW)	181.43
3	SG (MW)	176.62
4	AG/SG %	109.31
5	Avg. Hz	50.92
6	Dev. MW	16.45
7	Dev. Rate	0
8	Dev. (Rs.)	0
9	Addi. Dev.(Rs.)	0
10	Total Dev.	0.0
11	Fuel Cost(Rs.)	-6226.32
12	Net Gain(Rs.)	-6226.32
13	Cont. +ve blocks	1
14	Cont. -ve blocks	0
15	Sign Violations	0

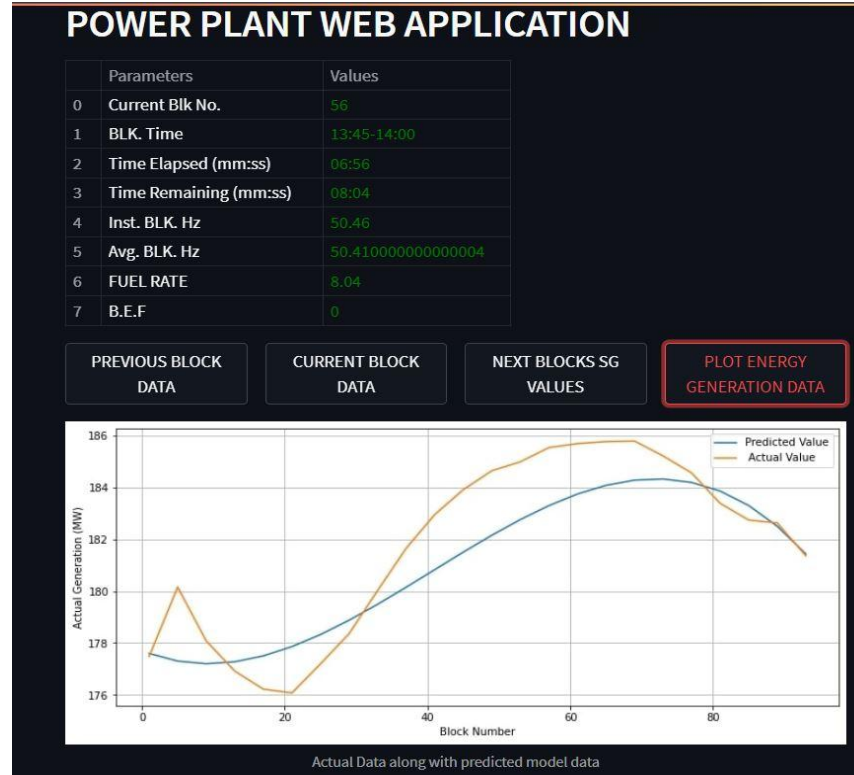
Next Blocks SG Values

This button shows the **Scheduled Generation** (SG) Values (in MW) of the Powerplant corresponding to next 4 time blocks.



Graph:

On clicking plot Energy data button, we get the graph for the actual value and prediction model as follows:



References

Constraints and protocols for calculations of various parameters

- Letter to Generators SCED (Dated 18 April 2019) - POSOCO
- DSM 5th Amendment (Dated 28 May 2019) - Central Electricity Regulatory Commission
- Notification 132 (Dated 6 January 2014) - Central Electricity Regulatory Commission

Polynomial Regression Model

- Eva Ostertagová, Modelling using Polynomial Regression, Procedia Engineering, Volume 48, 12012, Pages 500-506, ISSN 1877-7058, <https://doi.org/10.1016/j.proeng.2012.09.545>.