Assessing System Reliability of a Solar Wind Energy System using Stochastic Optimization

Romita Biswas

Whiting School of Engineering, Johns Hopkins

ABSTRACT

Introducing a large amount of variable wind and solar generation into an existing electric power system can present significant risk to the reliability of the power grid. The Washington D.C. area has been assessed to analyze the worst case scenario of integrating wind and solar power. This paper explores energy-based probabilistic prediction models to assess the impact of the stochastic characteristics of wind and solar resources on system reliability. Using the poorest performing month the system size is predicted by minimizing the instances peak demand is not met. Reliability is considered for both the largest and smallest possible system. Energy and operational costs are not considered when analyzing the power system.

TABLE OF CONTENTS

1. INTRODUCTION

The Mid-Atlantic region of the United States is predominantly fueled by coal and natural gas. Coal is abundantly found in the region while natural gas prices have lowered over the decade. Cities around the globe have proposed integrating renewable energy resources into the existing power grid including Washington D.C. In order to integrate renewable energy resources, it is critical that the reliability and capability of the system is assessed in relation to the D.C. area. It has been found that the Mid-Atlantic region suffers from variable weather conditions throughout the year presenting a challenge to those wishing to integrate renewable energy. Despite being a clean and abundantly available source, renewable energy, especially wind energy, suffers from lack of energy density and intermittency[3][10]. In this region integrating renewable energy can present a risk in providing continuous power and meeting peak demand. Therefore it is vital that the fluctuating behaviors of these systems are appropriately represented to accurately estimate the size of the hybrid renewable energy system.

In this paper we explore the integration of only solar and wind power. Certain renewable energy systems like hydropower have already been integrated into the existing power grid in Maryland. It is in this paper's interest to capture the sizing needs of solar and wind power integration. In previous years the prior art focused on optimizing renewable energy applications through deterministic approaches[11]. It has been found that the issue with deterministic approaches is it assumes ideal conditions and perfect information, not taking randomness into account. To capture the dynamic behavior of both the renewable energy supply and the energy demand, randomness must be taken into account. Therefore studies including this one are moving towards stochastic optimization where uncertainties and probabilities are considered as inputs and its influence is evaluated on the output of the system. In general, probabilistic reliability techniques are required to model the impacts of wind and solar energy resources on system reliability and capability[11]. The capacity state probability model defined in this paper is ultimately dependent on uncertain weather patterns. Unfortunately the impact of climate change on changing weather patterns has not been taken into account in this paper.

1.1 Washington D.C.

The residential sector of Washington D.C. has been chosen to analyze the capability of integrating wind and solar generation into the current electric power grid. The residents of D.C. consume 11,517,693.05 MWh per year or an average of 1314.81 MWh per hour [8]. The District of Columbia has set a goal to source 80% of its electricity from clean energy by 2030. In order to meet this goal both wind and solar energy must be integrated into the power grid. In this paper it is assumed that each state is responsible for its own power generation, meaning locations in Maryland have only been considered for placement of wind and solar farms.

The locations around D.C. were chosen based on wind speeds and solar irradiance throughout the year. Wind speeds were highest near the coast while solar irradiance was highest near the mountains. Wind turbines generate a significantly higher amount of energy compared to solar panels, while introducing a greater amount of fluctuation. Therefore wind turbines have a higher influence on the reliability of a renewable energy power grid[10]. The assessment was biased towards analyzing the worst case scenario that wind turbines could introduce.

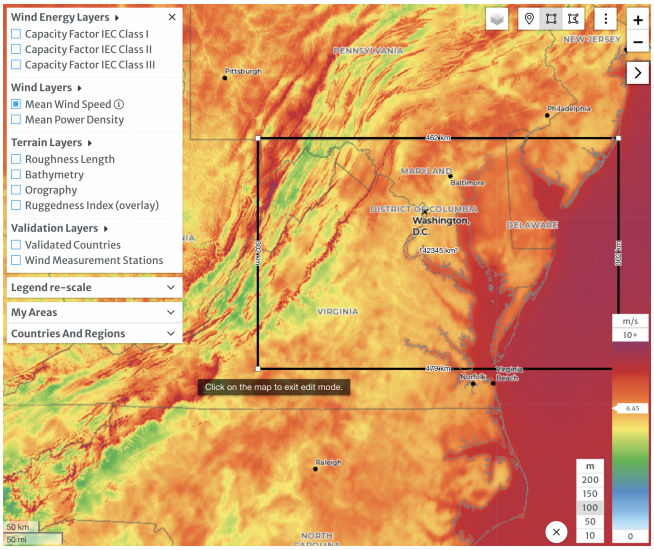

Figure 1: Wind Speed Map of Washington D.C. Area[5]

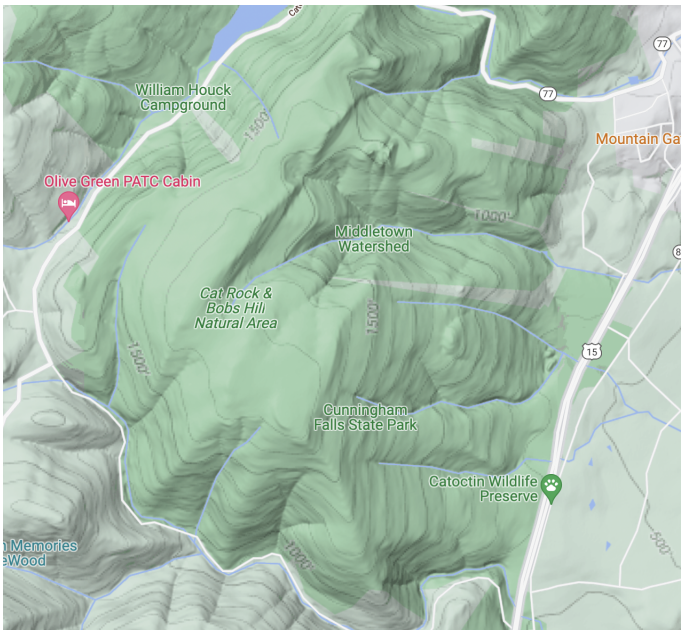

Figure 2: Highest Terrain near Baltimore in Maryland

According to the map, Annapolis has been chosen based on its strong wind profile in Maryland. Annapolis has average wind speeds of 9 m/s throughout the year which is the ideal wind speed for most wind turbines. The Catoctin Mountains have been chosen based on elevation. The higher the elevation the stronger the solar irradiance.

1.2 The Month of August

To analyze the worst case scenario of the renewable energy system only the worst month was assessed. As shown in Figure 3, wind speeds are lowest in August. It is assumed that if the system can support peak

demand at minimum wind speed index, the system should be reliable year round. In contrast to weak wind speeds, solar irradiance is complementary in the month of August as it is one of the best for generating solar power due to high solar elevation and long days.

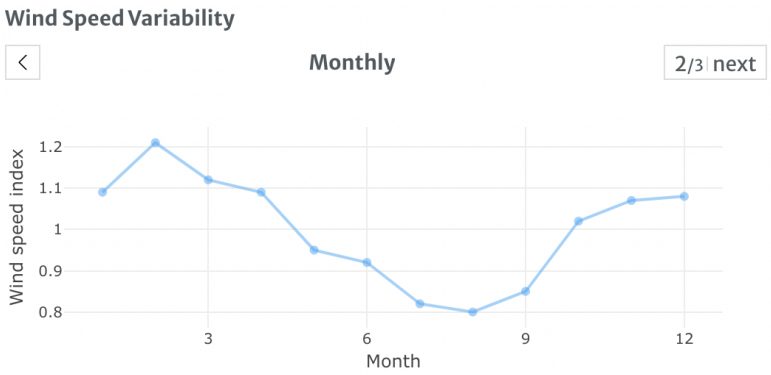

Figure 3: Monthly Wind Speed Variability of Annapolis[5]

Each region in the United States shares a balancing authority in relation to its geographic characteristics. Geographic characteristics determine temperature and humidity of each region which requires different energy demands for its residents. Maryland is part of the Mid-Atlantic region. Figure 4 shows the overall trend of the whole region. It was assumed that Washington D.C. followed the same trend as hourly demand specifically for Washington D.C. was not found.

