SUGAR DB

Authors: Carter Weaver, Alexander Penney, Yuchen Dai
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A microgrid energy management system to economically optimize the generator and ESS dispatch over a 24hr horizon given predicted renewable generation and dynamic loads based on weather data. Includes a web application to input custom microgrid architectures and generation capabilities, and output optimization schedules, costs, and power flow visualizations.**

***Index Terms*—Microgrid, Optimization, Renewables, Machine Learning**

1 INTRODUCTION

With the growing use of renewable energy sources, there's a need for better ways to efficiently simulate power flow in microgrids, like the one in Figure 1. Microgrid developers are interested in knowing how their grid will perform from a feasibility and economic standpoint. With the addition of stochastic wind and solar generation, optimal control of a microgrid requires accurate and reliable forecasting. Existing commercial Optimal Power Flow (OPF) tools are primarily designed for stable, large-scale distribution networks without the intricacies of batteries and renewables. Furthermore, traditional microgrid sources such as diesel generators are problematic due their environmental issues and ramping constraints. Therefore an energy storage system (ESS) is often included to provide real-time power injections to balance supply and demand and perform energy arbitrage. An accessible web-based management application will allow the developer to simulate the operation of their customized grid at the 1-hour scale using an optimized economic dispatch algorithm and a generation and load forecasting tool that will give them rough estimates of economics and stability.

2 USE-CASE REQUIREMENTS

The Sugar-DB simulation tool satisfies a specific set of requirements needed by microgrid planners and controllers, with each requirement linked to a specific design requirement:

1. Allow users to provide their own microgrid architecture with ESS and renewable generation via GridLAB-D design files and specify any US location for their microgrid simulation to take place.

2. Accurately forecast power generation for any solar and wind power sources in the grid as well as demand for any loads in the grid. Make predictions for the full day ahead with a 1-hour rolling dispatch interval.

3. Solve a multi-period OPF problem for the defined grid architecture over a 24-hour horizon at 1-hour dispatch intervals to produce the optimal energy storage and generation dispatch.

4. In the web interface, visualize power flows, energy forecasts, and other relevant statistics related to the user's simulated microgrid.

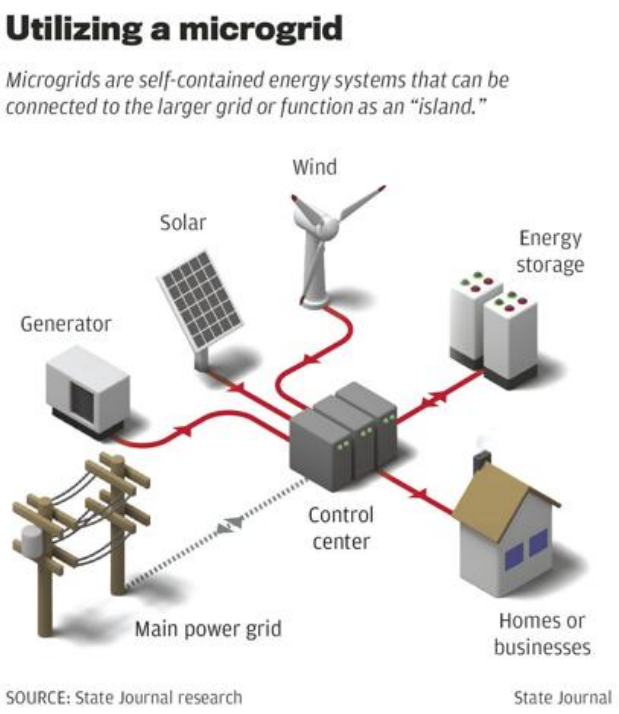

Figure 1: Overview of nodes in a standard microgrid [1]

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

Here we present the full system architecture for our application. At the foundation, we have a suite of prediction models that draw from OpenWeatherMap API, incorporating variables such as temperature, wind speed, and humidity to accurately model solar, wind, and load forecasts. The generation and demand predictions then feed into the optimizer and produce information such as the grid states and dispatch schedules.

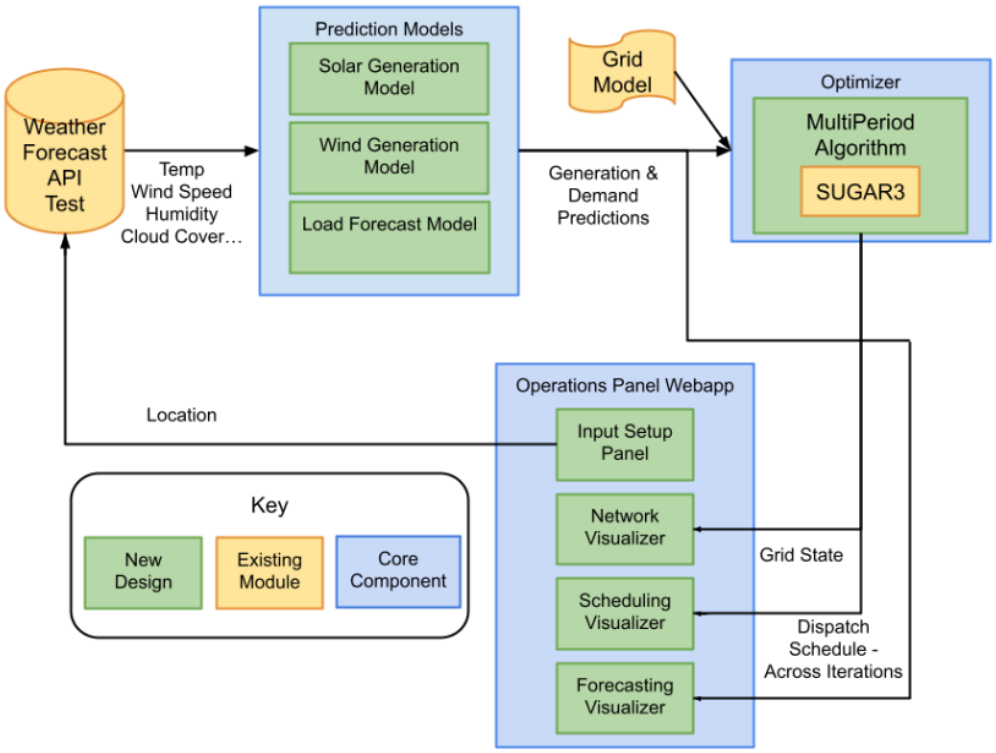

Figure 2: Block Diagram for the System Architecture

On the application side, the Operational Panel Webapp is the user's gateway, featuring an Input Setup Panel that allows for the specification of the location and the upload of the user's customized microgrid design. The Network Visualizer then graphically displays the microgrid layout, providing clarity on node connections, paths, and types, as well as real-time power flows. Additionally, we incorporate a Scheduling Visualizer and Forecasting Visualizer, offering users a granular view of their microgrid's operation schedule and energy forecasts. [Fig. 2]

4 DESIGN REQUIREMENTS

Design requirements for each of the three core components are divided into **Functional Requirements** which define correct behavior and **Performance Requirements** which define the tolerable speed and error of processes.

4.1 Energy Forecasting

Functional Requirements

- **Weather API** - Access real-world, hourly weather forecasts within 1 second (Use Case Requirement 2)
- **Model Features** - Predict renewable power generation and demand (% of capacity) at a specific timestamp using only weather features accessible through a weather forecasting API (Use Case Requirement 2)

Performance Requirements

- **Model Error** - All models achieve better performance than naive approach and comparable accuracy to our goals in 7 (Use Case Requirement 2)
- **Model Speed** - New predictions are generated in less than 10 seconds (Use Case Requirement 2)

4.2 Optimization

Functional Requirements

- **Correct Three Phase Linear Battery Model** - battery state variables meet equality and inequality constraints that define this model at each time step (Use Case Requirement 3)
- **Distribution Device Models** - power flow solver supports the following devices commonly found in distribution systems:
  - Capacitors
  - Fuses
  - Synchronous Generators
  - PQ Loads
  - Reactors
  - Shunts
  - Switches
  - Three Phase Transformer
  - Pi-Model Lines

- **Robust Convergence** - multi-period optimization problem converges to a local optimum for all 3-phase IEEE test cases (Use Case Requirement 3)

Performance Requirements

- **Fast Convergence** - multi-period optimization problem converges within 15 minutes on Lenovo Thinkpad with 8 × Intel® Core™ i5-8250U CPU @ 1.60GHz. (Use Case Requirement 3)
- **Accurate Power Flow** - single state powerflow solutions are within 0.2% error of verified GridLab-D solutions. (Use Case Requirement 3)

4.3 WebApp

Functional

- **Customizable Microgrid Architecture Design** - take in the user's customized GridLAB -D microgrid design via file upload (Use Case Requirement 1)
- **Battery Charge Profile Visualization** - real-time power flow predictions on the user's customized microgrid design with a clean, reasonable, and informative visualization (Use Case Requirement 4)
- **Forecasted Generation and Load Visualization** - Visualize forecast, load, and dispatch (Use Case Requirement 4)

Performance

- **Response Time** - under 1 second

5 DESIGN TRADE STUDIES

5.1 Forecasting Alternatives

- **Simple Statistical Methods:**

In our design process, we considered the tradeoffs between employing ML models and simpler statistical methods for energy forecasting. Simple statistical methods, such as moving averages and exponential smoothing, offer straightforward implementations, are computationally efficient, and are easy to interpret. However, these methods rely heavily on recently observed data related to the specific scenario of interest in order to output accurate forecasts, and real-world feedback is not available within our simulation. In addition, they tend to struggle to capture complex relationships and nonlinear trends present in energy generation data. [2]

On the other hand, ML models, such as Random Forest regression models, offer a more sophisticated approach to forecasting. By training on historical data, Ensemble ML methods have the capability to capture intricate patterns and dependencies in the data, making them potentially more accurate for predicting energy generation. [3]

- **Physics-Based Methods:**

Physics-based models offer an alternative approach to energy forecasting, relying on fundamental principles and physical laws governing energy generation processes. One example is the Weather Research and Forecasting (WRF) model: a widely used mesoscale numerical weather prediction system, which simulates atmospheric processes based on fundamental physical principles.[4] Physics-based models can provide valuable insights into the underlying mechanisms driving energy generation, allowing for more interpretable and explainable forecasts. However, they require extensive domain knowledge and computational resources to develop and calibrate, limiting their practical applicability in the range of scenarios we intend to capture. For our purposes, more generalizable and less time-consuming options, such as random forest regressors, are preferred.

5.2 Optimization Alternatives

- **ML and Genetic Algorithm Methods:**

The approach taken in this project follows methodology used in optimization solving through the setup of a Lagrange function and a newton-raphson solver to find the zeros of the corresponding Karush-Kuhn-Tucker conditions. There are entirely different ways of finding optimal optimization trajectories, however, including ML models and genetic algorithms. ML-based models for multi-period energy dispatch offer ease of formulation and implementation but suffer from lack of interpretability due to the black box effect. Genetic algorithms offer an effective approach towards trajectory optimization but are limited by their lack of physics modeling [5]. Therefore, we chose to use a well defined optimization problem that incorporates physical grid constraints.

- **Generic Optimization Problem Solvers**

There exist a plethora of general purpose optimization solvers that can theoretically find the DC optimal power flow for a microgrid because it is a linear program. Some of the most popular include the Matlab FMINCON tool (a Mixed-Integer Linear Program solver), CPLEX, and Gurobi. The AC-OPF formulation used in this project, as described in Section 6.2.1 includes non-linear constraints which result in a non-convex optimization problem that cannot be solved using these tools. There exist many proprietary AC-OPF solvers that can handle these constraints such as PowerWorld, PSSE, and ETAP but they are impossible to adapt into a multi-period AC-OPF problem because they are closed-source. Therefore we chose to use the SUGAR3 AC-OPF solver because it is scalable, accessible through a Python implementation, and robust to various initial conditions. In addition, because SUGAR3 is developed at CMU we can enable close collaboration in our multi-period extension to the framework.

6 SYSTEM IMPLEMENTATION

6.1 Energy Forecasting Models

This section will highlight the data, methods, and tech stack associated with our energy forecasting design plans. The goal of our forecasting efforts is to train models that can accurately predict wind and solar energy generation as well as energy demand in any location in the U.S. These predictions will be leveraged by our optimization framework as inputs to the multi-period solver.

6.1.1 Data

For the forecasting component of our application, The strength of our predictive models relies upon the accuracy and relevance of the available data. As such, a diverse range of data sources must be leveraged to meet our requirements. Before model deployment, historical weather and power generation/consumption datasets will be used for training and evaluation:

Solar Data

- **Source** - 2022 dataset downloaded from the National Solar Radiation Database combined with cloud coverage data from the OpenWeatherMap API.[6, 7]
- **Location** - Pittsburgh, PA
- **Characteristics**
  - Features: Month, Day, Hour, Temp, Humidity, Pressure, Cloud Cover
  - Target: Global Horizontal Irradiance (GHI) converted to photovoltaic (PV) power (% of capacity)
  - 8760 data points

Wind Data

- **Source** - 2018 SCADA wind turbine dataset downloaded from Kaggle.[8]
- **Location** - Yalova, Turkey
- **Characteristics**
  - Features: Month, Day, Hour, Wind Speed, Wind Direction, Pressure
  - Target: Wind Power (% of capacity)
  - 5053 data points

Load Data

- **Source** - 2013 collection of datasets measuring electric load in 21 households, downloaded from the University of Strathclyde, Glasgow.[9]
- **Location** - United Kingdom
- **Characteristics** - 3 input features, 1 target variable, 5053 data points.
  - Features: Month, Day, Hour, Temp, Humidity, Pressure, Cloud Cover
  - Target: Electrical Demand (% of nominal)
  - 8760 data points

These datasets are pre processed by normalizing dates, times, and power values, imputing missing values, and aggregating features. The model outputs are scaled between 0 and 1 in addition to each model's post processing (i.e. weighting solar outputs by time between sunrise and sunset). Following training, validation, and integration with our application, our models can take in future hourly weather forecasts that mirror our training features and predict each target variable for the day ahead, in 1 hour intervals.