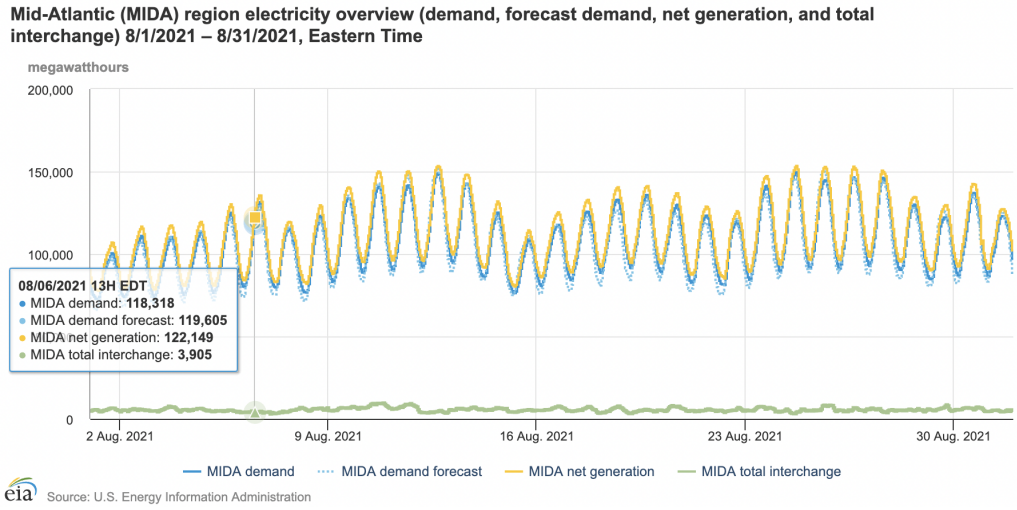

Figure 4: Mid-Atlantic Region electricity overview for August [7]

2 Mathematical Approach

To assess the reliability and capacity of a renewable energy system powered purely by solar and wind, a sequential monte carlo simulation has been performed for each of the resources. Representing the correct uncertainties is critical[11]. To capture the distribution dynamics of each resource it has been attempted to find a close approximation of the uncertainties' true distribution. 5 years of data is used to maintain reliability as weather trends have changed over the decade. Analysis was performed on 5 years of data for

the month of August, where each hour is treated as its own marginal distribution. The marginal distribution of each hour has been used to infer the probability distribution of each day by performing a Monte Carlo simulation. The Monte Carlo simulation generates samples of a given week in August. Both true samples and generated samples are used to train a regression function whose performance was evaluated against true test samples. The best regression function is used to simulate the future weeks in August using a max kernel filter. The future weeks are used to estimate the size of the solar-wind power energy system. The sizing of the system was simulated 50 times and the results from this simulation have been used to assess the reliability of the overall system.

2.1 Basic Concepts of Wind and Solar Generation

2.1.1 Wind Generation

Wind turbines use wind to generate electricity, transforming translational kinetic energy into rotational kinetic energy. Harnessing wind for electricity generation relies heavily on various external environmental aspects, such as season, time of day, and topography. The power generated by the wind turbine is proportional to the cube of the wind speed:

$$P = \frac{\pi}{2} r^2 v^3 \rho\eta$$

Equation 1: Wind Power Generation

where r is the radius of the blades of the turbines, v is the wind speed, ρis the air density, ηis the efficiency of the wind turbine, and P is the generated power. The maximum efficiency of a turbine is 59.3% according to Betz Law of a wind turbine in open flow. Turbines have two types of generation control, stall and pitch, which are used to optimize and protect the turbine. Extremely low speeds are not sufficient for starting the generation while very high speeds may damage the turbine components. There is one speed for triggering the generation called the cut-in speed and another for stopping the generation called the cut-out speed. For speeds that fall outside of the cut-in and cut-out speed, no power is generated.

For this paper the wind turbine chosen was the Gamesa G128-5.0MW whose power curve is shown below in Figure 5[9]. The efficiency of this turbine is rated at 97% meaning $\eta = 0.97 * 59.3\% = 57.52\%$. The diameter of the wind turbines is 128 meters. The air density ρ is 1.224 at sea level.

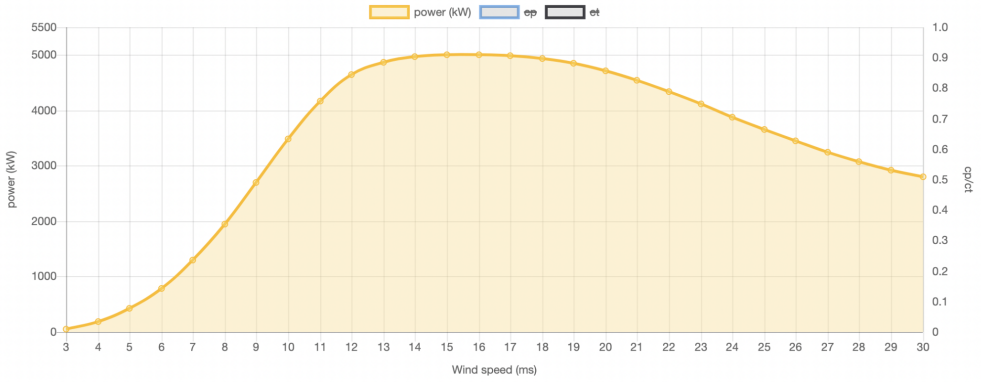

Figure 5: Power Curve of Gamesa G128-5.0 W within cut-in and cut-out speeds

2.1.2 Solar Generation

Solar panels use solar radiation to generate electricity, transforming photons that hit the photovoltaic panels into DC current. When the photons hit the panel they are absorbed by the panel's semiconducting silicon material. The movement of the electrons generates the DC current. Solar panels rely on solar elevation, cloud cover, topography, and solar irradiance. The power generated by the solar panel is proportional to the total area:

$$E = AyHr$$

Equation 2: Solar Power Generation

where A is the area of the solar panel, y is the solar panel yield, H is the solar radiation, and r is the performance ratio. The solar radiation changes throughout the year as it is dependent on solar elevation and cloud cover. The solar radiation for a given hour is:

$$H = R_0(1 - 0.75\eta^{3.4})$$

Equation 3: Solar Irradiance

$$R_0 = 990sin(\frac{\varphi_{tp} + \varphi_p}{2}) - 30$$

Equation 4: Clear sky insolation

where R_0 is clear sky insolation, η is cloud cover percentage, φ_{tp} is the solar elevation at the previous hour, and φ_p is the solar elevation at the current hour.

For this paper the solar panel chosen is the Vertex 670W+ Module. The area of the panel is A = 2.9106 from 66 210 mm silicon wafers. The solar panel yield is rated at 21.6%. The losses are estimated to be 0.0651 based on DC to AC loss and random loss [9].

2.2 Uncertainty Modeling in Stochastic Optimization

All data collected is in 24 hour format over 31 days for the month of August. The data is partitioned by hour such that the data appears as such:

$$\begin{pmatrix} hour\ 0 \\ hour\ 1 \\ ... \\ hour\ 22 \\ hour\ 23 \end{pmatrix} = \begin{pmatrix} weather\ 0 & . & . & . & weather\ n-1 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ weather\ 23 & . & . & . & weather\ n-1 \end{pmatrix}$$

Equation 5: Matrix for weather data points

$$E(X) = \frac{\sum_{i=0}^{n-1} X_i}{n}$$

Equation 6: Mean for sample weather points across row

$$\sigma(X) = \sqrt{\frac{1}{n}\left[\sum_{i=0}^{n-1} X_i^2 - n * E^2(X_i)\right]}$$

Equation 7: Standard Deviation for sample weather points across row

In Matlab the data was indexed 1 through 24. For each row in the matrix the mean and standard deviation was calculated. Although the distribution for each hour of a day in August for a given year will not appear Gaussian, it was assumed that over many years it will appear Gaussian according to the Central Limit Theorem as each hour is an identical independent random variable relative to each day. In previous papers a similar assumption has been made [6].