6.1.2 Baseline Model

In order to help evaluate the performance of our final models, we are using linear regression as a naive approach, given its simplicity and interpretability. The resulting error metrics act as baseline values and the coefficients given by each model indicate the importance of different features in predicting the target variable, which we can use for fine-tuning later models.

6.1.3 Final Model

Our initial plan for the final forecasting pipelines included LSTMs as the primary model, due to their ability to capture trends in time series data, however these ended up being uneffective for our purposes, since we don't have observed ground truth from every location to learn from. Instead, we used an AutoML tool developed at the Auton Lab at CMU, which generated rankings of 10 potential model choices based on their performance on each of our datasets. From experimentation with these options, the best metrics were achieved by Random Forest, a method that is particularly effective for handling complex datasets with non-linear relationships and interacting variables, which are characteristic of energy data. A full diagram of our final pipeline is shown in 13.[10, 11, 12]

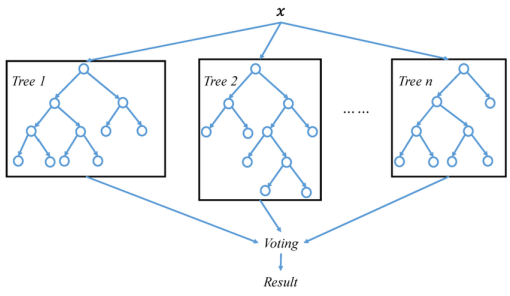

Figure 3: General Architecture of Random Forest [13]

A diagram of the basic structure of a random forest regressor (or classifier) can be found in 3, highlighting the aggregation of multiple decision trees to form a more accurate

and stable prediction. This configuration allows the model to leverage diverse data splits, ensuring each tree in the forest considers a random subset of features and data points, which enhances the generalization of the model. The final decision at each node is made based on the majority vote or average of predictions, providing a balanced insight into the predictive trends captured across the individual trees. This process not only improves prediction accuracy but also gives insights into feature importance, helping us to refine the models for forecasting tasks.

6.1.4 Solar Conversions

Due to limitations in data availability and the direct relationship between irradiance and PV power, our prediction model for solar panels first predicts GHI (% of standard GHI) from weather features before being converted to generated power (% of capacity) via equation 1. P_{PV} is output power, P_{Peak} is the panel's rater power, and T_c is the panel's cell temperature.[14]

$$P_{PV}(t) = P_{Peak}(\frac{G(t)}{G_{standard}}) - \alpha_T[T_c(t) - T_{standard}] \quad (1)$$

6.1.5 Libraries and Frameworks

The code for building, training, and evaluating our ML models was written in Python, along with the following software tools:

- Pandas: For data manipulation and preprocessing tasks.
- NumPy: For dataset array operations and mathematical computations.
- Scikit-learn: For our baseline models as well as for preprocessing, model selection, and evaluation.
- TensorFlow: For building and training LSTM models, providing a high-level API for neural network construction.
- Requests: For making HTTP requests to online APIs, enabling data retrieval from weather services.

6.2 Multi Period Optimization Solver

6.2.1 Optimization Formulation

To construct our multi-period optimization framework, we extend SUGAR AC-OPF, which optimizes steady-state operation for a single time period to include time-variant energy storage systems.

A. SUGAR3 SUGAR [15] solves AC OPF at the transmission scale using circuit based heuristics to ensure robustness to initial conditions and convergence to a local minima. SUGAR3 extends SUGAR to three phase distribution systems using a similar equivalent circuit analysis framework [16].

Optimizing a single time-period AC-OPF is represented by a Lagrange function with dual and slack variables to represent network equality constraints and device limits respectively, shown in 2

$$\mathcal{L}_{ACOPF} = C_g \|P_g\|_2^2 + w_f \|I_f\|_2^2 + \lambda^T(g(X) + I_f) + \mu^T h(X) \quad (2)$$

Where

- C_g is generator cost ($)
- P_g is synchronous generator real power (MW)
- λ is dual variable vector for equality constraints
- $g(X)$ is vector of non-linear equality constraints given by power flow equations
- μ is vector of slack variables for inequality constraints
- $h(X)$ is vector of inequality constraints given by line and device limits
- I_f is an feasibility current injected at each node to ensure feasibility
- w_f is a large weight to minimize feasibility currents

Figure 4 shows the methodology used in SUGAR3. A multi-period optimization formulation and implementation for a microgrid using SUGAR3 was developed

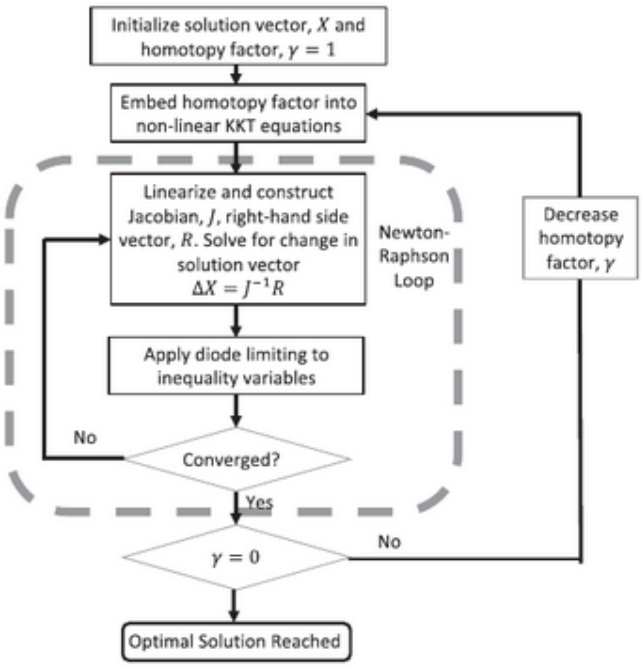

Figure 4: SUGAR3 Single Period AC-OPF Methodology

B. Energy Storage System Model Time variant ESS models are incorporated in the optimization problem with constraints of state of charge (SOC) and maximum charge and discharge rates. ESS are constantly discharging and charging in the microgrid but their behavior can be modeled through average real discharge power P_d and real charge power P_c during a time period Δt. The ESS dynamic charging equation 3 couples the SOC at time t and the SOC at previous time step $t - \Delta t$. The charging and discharging efficiency terms η_c and η_d are between $[0, 1]$. The SOC, B, has maximum and minimum charge limits $\overline{B}$ and $\underline{B}$ as seen in equation 4. The charge and discharge rates are also bounded from above by maximum rates $\overline{P_c}$ and $\overline{P_d}$ shown in equation 5.

$$B^t = B^{t-\Delta t} + \Delta t(\eta_c P_c^t - \eta_d P_d^t) \tag{3}$$

$$\underline{B} \leq B \leq \overline{B} \tag{4}$$

$$P_c \leq \overline{P_c} \qquad P_d \leq \overline{P_d} \tag{5}$$

C. Slack Bus Cost In the power flow method, the slack bus accounts for the mismatch between the rest of the generation and loads, and is set as a reference node with known voltage and angle. The slack bus can be thought of as an infinite source, which is equivalent to the bus connecting a microgrid to the external grid. Therefore the slack power in this simulation is the imported power from the grid. We add the cost of importing external power to the objective function through the square of the power P_g and a cost weight C_g.

Multi Period Formulation

The models for generator ramping constraints and ESS dynamics are inherently time variant, requiring outputs from the previous state at each time period. Realistically optimizing a microgrid with ESS requires correct modeling multiple periods to ensure SOC remains within limits. Therefore a multi-period optimization problem was developed to solve for dispatch at each time period while following time-varying constraints. The following formulation 6 describes the multi-period optimization problem over n periods from time $t = 1$ to t_n with increments of Δt.

$$\begin{aligned}
&\min_X \sum_{t=1}^{t_n} C_g^t (P_g^t)^2 + (C_d^t P_d^t - C_c^t P_c^t) + \|I_f\|_2^2 \\
&s.t. \quad g^t(X^t) = 0 \quad 1 \leq t \leq t_n \\
&h^t(X^t) \leq 0 \quad 1 \leq t \leq t_n \\
&\underline{B} \leq B^t \leq \overline{B} \quad 1 \leq t \leq t_n \\
&B^t = B^{t-\Delta t} + \Delta t(\eta_c P_c^t - \eta_d P_d^t) \\
&P_c^t \leq \overline{P_c} \\
&P_d^t \leq \overline{P_d}
\end{aligned} \tag{6}$$

This multi-period optimization problem minimizes the total cost of imported power and ESS dispatch over all periods while ensuring network feasibility. The Lagrange function $\mathcal{L}_{MP}$ associated with this constrained optimization problem is shown below.

$$\begin{aligned}
\mathcal{L}_{MP} = &\sum_{t=1}^{t_n} C_g^t \left\|P_g^t\right\|_2^2 + (C_d^t P_d^t - C_c^t P_c^t) + \|I_f\|_2^2 \\
&+\lambda_t \cdot g^t(X^t) + \mu_t \cdot \tilde{h}^t(X^t) \\
&+\lambda_{Bt} \cdot \left(B^t - B^{t-\Delta t} - \Delta t(\eta_c P_c^t - \eta_d P_d^t)\right)
\end{aligned} \tag{7}$$

The inequality constraint function $\tilde{h}^t(X^t)$ in equation 7 incorporates the ESS inequality constraints (4, 5). Notably, the Lagrange function $\mathcal{L}_{MP}$ is the sum of the SUGAR3 Lagrange functions 2 over all time periods with the added coupling terms from the ESS.

A. Differential Dynamic Programing To solve this optimization problem, a grid state at each time period must be found that satisfies the KKT conditions. This is accomplished through differential dynamic programming (DDP). DDP simulates the grid dynamics over multiple periods to calculate a total cost through a forward pass, then updates the coupling terms based on results from the forward pass. During a forward DDP pass, the algorithm feeds forward variables from the previous time step and sets future variables constant. At convergence, the backward pass does not affect the next forward pass, indicating that the algorithm reached a minimum.

B. Using Forecasted Generation and Loads The forecasted generation and load outputs from the machine learning model are used as inputs to the multiperiod optimization problem at each period. At a time t, the machine learning model outputs generation capacity factors c_w^t and c_s^t for wind and solar generation and a load factor LF^t. c_w^t and c_s^t will be multiplied by the nameplate real power capacity of the generators defined in the grid model and the corresponding power within the grid state X^t will be updated. LF^t will adjust all loads within the grid state X^t.

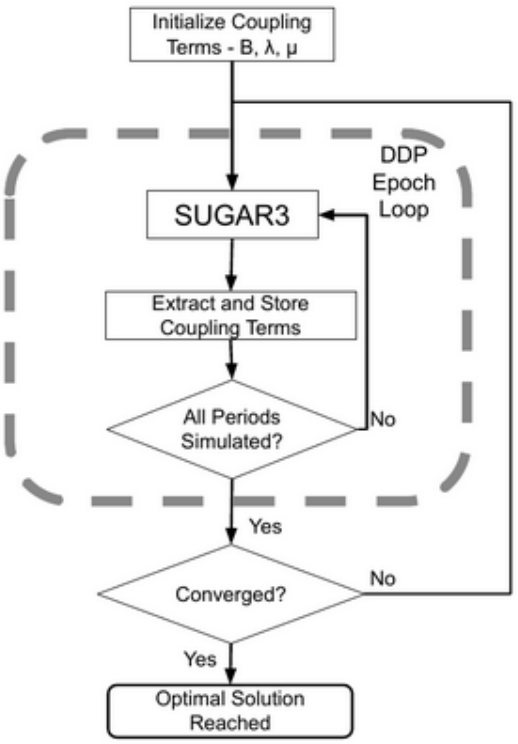

Figure 5: Multi Period AC-OPF Methodology

6.3 Web App

For our web application's backend, we've chosen the Django Web Framework for its rapid development, fully-featured functionalities, security, and scalability. Django promotes reusability to reduce the amount of redundant code and make the development process faster by providing many ready-to-use components for common web development tasks. It has features that cover many needs of web development such as authentication, URL routing, template engine, object-relational mapper, etc. It also focuses on security, providing built-in protection against many common vulnerabilities by default. Furthermore, Django uses a component-based architecture, which makes it easy to scale the application up or down as needed and handle high traffic volumes.

The frontend is crafted with pure CSS/HTML enhanced with Vis.js for dynamic, interactive visualizations. A key feature is using a parser, which transforms complex microgrid data into a user-friendly readable format, enabling detailed visualizations of network connections, power flows, and forecasting data.

Finally, the web app will be deployed on AWS EC2 for its reliability and accessibility. The AWS EC2 allows scaling the compute capacity up or down automatically according to the application's needs, providing flexibility for handling varying traffic loads efficiently, and ensuring that the application is responsive even during peak times. Also, AWS provides a highly reliable environment where replacement instances can be rapidly and predictably commissioned. Additionally, with EC2, we pay only for the compute time you consume, which can further optimize the cost.