Each hour of a day in August has been treated as its own random variable, while each hour has been treated as its own random process where t is defined to be an hour and s the day. The moments of time samples of the random process is used to calculate the autocovariance to determine if there is a relation between the current time series and a delayed version of itself. The random process is defined as such:

$$W(X_i, s) = @hour_i[day1 \ldots dayN]$$

Equation 8: Random Process "A day's weather"

where X is the weather data at hour i. The autocovariance is taken for the time samples from 1 to 24 and from 2 to 25. W is structured as a 1 x N array. The autocovariance is taken at lag 1 to determine the relation between the current time series and the successive time series. The autocovariance is taken at lags 1 to 23 to determine the relation between the current time series and all following delayed time series. From the autocovariance the correlation coefficient is taken to determine the type and strength of the relation.

$$C(X_{1,} X_2) = E((W(X_1) - \mu(X_1)) * (W(X_2) - \mu(X_2)))$$

Equation 9: Autocovariance between hours

$$r(X_1, X_2) = \frac{C(X_{1,} X_2)}{\sqrt{C(X_{1,} X_1)} * \sqrt{C(X_{2,} X_2)}} = \frac{C(X_{1,} X_2)}{\sigma(X_1) * \sigma(X_2)}$$

Equation 10: Correlation Coefficient Strength

If an autocovariance trend is visible and the correlation coefficient is strong throughout lags, a relation is present, so an autoregression model can be used to predict future weather patterns.

2.3 Sequential Monte Carlo Simulation

To simulate daily weather patterns we can sample the Gaussian pdfs of each hour sequentially to generate random days in August. The Monte Carlo simulation appears to be an unbiased random estimator using the mean and standard deviation of a sampled distribution[10][2].

$$Simulated\ Hour\ 1\ =\ sample(Gaussian\ pdf(hour1))$$

Equation 11: Monte Carlo Sample

A mass number of samples will be generated. These samples will be used to train and test autoregression models. The samples will also be used in the final simulation to predict future values to assess the reliability of the system. The chosen autoregression model will be treated as a convolutional filter. The convolutional filter will be used on these Monte Carlo Simulation samples to filter any noise and output a simulated month of August for N simulations. Any sample that is less than 0 is removed and set to 0 as weather patterns do not produce negative values.

2.4 Autoregression Models

If a relation is found between delayed time samples of weather data, different autoregression models will be trained to predict future weeks. The autoregression models will be trained using three different types of data. They will be trained on only mean data, true data from the 1st week of August 2021, and the Monte Carlo Simulated week. They will be assessed on predicting the 1st week of August 2021 and the 2nd week of August. We will assess three different models in MATLAB: LPC, ARYULE, and ARIMA. LPC is a FIR Filter attempting to find the coefficients of a pth-order linear predictor based on past samples of a real-valued time series. ARYULE is an autoregressive all pole model attempting to find normalized autoregressive parameters for an input. ARIMA is most commonly used in previous papers. ARIMA is an univariate autoregressive integrated moving average model or a linear time series model for a univariate response process.

The autoregression model will be assessed based on MSE. Most models are assessed on R^2 factor usually. However this model isn't a true Deep Learning or Machine Learning model, so the assessment using R^2 factor wasn't performed.

$$MSE\ =\ \frac{1}{n}\sum_{i=0}^{n-1}(Y_i\ -\ \widehat{Y}_i)^2$$

Equation 12: Mean Squared Error

2.5 Max Convolutional Kernel

A kernel is used to remove any noise from the generated samples. The kernel acts as a convolutional filter between the Monte Carlo samples and the best autoregressive model. The autoregression model produces an equation which is convoluted with the simulated hours of the day.

$$y[n]\ =\ x[n]\ **\ h[n]\ =\ \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Equation 13: Convolution Equation

Convolution can produce an output larger than the sample size. In order to mimic common kernels in data science, the max value after convolution is taken as the final value for an hour of a day in the month of August[4]. The final matrix will appear to be a simulated month of August as such:

$$weather(hour,\ day)\ = MAX(\sum_{hour,day=-\infty}^{\infty} MCS[hour,\ day]AR[n\ -\ hour,\ day])$$

Equation 14: Final Simulated weather at hour of day

where MCS is monte carlo simulated hour and AR is the autoregressive equation.

2.6 Sizing the System

The simulated values will be inputted into the power equations mentioned in section 2.1 to give power coefficients for each resource. These coefficients will be treated as scalars for vectors that represent the number of solar panels and wind turbines. To find the optimal number of wind turbines and solar panels we set up the objective equation:

$$Hourly\ Energy\ Demand\ =\ SP_{power}\ *\ [number_{panels}]\ +\ WE_{power}\ *\ [number_{turbines}]$$

Equation 15: Objective Equation to meet Hourly Energy Demand

There is an objective function for each hour of each day in August. Two matrix equations are set up to find the best fit for the objective function. One matrix equation is set for where the output is the number of turbines. Another matrix equation is set for where the output is the number of panels. Essentially each equation is searching for this:

$$number_{panels}\ =\ \frac{Hourly\ Energy\ Demand - WE_{power}*[number_{turbines}]}{SP_{power}}$$

Equation 16: Objective Equation for Number of Solar Panels

$$number_{turbines}\ =\ \frac{Hourly\ Energy\ Demand - SP_{power}*[number_{panels}]}{WE_{power}}$$

Equation 17: Objective Equation for Number of Wind Turbines

The objective equation searches across its input vector looking to minimize each hour of each day. The objective matrix contains 24*31 equations. As this is computationally infeasible for a large number of simulations, logic is added to minimize this process and optimize processing. The matrix is set up in the form of y = -Ax + b.

Number of Panels-output	Hourly Energy Demand/WE power - Solar Energy/WE power * [1…N]
$\begin{pmatrix} number\ of\ panels\ 1 \\ \vdots \\ number\ of\ panels\ 744 \end{pmatrix}$	$\begin{pmatrix} \frac{demand}{wind\ energy}\ 1 \\ \vdots \\ \frac{demand}{wind\ energy}\ 744 \end{pmatrix} - \begin{pmatrix} \frac{solar\ energy}{wind\ energy}\ 1 \\ \vdots \\ \frac{solar\ energy}{wind\ energy}\ 744 \end{pmatrix} \cdot (1 \quad . \quad . \quad . \quad . \quad N)$

Equation 18: Objective Matrix containing objective equations across month

The output to our objective equation will produce multiple results. In order to assess system reliability only 4 scenarios will be considered: maximum number of turbines and panels, minimum number of turbines and panels, maximum number of turbines and minimum number of panels, maximum number of panels and minimum number of turbines. The output of these 4 scenarios will be used to simulate the amount of energy produced at each hour.

2.7 System Reliability

The energy produced at each hour will be compared to the total energy demand at each hour. If the energy demand is not met at hour i, it will be noted as a shortage. The total number of shortages for a simulated month will be the final output of the monte carlo simulation. The system reliability will be calculated for N iterations of the simulation using the equation:

$$System\ Up\ \% \ = \ \frac{Total\ Number\ of\ Hours - Number\ of\ Shortages}{Total\ Number\ of\ Hours} * 100\%$$

Equation 19: System Reliability Factor

3 Simulation

3.1 Variable Wind Power Simulation

3.1.1 Analyze the Wind Speed of each hour in a day in August

To analyze the collected data of Annapolis in August, 5 years of data was collected. Data was initially collected in a 1D array of (1x24*31). A 2D Matrix [hourly_wind_month] was created to collect 24 hours of data of size N. For each month, the ith hour was sampled and stored in the ith row of [hourly_wind_month].