Currently, we plan to run the machine learning model and optimizer locally. However, if running locally will affect the overall performance, we'll deploy the machine learning model on a platform such as AmazonSagemaker and transmit data back and forth through Websockets.

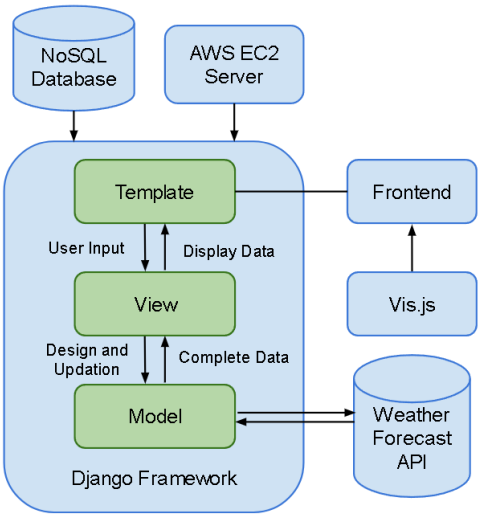

Figure 6: Web App Pipeline

7 TEST & VALIDATION

7.1 Forecasting Model Evaluation

For the machine learning component of our design, we employed a comprehensive testing strategy to evaluate the effectiveness and reliability of our forecasting models.

7.1.1 Baseline Model

We used linear regression as a naive forecasting approach to compare to our final models, given its simplicity and interpretability. Across the three fitted linear regression models, The average R^2 score was 0.34 and the average NRMSE score was 31%. Both of these baseline metrics were surpassed by all three of our final pipelines.

7.1.2 Performance and Timing Metrics

In addition to exceeding the metrics of our baseline models, we also achieved our performance goal of normalized root mean squared error (NRMSE) $< 20\%$, which is based on prior research in the field of energy forecasting using machine learning methods:

Performance Metrics	Goal	Actual
Wind Power	< 20% NRMSE	9.9% NRMSE
Solar Power	< 20% NRMSE	8.6% NRMSE
Load	< 20% NRMSE	11.8% NRMSE

Figure 7: NRMSE Scores for Forecasting Models

As for the timing requirements set in our design review, our final pipeline easily meets our goals, running far faster than expected:

Timing	Goal	Actual
Train Time	minutes	< 10 seconds
Predict Time	seconds	< 5 seconds

Figure 8: Timing Metrics for Forecasting Models

7.1.3 Model Behavior Across Weather Conditions

To verify the behavior of our ML models, we plotted averages of our predictions on test data against the ground truth values and compared them with global trends. Figure 9 shows that our solar power predictions match the expected real world behavior since power is only generated at daylgith hours and generation increases with temperature in general.

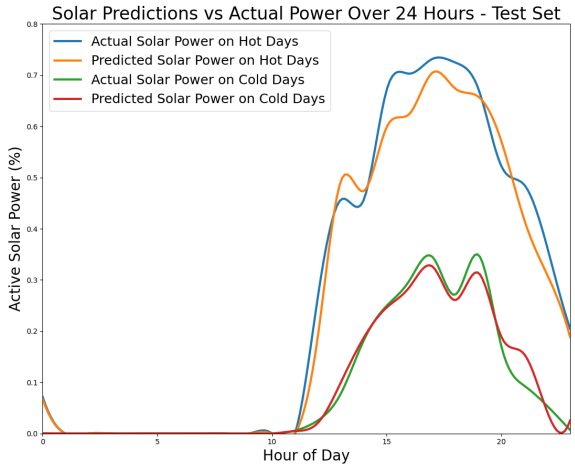

Figure 9: Average Solar Power on Hot vs Cold Days

Figure 10 shows that our load predictions match the expected real world behavior since demand peaks in the mornings and evenings, when people are at home using appliances. It also shows a slight increase in cold weather due to heating requirements

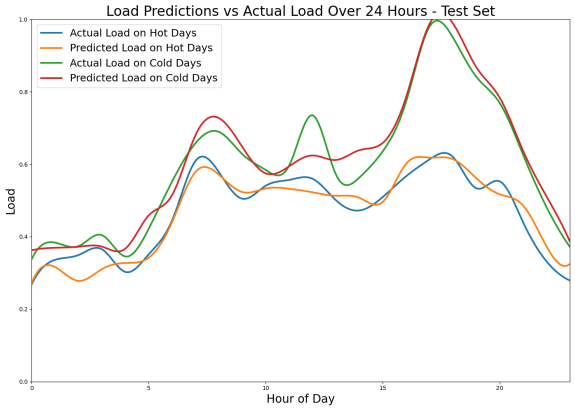

Figure 10: Average Load on Hot vs Cold Days

Figure 9 shows that our wind power predictions match the expected real world behavior since power increases with wind speed in direct coordination with the test ground truth.

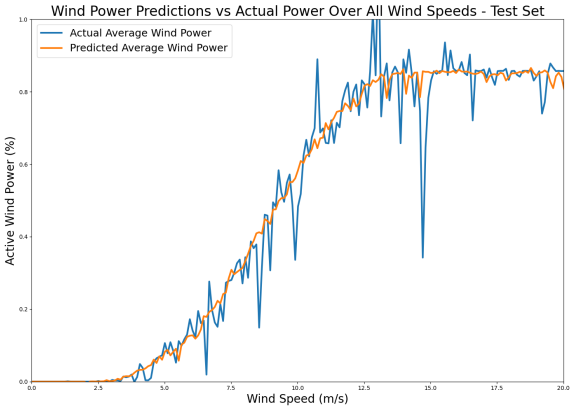

Figure 11: Average Wind Power vs Wind Speed

7.2 Optimization Algorithm Validation

To ensure that the optimization algorithm generates cost-effective and feasible dispatch sequences within the required time frame, we will perform various correctness and performance tests.

7.2.1 Functional Testing

To ensure that modifications to the SUGAR3 optimization framework still produce correct results, the steady-state solution of each time period was compared to the solution of a verified AC-OPF solver - GridLabD [17]. The SUGAR3 solutions all showed a maximum error no larger than 0.2% which is the tolerance specified by the developers of SUGAR3. This test verified that powerflow was solved correctly.

To verify correct battery behavior, the residuals of the battery equality and inequality constraints during all multi-period testcases for the 2 bus microgrid reference model

were less than 1e-20. This verifies that the battery follows the constraints imposed by our optimization problem.

An example optimized battery dispatch is shown below in 12. The two plots contain the average discharge power and the cost importing power for each hour in a 24 hour simulation. As expected, as the price of power increases, the battery discharges more to minimize the amount of imported power. Each colored line represents the battery dispatch for a single epoch, with epoch 0 in purple and subsequent epochs of the DDP loop in colors moving towards yellow on the rainbow. This shows how DDP converges in only a few epochs to an optimal solution.