Transfer each hour of day into hourly_wind_month

```
for hour = 1:24
   for day = 1:31
       hour_of_day = (day-1)*24 + hour;
       if(hour_of_day <= max_hour)
           hourly_wind(day) = windspeed_hourly(hour_of_day);
       end
   end
   hourly_wind_month(hour, :) = hourly_wind;
end
```

This transformation was done for each month of the 5 years. Once the hourly_wind_month was filled, the standard statistic parameters were found for each row or hour.

Find Standard Statistics Parameters

```
for i = 1:24
   mean_hr_windspeed(i) = mean(hourly_wind_month(i, :));
   stnd_hr_windspeed(i) = std(hourly_wind_month(i, :));
   var_hr_windspeed(i) = stnd_hr_windspeed(i)^2;
end
```

Once the parameters were found a pdf of -5*sigma, 5*sigma was generated. This pdf was not used however. For sampling, randn was used to randomly simulate wind values for each hour. Next the autocorrelation plot was analyzed to see if any trend in the data could be identified visually.

```
scatter(windspeed_hourly(1:24), windspeed_hourly(2: 25), 'bo')
```

The autocorrelation was done only for the first day of August. The autocorrelation was computed relative to a day, to identify a trend within a day not over the week or month.

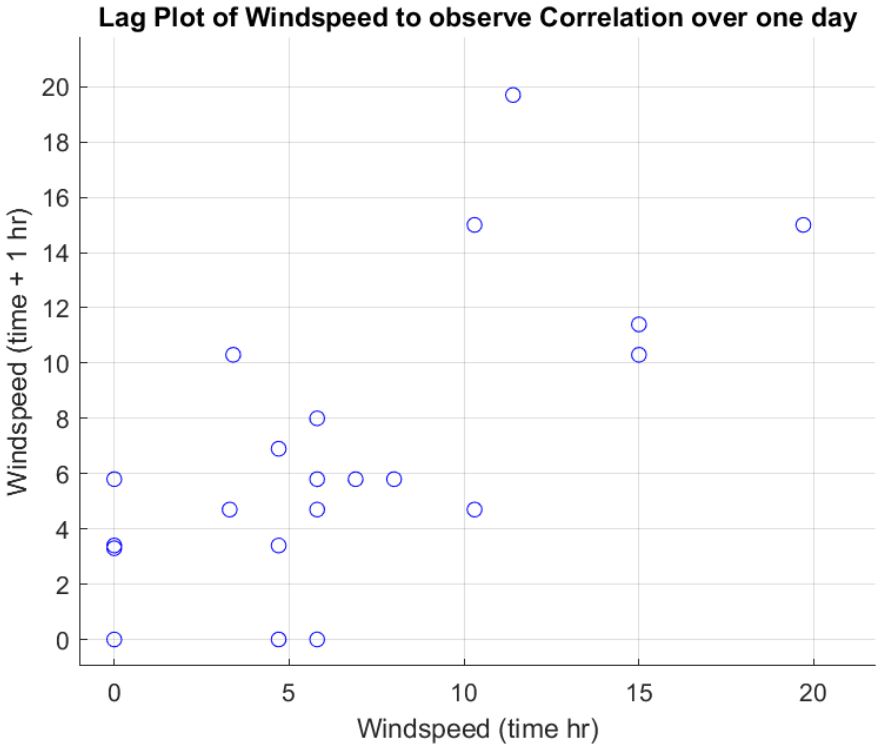

Figure 6: Lag (1) Plot of Windspeed shows Positive Autocorrelation Trend over Single Day

As the lag plot showed a general positive trend, the autocorrelation over all lags and successive lags was computed. For standard autocorrelation over all lags, later time sequences are correlated with the time sequence at 1. Recall Equation 10, the same equation was used to compute autocorrelation over all lags. The plot is computed by correlating each sequence corresponding to an hour. As hourly_wind_month is in the form of X(t, s), the autocorrelation function can be computed by

Rx(T, s) = E[X(t1 + T, s)*X(t1, s)] as shown below

AutoCorrelation between first hour and all latter hours

```
R_lag_windspeed = zeros(24, 1);
for i = 1:23
   %Auto Correlation function between first hour and latter hours Rx(T) =
E[X(t+T)X(T)]
   windspeed_tau = hourly_wind_month(i+1, :) - mean_hr_windspeed(i+1);
   windspeed_t = hourly_wind_month(1, :) - mean_hr_windspeed(1);
   sigma_tau = stnd_hr_windspeed(i+1);
   sigma_t = stnd_hr_windspeed(1);
   R_lag_windspeed(i) = mean(windspeed_tau.*windspeed_t)/(sigma_tau*sigma_t);
end
```

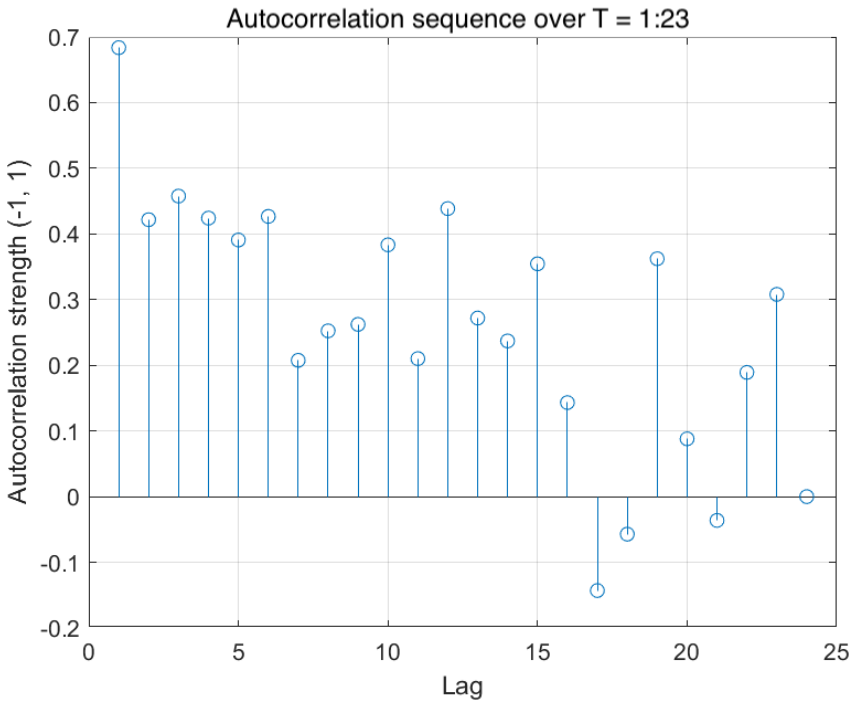

Figure 7: Autocorrelation Sequence over T = 1:23

The plot shows high correlation between wind speeds throughout the day. Wind speeds seem highly dependent on the first wind speed of the day even till midday.

The autocorrelation function below computes autocorrelation between successive lags. Compared to the previous function, one process at t is stationary while the other process moves throughout time. Here both sequences move together throughout time by 1 step.