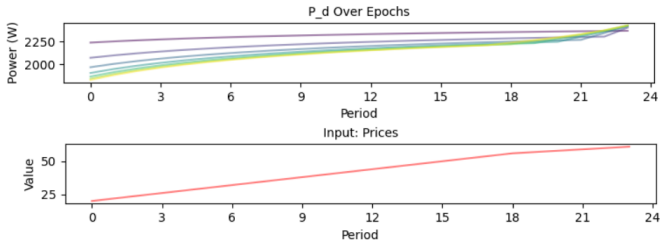

Figure 12: Battery Dispatch and Slack Costs

7.2.2 Performance Testing

To ensure sufficiently fast convergence time to meet the 1 hour dispatch requirement, the multi-period optimization solver was tested on a suite of realistic distribution network test cases from IEEE. These include a 13 bus case, an 4 bus case, and a 2 bus case, each with a single battery. For each of these testcases, the multiperiod optimizer converged within less than 20 minutes.

7.3 Web App

Our web app underwent a series of testing suites to ensure a seamless user experience.

- **Unit Testing** - confirm each component's functionality.
- **Integration Testing** - ensure each component works harmoniously.
- **User Integration Testing** - check if the UI is intuitive and visually appealing.
- **Compatibility Testing** - verify our app functions across browsers like Chrome, Firebox, and Safari.
- **Usability Testing** - gather feedback from real users to evaluate the ease of use, intuitiveness, and overall experience through task success rate, time on task, and user satisfaction ratings.

8 PROJECT MANAGEMENT

8.1 Schedule

The schedule is shown in Fig. 15.

8.2 Team Member Responsibilities

Responsibilities for this project are split between the three components of the application: ML forecasting, optimization, and web app development. Carter is focused primarily on training and deploying the forecasting models, Alby is focused on the optimization framework, and Yuchen is focused on creating and designing the web interface. Despite this division of workload, we collaborated on many tasks, especially on the final integration with the webapp.

8.3 Bill of Materials and Budget

We plan used a general purpose cloud computing ec2 instance from AWS. The final cost of the AWS instance was around $94 in fees.

8.4 Risk Management

8.4.1 Data Quality and Diversity

- **Risk** - Insufficient or poor-quality data may hinder the accuracy of our forecasting models, especially if they do not capture a range of locations and conditions.
- **Mitigation** - We conducted data preprocessing and cleaning to address missing values, outliers, and inconsistencies.

8.4.2 Model Complexity and Performance

- **Risk** - Overly complex models may lead to computational inefficiencies or overfitting, while overly simplistic models may result in poor predictive performance.
- **Mitigation** - We performed feature selection and dimensionality reduction techniques to streamline model complexity. We then evaluated our model performance with cross-validation to ensure robustness and generalizability.

8.4.3 External Dependencies

- **Risk** - Dependence on third-party APIs, libraries, or tools may introduce disruptions to our project workflow and data access if a tool fails.
- **Mitigation** - We identified alternative APIs and libraries as backups in case of service interruptions or changes in third-party providers, but didn't need to use them during the project demo.

9 RELATED WORK

- **SUGAR 3** - Regarding relevant projects, SUGAR3 lays the groundwork for our project. The SUGAR3 optimization methodology is an extension of the SUGAR framework, designed specifically for solving the Alternating Current Optimal Power Flow (AC-OPF) problem at the transmission scale. The original SUGAR methodology utilizes circuit-based heuristics to achieve robustness against initial conditions and to ensure convergence towards a local minimum. SUGAR3 builds upon this by applying a similar equivalent circuit analysis framework to three-phase distribution systems.

  In the context of optimizing a single time-period AC-OPF, SUGAR3 employs a Lagrange function that incorporates both dual and slack variables. These variables are crucial for representing network equality constraints (which ensure that the power flow equations are satisfied) and device limits (which enforce the physical and operational limitations of the devices in the network), respectively. This approach allows for a systematic and efficient optimization of power flow within a three-phase distribution system, ensuring that solutions are both feasible and optimized within the defined constraints.

- **GLM Plotter** - It is an interactive visualization app for power system networks described in Gridlab-D Model files. This open-source app uses Flask as the back end and D3's force layout as the front end. It serves as a great reference for visualizing the microgrid. The GitHub repository can be viewed through the following link: https://github.com/jdechalendar/glm-plotter.

- **Renewables.ninja** - It is a website that runs simulations of hourly power output from wind and solar PV farms by clicking anywhere on the map, which is a great source of design reference for web apps. The website can be viewed through the following link:https://www.renewables.ninja/.

10 SUMMARY

Our design to create an application for simulating the behavior of custom microgrid architectures allows microgrid developers to estimate costs of their systems using an intelligent battery dispatch algorithm as well as visualize power flows and energy forecasts. By allowing users to input their custom microgrid architecture, including energy storage systems (ESS) and renewable generation sources, our application offers a comprehensive platform for efficient energy management. The impact of our design extends to anyone interested in the feasibility and economic performance of microgrids, providing them with the tools to simulate and optimize their grids at a 1-hour scale and make informed decisions about their operation. Challenges in implementing this design included ensuring the accuracy and reliability of our forecasting models as well as meeting the ambitious requirements we set for our optimization and visualization features. Additionally, integrating the web interface with external tools and APIs presented technical hurdles that required significant time to debug.

An important lesson learned was that becoming too siloed in our different component without constantly communicating made integration more difficult as we had to adjust our I/O code to fit everything together.

Future work would include expanding our testcases to larger, more realistic microgrid models, using industry supplied cost weights on battery operation and import power pricing, and comparison of overall system cost with other optimized and non-optimized dispatches.

References

[1] Wisconsin State Journal. *Utilizing a Microgrid*. [Online; accessed February, 2024]. 2019. URL: https://madison.com/utilizing-a-microgrid/image_ee8ffdbb-4e70-5512-bf33-82f57eb1aac1.html.

[2] Tanveer Ahmad, Hongcai Zhang, and Biao Yan. "A review on renewable energy and electricity requirement forecasting models for smart grid and buildings". In: *Sustainable Cities and Society* 55 (2020), p. 102052.

[3] Blessing Olatunde Abisoye, Yanxia Sun, and Wang Zenghui. "A survey of artificial intelligence methods for renewable energy forecasting: Methodologies and insights". In: *Renewable Energy Focus* (2023), p. 100529.

[4] Jordan G Powers et al. "The weather research and forecasting model: Overview, system efforts, and future directions". In: *Bulletin of the American Meteorological Society* 98.8 (2017), pp. 1717–1737.

[5] Stefano Leonori et al. "Optimization strategies for Microgrid energy management systems by Genetic Algorithms". In: *Applied Soft Computing* 86 (2020), p. 105903.

[6] Manajit Sengupta et al. "The national solar radiation data base (NSRDB)". In: *Renewable and sustainable energy reviews* 89 (2018), pp. 51–60.

[7] OpenWeatherMap Ltd. *OpenWeatherMap API*. [Online; accessed February, 2024]. 2012. URL: https://openweathermap.org/api.

[8] Berk Erisen. *Wind Turbine Scada Dataset*. data retrieved from Kaggle. 2018. URL: https://www.kaggle.com/datasets/berkerisen/wind-turbine-scada-dataset.

[9] David Murray et al. "A data management platform for personalised real-time energy feedback". In: *Procededings of the 8th International Conference on Energy Efficiency in Domestic Appliances and Lighting*. 2015.

[10] Germánico López and Pablo Arboleya. "Short-term wind speed forecasting over complex terrain using linear regression models and multivariable LSTM and NARX networks in the Andes Mountains, Ecuador". In: *Renewable Energy* 183 (2022), pp. 351–368.

[11] Hui-Min Zuo et al. "Ten-minute prediction of solar irradiance based on cloud detection and a long short-term memory (LSTM) model". In: *Energy Reports* 8 (2022), pp. 5146–5157.

[12] Xiao Zhou et al. "Wind power forecast based on variational mode decomposition and long short term memory attention network". In: *Energy Reports* 8 (2022), pp. 922–931.

[13] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[14] Hussein A Kazem and Jabar H Yousif. "Comparison of prediction methods of photovoltaic power system production using a measured dataset". In: *Energy conversion and management* 148 (2017), pp. 1070–1081.

[15] Amritanshu Pandey, Aayushya Agarwal, and Larry Pileggi. "Incremental model building homotopy approach for solving exact ac-constrained optimal power flow". In: *arXiv preprint arXiv:2011.00587* (2020).

[16] Marko Jereminov et al. "An equivalent circuit formulation for three-phase power flow analysis of distribution systems". In: *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*. IEEE. 2016, pp. 1–5.

[17] David P Chassin, Kevin Schneider, and Clint Gerkensmeyer. "GridLAB-D: An open-source power systems modeling and simulation environment". In: *2008 IEEE/PES Transmission and Distribution Conference and Exposition*. IEEE. 2008, pp. 1–5.