Rx(T, s) = E[X(t1 + T, s)*X(t1, s)] where (t1 + T - t1) = 1, for all t

Autocorrelation Function between successive hours

```
for i = 1:24
   %Auto Correlation function between successive hours Rx(T) = E[X(t+T)X(T)]
   if i<24
        windspeed_tau = hourly_wind_month(i+1, :) - mean_hr_windspeed(i+1);
        windspeed_t = hourly_wind_month(i, :) - mean_hr_windspeed(i);
        sigma_tau = stnd_hr_windspeed(i+1);
        sigma_t = stnd_hr_windspeed(i);
        R_hr_windspeed(i) =
mean(windspeed_tau.*windspeed_t)/(sigma_tau*sigma_t);
   else
        windspeed_tau = hourly_wind_month(1, :) - mean_hr_windspeed(1);
        windspeed_t = hourly_wind_month(i, :) - mean_hr_windspeed(i);
        sigma_tau = stnd_hr_windspeed(1);
        sigma_t = stnd_hr_windspeed(i);
        R_hr_windspeed(i) =
mean(windspeed_tau.*windspeed_t)/(sigma_tau*sigma_t);
   end
end
```

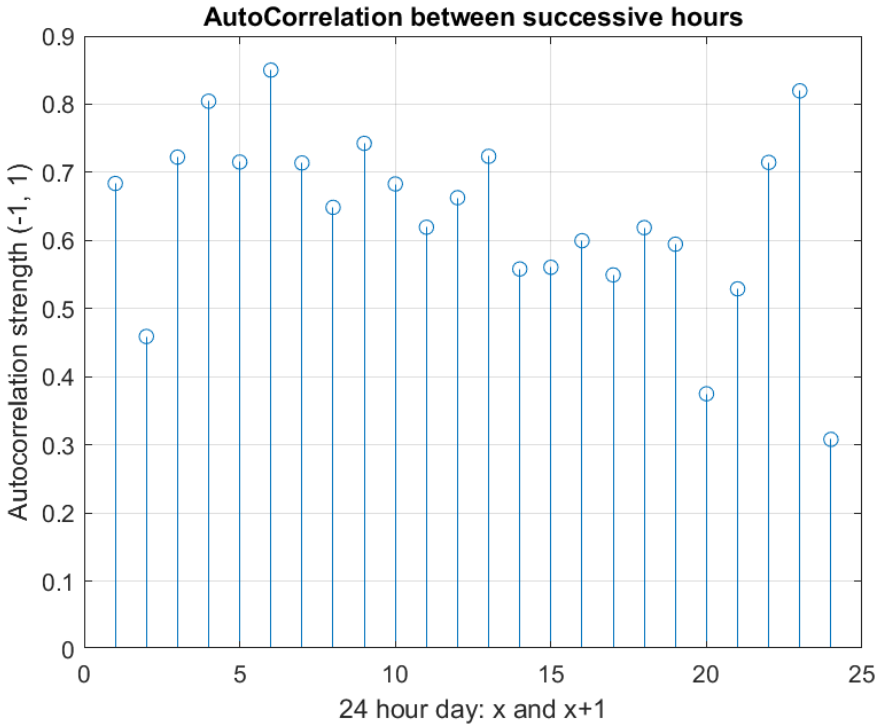

Figure 8: Autocorrelation Sequence over all sequences where T = 1, and t increments by 1

The plot shows very high correlation between successive sequences. It seems that these random processes are not independent of each other. Rather they are highly correlated and therefore highly dependent. The wind speed at one hour will determine the wind speed at the next.

3.1.2 Wind Speed Models

As seen in section 3.1.1, the random sequences of hourly wind speeds X(t, s) as t = [1:24] are highly dependent on each other and have high correlation. Following this observation it was decided simulating a Random Walk or Wiener Process would not be possible, to generate wind speeds throughout any given day. An autoregressive model is appropriate to model these wind speeds. Various autoregressive functions have been explored with different training and test inputs.

The most commonly used model in prior art is ARIMA. The ARIMA model was trained along with the ARYULE, and LPC models. ARYULE is an autoregressive all-pole model. LPC is pth order linear predictor that acts as an FIR model that predicts the next values of a given input signal. As LPC is an FIR filter it has linear group delay so it does not introduce skewness. The drawback of an LPC predictor is that FIR filters smooth signals. So wind speeds with high spikes will not be included in the prediction.

To find the best trained model, MSE was used as the scoring factor of each model. R^2 is usually used for prediction models, however the models trained are not Machine Learning models.

The models were generated and trained as defined in the table below:

ARIMA Model

Training Data	Test Data	Input Parameters (order)
Mean of wind speeds	Week 2 of august 2021	1, 1, 1
Week 1 of august 2021	Week 2 of august 2021	1, 1, 1
Monte carlo (randomly sampled week)	Week 1 of august 2021	1, 1, 1
Monte carlo (randomly sampled week)	Week 2 of august 2021	1, 1, 1

ARYULE Model

Training Data	Test Data	Order
Mean of wind speeds	Week 2 of august 2021	7
Week 1 of august 2021	Week 2 of august 2021	7
Monte carlo (randomly sampled week)	Week 1 of august 2021	7
Monte carlo (randomly sampled week)	Week 2 of august 2021	7

The order of 7 was chosen based on autocorrelation plots of the error = output_model - test_data produced. Filters of other orders were explored, but are not mentioned here for the sake of brevity.

LPC Model

Training Data	Test Data	Order
Mean of wind speeds	Week 2 of august 2021	7
Week 1 of august 2021	Week 2 of august 2021	7
Monte carlo (randomly sampled week)	Week 1 of august 2021	7
Monte carlo (randomly sampled week)	Week 2 of august 2021	4

The order of 7 and 4 was chosen based on autocorrelation plots of the error = output_model - test_data produced. Filters of other orders were explored, but are not mentioned here for the sake of brevity.

While testing the filters, it is ensured that all values are positive as wind speeds cannot be negative realistically. The error of each filter is based on the modified output of the model and the true test data.

An example is given below for the ARYULE model trained on monte carlo data and compared to week 2 of August 2021 data.

ARYULE model trained, output modified, signal plot, xcorr plot, MSE

```
[yule_mc_eq,p4] = aryule(monte_carlo_week_1,7);
est_yule_mc_x = filter([0 -yule_mc_eq(2:end)],1,monte_carlo_week_1);
%compare results visually
for i = 1:length(est_yule_mc_x)
   if est_yule_mc_x(i) < 0
       est_yule_mc_x(i) = 0;
   end
end
yule_mc_err = true_data - est_yule_mc_x';
[yule_mc_acs,yule_mc_lags] = xcorr(yule_mc_err,'coeff');
MSE_yule_mc = sum(yule_mc_err.^2)/length(yule_mc_err);
%% Plots
figure
plot(1:168, true_data, 1:168, est_yule_mc_x,'--')
title('Yule Walker Monte Carlo Input Compare to Week 2');ylim([-1 30])
grid on
figure
plot(yule_mc_lags,yule_mc_acs)
grid
title('Yule Walker Monte Carlo Input Compare to Week 2');
xlabel('Lags')
ylabel('Normalized Autocorrelation')
ylim([-0.2 1.1])
grid on
```