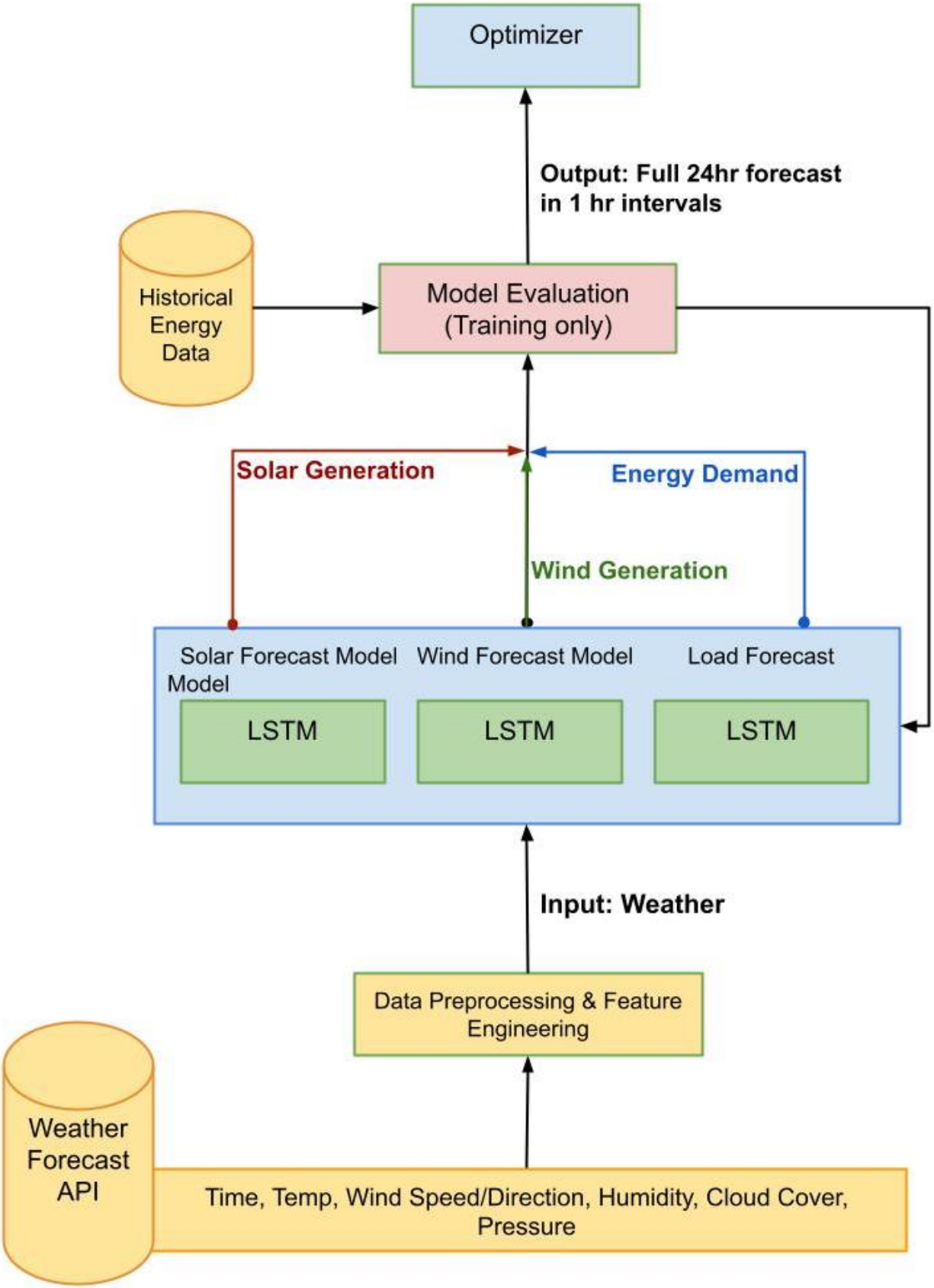

Figure 13: ML Pipeline for Power Generation/Demand Forecasting

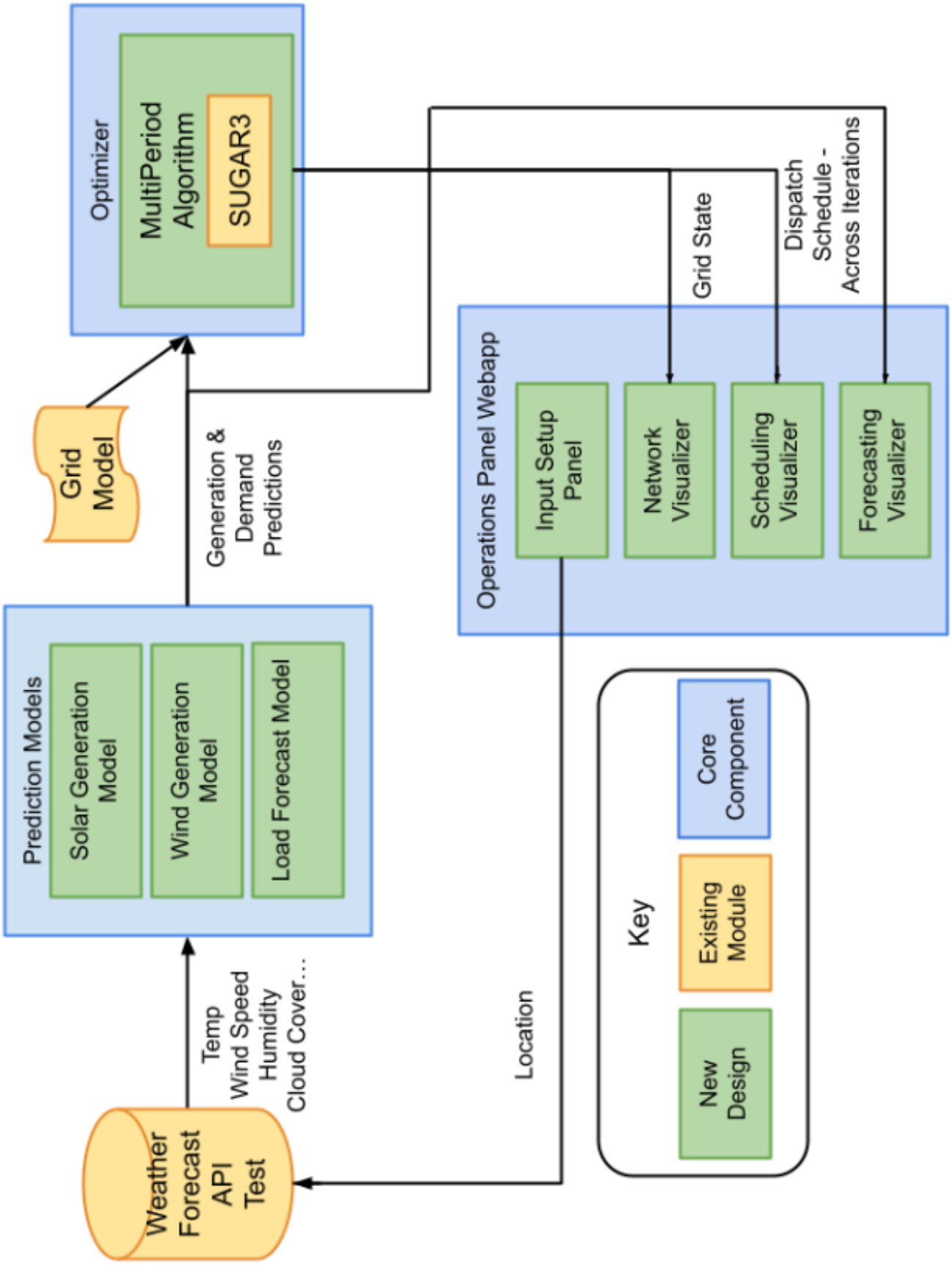

Figure 14: A full-page version of the same system block diagram as depicted earlier.

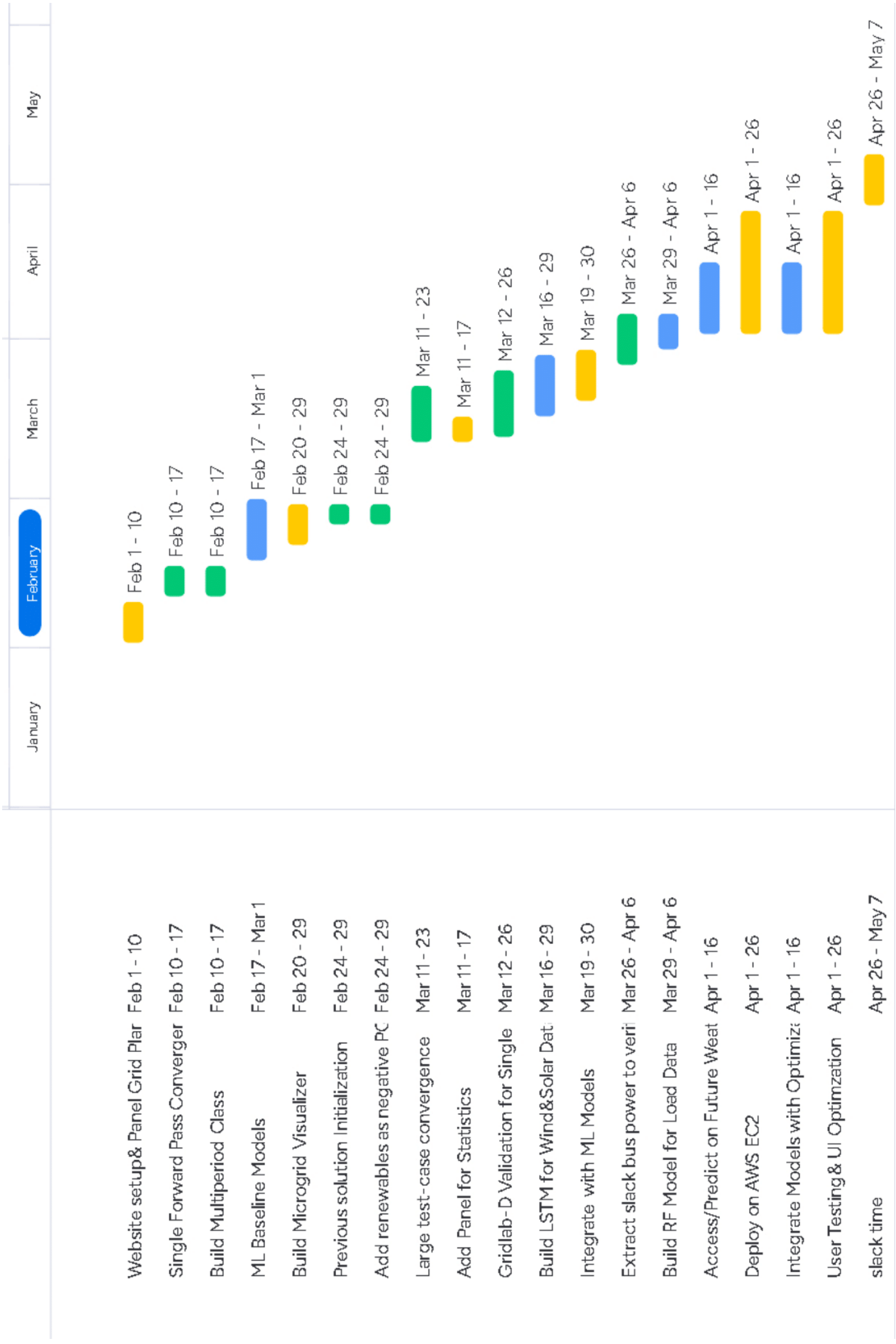

Figure 15: Gantt Chart