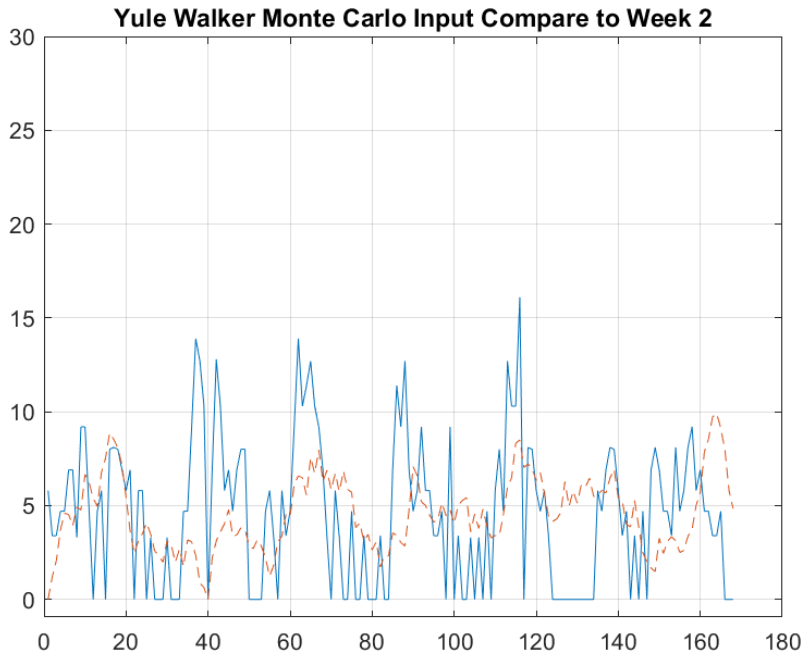

Figure 9: True Data of Week 2 (Blue) vs Predicted Data (Dashed) over 168 hours

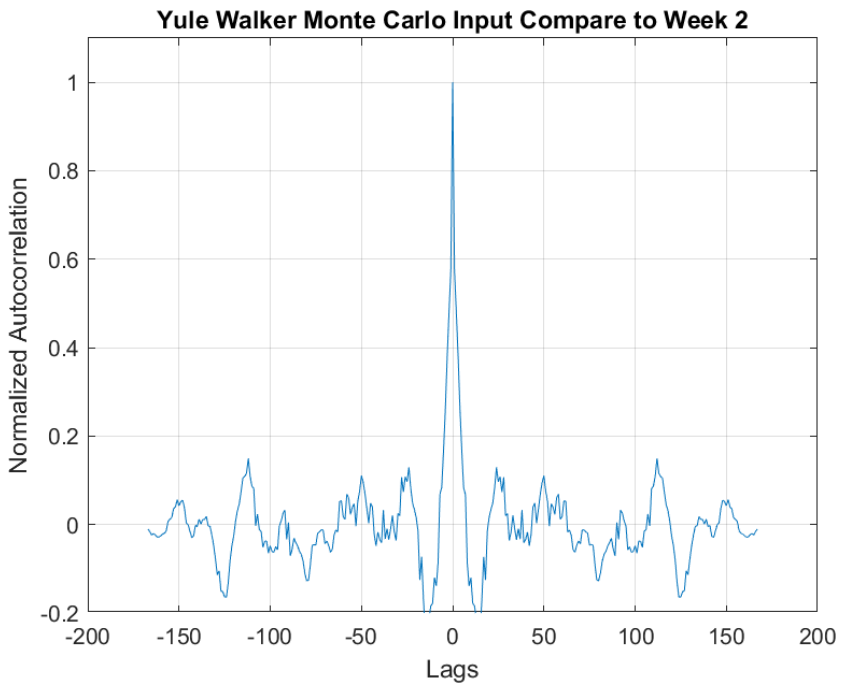

Figure 10: The Autocorrelation plot of the true error between the predicted and true data

Although an autocorrelation function with an impulse embedded in Gaussian noise is ideal, it is not realistic for wind speed data. Strong autocorrelation across hours have been found, so the plot is acceptable. The highest correlation outside of the zero range is 0.17 which is acceptable.

The function outputs the best regression equations for each model to the function main_analysis.m. Under main_analysis.m the best model is chosen based on the MSE of its corresponding model.

3.1.3 Simulating Wind Speeds using AR Model

Using a Monte Carlo simulation wind speeds are generated for the month of August. The Monte Carlo simulation is a basic simulation performed by simply sampling the Gaussian pdfs of the ith hour for each ith hour of 31 days. Advanced algorithms are not used to search the Gaussian space for the Monte Carlo simulation. The Monte Carlo is only dependent on i = [1:24]. Similar to section 3.1.2, values under 0 are not accepted and set to 0.

Monte Carlo Simulation of wind speeds for a month

```
N = 1; %number of simulations
monte_carlo = zeros(24, N*31);
wind_simulate_day = zeros(24, N*31);
for j = 1:N*31
   for i = 1:24
       monte_carlo(i, j) = normrnd(mean_wind(i), std_wind(i));
       if(monte_carlo(i, j) < 0)
           monte_carlo(i, j) = 0;
       end
   end
end
```

Once the whole month has been generated the AR model is applied, acting as a convolutional filter. Convolving a signal with a set of numbers acts as a filter. In this filter a max function has also been applied, so that the maximum value of the filtered signal is chosen as the output for a given hour. Note the wind_equation in the code snippet below means the AR model coefficients.

Max Convolutional Kernel

```
for j = 1:N*31
   for i = 1:24
       wind_simulate_day(i, j) = max(conv(wind_equation, monte_carlo(i, j)));
   end
end
```

3.1.4 Wind Power Calculation

Once the wind simulate day matrix has been generated, the corresponding wind power output at the given speeds can be calculated. The wind power simulated matrix is generated by calculating the wind power at each index for its given wind speed. The wind speed in the original data was in mph, for the wind power equations speed must be given in m/s. A speed factor of 0.44704 is applied for the conversion. The rest of the specifications for the wind turbine has been specified previously in 2.1.1. The cut-in speed for the motor is 2m/s. The cut-out speed for the motor is 27 m/s.

Wind Power calculation, simulated from wind speeds simulated

```
diameter = 128.0; %diameter of rotor
r = diameter/2;
rho = 1.225; %air density
nu = 0.59; %maximum efficiency of turbine according to betz law
speed_factor = 0.44704;
watt_mw = 1/1000000;
wind_power_simulated = zeros(24, N*31);
for j = 1:N*31
   for i = 1:24
       speed = wind_simulate_day(i, j)*speed_factor;
       if speed < 1.9
           speed = 0;
       elseif speed > 27.01
           speed = 0;
       end
       wind_power_simulated(i, j) = watt_mw*pi/2*r^2*speed^3*rho*nu/1;
   end
end
```

3.2 Variable Solar Power Simulation

3.2.1 Analyze the Cloud Cover of each hour in a day in August

To analyze the collected data of the Catoctin Mountains in August, 5 years of data was collected. Similar to the wind speed data, cloud cover data was initially collected in a 1D array of (1x24*31). A 2D Matrix

[hourly_cloud_month] was created to collect 24 hours of data of size N. For each month, the ith hour was sampled and stored in the ith row of [hourly_cloud_month].

Transfer each hour of day into hourly_wind_month

```
for hour = 1:24
   for day = 1:31
       hour_of_day = (day-1)*24 + hour;
       if(hour_of_day <= max_hour)
           hourly_cloud(day) = cloudcover_hourly(hour_of_day);
       end
   end
   hourly_cloud_month(hour, :) = hourly_cloud;
end
```

This transformation was done for each month of the 5 years. Once the hourly_cloud_month was filled, the standard statistic parameters were found for each row or hour.

Find Standard Statistics Parameters

```
for i = 1:24
   mean_hr_cloudcover(i) = mean(hourly_cloud_month(i, :));
   stnd_hr_cloudcover(i) = std(hourly_cloud_month(i, :));
   var_hr_cloudcover(i) = stnd_hr_cloudcover(i)^2;
end
```

Once the parameters were found a pdf of -5*sigma, 5*sigma was generated. This pdf was not used however. For sampling, randn was used to randomly simulate cloud cover values for each hour. Next the autocorrelation plot was analyzed to see if any trend in the data could be identified visually.

```
scatter(cloudcover_hourly(1:24), cloudcover_hourly(2: 25), 'bo')
```

The above autocorrelation was done only for the first day of August. The autocorrelation was computed relative to a day, to identify a trend within a day not over the week or month.

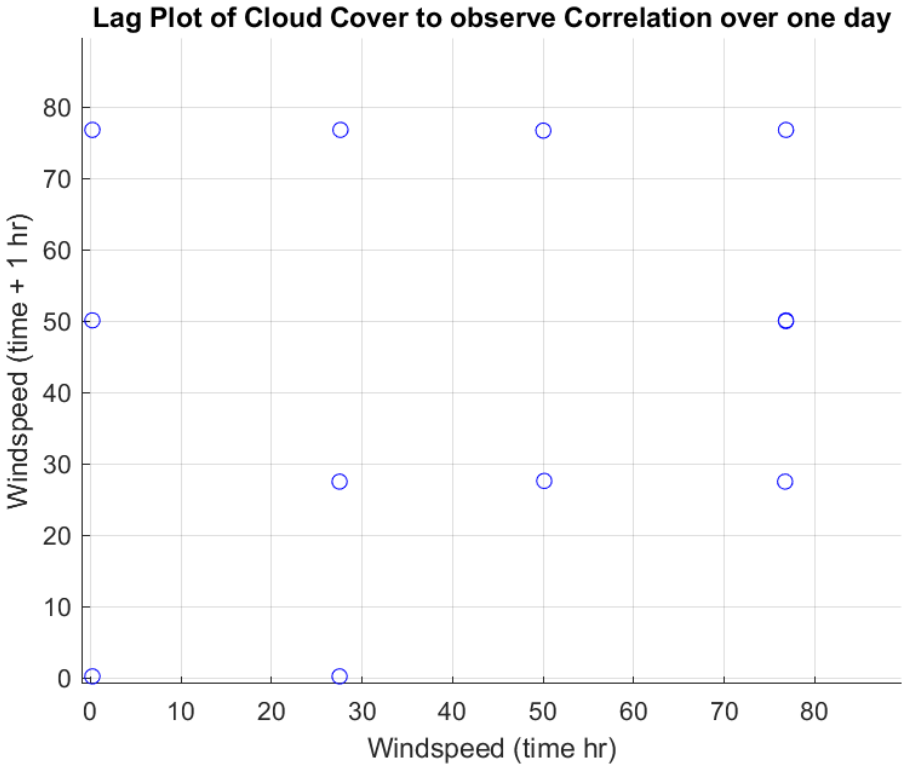

Figure 11: Lag (1) Plot of Cloud Cover shows No Autocorrelation Trend over Single Day

As the lag plot showed no trend and appeared random, no effort to calculate the autocorrelation was made. The autocorrelation was plotted but it also appeared random. As the cloud cover was truly random, no AR model was generated and trained to simulate cloud cover values. Cloud cover values were randomly simulated using Monte Carlo simulation across each hour for a given month.

3.2.2 Simulating Cloud Cover

Using a Monte Carlo simulation wind speeds are generated for the month of August. The Monte Carlo simulation is a basic simulation performed by simply sampling the Gaussian pdfs of the ith hour for each ith hour of 31 days. Advanced algorithms are not used to search the Gaussian space for the Monte Carlo simulation. The Monte Carlo is only dependent on i = [1:24]. Similar to section 3.1.2, values under 0 are not accepted and set to 0.

Monte Carlo Simulation of cloud cover for a month

```
monte_carlo_sun = zeros(24, N*31);
for j = 1:N*31
   for i = 1:24
       cloud_cover = normrnd(mean_sun(i), std_sun(i));
       monte_carlo_sun(i, j) = cloud_cover/100;
       if(monte_carlo_sun(i, j) < 0)
           monte_carlo_sun(i, j) = 0;
       end
       if(monte_carlo_sun(i, j) > 97)
           monte_carlo_sun(i, j) = 97;
       end
```

```
   end
end
```

3.2.3 Solar Power Calculation

Once the [monte_carlo_sun] matrix has been generated, the corresponding solar power output can be calculated. The solar_radiance matrix is generated by calculating the solar radiance at each index for its given cloud cover and solar elevation. The solar radiance can be calculated by referring to Equations 3 and 4.

Solar Radiance Calculation

```
sun_angle_august = reshape(sun_elevation_angle, 24, 31);
Rknot = 990;
R_clear = 0;
solar_radiance = zeros(24, 31);
cloud_factor = 1;
%Probability for each day
for j = 1:N*31
   for i = 1:24
       if i == 1 && j == 1
          theta_p = -25;
       elseif i ==1
           theta_p = sun_angle_august(24, j-1);
       else
           theta_p = sun_angle_august(i - 1, j);
       end
       theta_c = sun_angle_august(i, j);
       theta = (theta_p + theta_c)/2;
       theta = theta*pi/180;
       R_clear = 990*sin(theta)-30;
       cloud_cover = monte_carlo_sun(i, j);
       if(theta_p < 0 && theta_c <0)
           solar_radiance(i, j) = 0;
       else
           cloud_factor = 1 - 0.75*(cloud_cover^3.4);
           solar_radiance(i, j) = abs(R_clear*cloud_factor);
       end
  end
end
```

Once the solar radiance matrix has been generated the solar power for the given solar radiance can be calculated at each index for hour and day. Solar elevation changes throughout the day. Refer to Equation 2 for solar power output and Section 2.1.2 for solar panel specifications.

Solar Power Calculation

```
Area_panel = 66*(210^2)/(1000^2);
yield = .216; %maximum efficiency
performance = 0.9349;
solar_power = zeros(24, 31);
```

```
num = 1000;
for j = 1:N*31
   for i = 1:24
       solar_power(i, j) = solar_radiance(i,
j)*Area_panel*yield*performance/1000;
   end
end
```

3.3 Sizing the System

From the solar_power and wind_power_simulated matrices, the total power generated can be calculated. Here the size of the system has to be optimized such that the total energy demand during the month is met. At some hours the total energy demand cannot be met regardless of the size because the solar power and wind power produced is 0 due to weather conditions. For this matrix set up, refer to Equation 15: $Hourly\ Energy\ Demand\ =\ SP_{power}\ *\ [number_{panels}]\ +\ WE_{power}\ *\ [number_{turbines}]$, there are a total of 24*31 equations to solve. Initially an attempt was made to use MATLAB's system toolbox to solve these as linear equations. However it produced no solution. An inequality solver was used as well, but the solution produced did not have any value.

The objective equation searches across its input vector looking to minimize each hour of each day. The objective matrix contains 24*31 equations. As this is computationally infeasible for a large number of simulations, logic was added to minimize this process and optimize the processing. The input vector for each equation is set from 1 to 1000 to force acceptable results. There is a point where convergence will occur for both of these equations at most instances, however the answer tends to infinity.

Two equations are solved concurrently, for both equations the input vector is set to 1 to 1000:

$$number_{panels} = \frac{Hourly\ Energy\ Demand - WE_{power}*[number_{turbines}]}{SP_{power}}$$

$$number_{turbines} = \frac{Hourly\ Energy\ Demand - SP_{power}*[number_{panels}]}{WE_{power}}$$

Through MATLAB it was found there is no intersection of either of the objective equations when the input vector was set from 1 to 1000. The workaround to this issue was to instead find the minimum distance between each output y across all iterations x between 1 to 1000.

First each column at input vector (i) was checked to verify at least one non-zero value existed. If the column was non-zero it was sorted. Then the difference function of the column was taken so that the differences between each value in the column was left. The minimum difference found was the answer to the objective equations. The output at the minimum difference was noted and that value was returned to the final matrix. In the final matrix there were many answers to the objective equation ranging from 1000 to 10000 number of turbines and panels.

Objective equation to solve for number of panels where number of turbines (1:1000) is the input

```
for day = 1:31
   for i = 1:24
       hr_day = (day-1)*24 + i;
       if solar_power(i,day) == 0
           num_p(:, i) = 0;
       else
           num_p(:, i) = (demand_hourly(hr_day) - wind_power_simulated(i,
day)*t)/solar_power(i,day);
       end
   end
   for k = 1:1000
           A = num_p(k, :);
           idx = find(A); %get nonzero elements
           if isempty(idx)
               min_num_p(k, day) = inf;
               min_op_p(k, day) = inf;
           else
               A = A(idx);
               A = sort(A);
               diff_A = diff(A);
               [min_diff, min_idx] = min(diff_A);
               min_op_p(k, day) = min_diff; %find minimum distance
               min_num_p(k, day) = A(min_idx);
           end
   end
end
```

Objective equation to solve for number of turbines where number of panels (1:1000) is the input

```
for day = 1:31
   for i = 1:24
       hr_day = (day-1)*24 + i;
       if wind_power_simulated(i,day) == 0
           num_t(:, i) = 0;
       else
           num_t(:, i) = (demand_hourly(hr_day) - solar_power(i,
day)*t)/wind_power_simulated(i,day);
       end
   end
   for k = 1:1000
       A = num_t(k, :);
       idx = find(A); %get nonzero elements
       if isempty(idx)
           min_num_t(k, day) = inf;
           min_op_t(k, day) = inf;
       else
           A = A(idx);
           A = sort(A);
           diff_A = diff(A);
           [min_diff, min_idx] = min(diff_A);
```

```
            min_op_t(k, day) = min_diff; %find minimum distance
            min_num_t(k, day) = A(min_idx);
        end

    end
end
```

The maximum and minimum values of the system were noted after the output of these objective equations.

```
[min_dist_p, min_dist_idx_p] = min(min_op_p);

[min_vals_p, min_vals_idx_p] = min(min_num_p);

[min_dist_t, min_dist_idx_t] = min(min_op_t);

[min_vals_t, min_vals_idx_t] = min(min_num_t);
```

3.4 Number of Shortages for a given System size

The system was analyzed for the maximum and minimum size of the system. To analyze the energy produced at each hour was compared to the total energy output based on the size of the system. If the demand was not met, it was added to the number of shortages for the simulation iteration.

Number of Shortages

```
for j = 1:31
    for i = 1:24
        hr_day = (j-1)*24 + i;
        total_power(i, j) = wind_power_simulated(i, j)*max_turbines...
            + solar_power(i, j)*max_panels;
        if (total_power(i, j) < demand_hourly(hr_day))
            number_of_shortages_tpmax = number_of_shortages_tpmax + 1;
        end

        total_power(i, j) = wind_power_simulated(i, j)*min_turbines...
            + solar_power(i, j)*min_panels;
        if (total_power(i, j) < demand_hourly(hr_day))
            number_of_shortages_tpmin = number_of_shortages_tpmin + 1;
        end

        total_power(i, j) = wind_power_simulated(i, j)*min_turbines...
            + solar_power(i, j)*max_panels;
        if (total_power(i, j) < demand_hourly(hr_day))
            number_of_shortages_t_pmax = number_of_shortages_t_pmax + 1;
        end

        total_power(i, j) = wind_power_simulated(i, j)*max_turbines...
            + solar_power(i, j)*min_panels;
        if (total_power(i, j) < demand_hourly(hr_day))
            number_of_shortages_tmax_p = number_of_shortages_tmax_p + 1;
```

```
        end
    end
end
```

3.5 System Reliability for a given iteration

This simulation was performed 50 times and was computationally exhaustive. The reliability was calculated by taking the total number of hours - number of shortages divided by total number of hours to determine how often the system was able to meet demand.

Returns Number of Shortages for the simulation iteration i

```
for i = 1:N
    [max, min, wind_max, sun_max] = main_analysis();
    max_system_rel(i) = max;
    min_system_rel(i) = min;
    wind_max_system_rel(i) = wind_max;
    sun_max_system_rel(i) = sun_max;

end
```

The system reliability will be calculated for N iterations of the simulation using Equation 19:

System Reliability (%Up)

```
total_hours = 24*31;
system_up_percent_max = 100*(total_hours - max_system_rel)/total_hours;
system_up_percent_min = 100*(total_hours - min_system_rel)/total_hours;
system_up_percent_wmax = 100*(total_hours - wind_max_system_rel)/total_hours;
system_up_percent_pmax = 100*(total_hours - sun_max_system_rel)/total_hours;
```

The results for each sizing of the system is shown below. Even at the maximum system size the reliability ranges from 40 to 60% yielding poor results as expected. August has the worst wind speeds for the whole year. The statistics for each system reliability is shown below the figures.

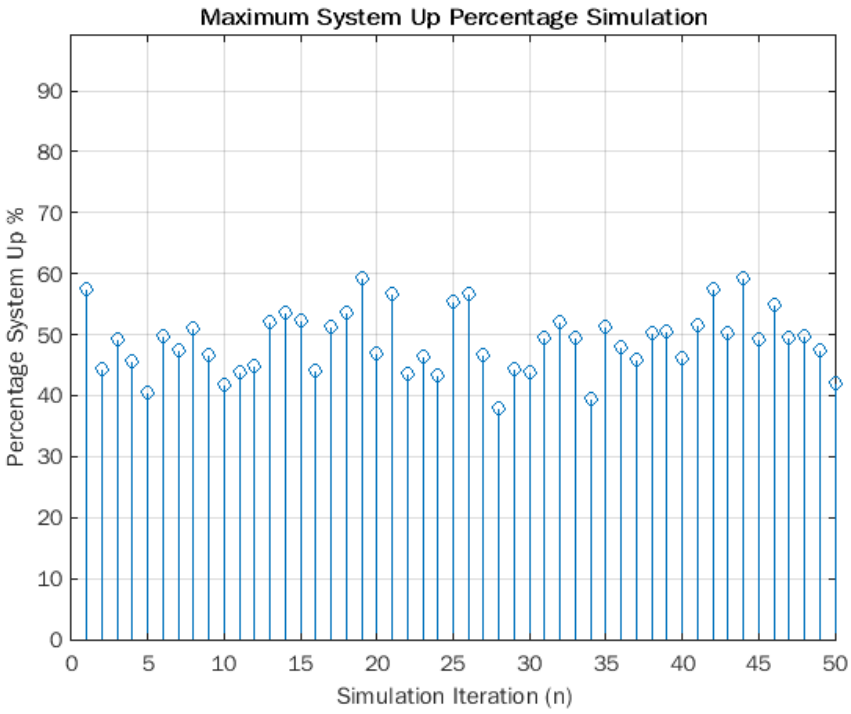

Figure 12: System Reliability for Maximum Size of system at iteration n

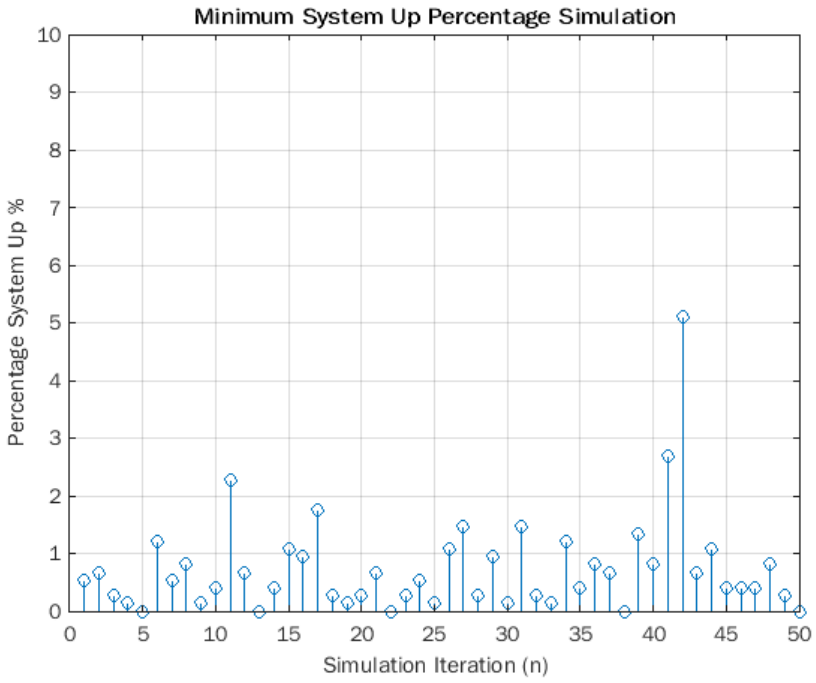

Figure 13: System Reliability for Minimum Size of system at iteration n

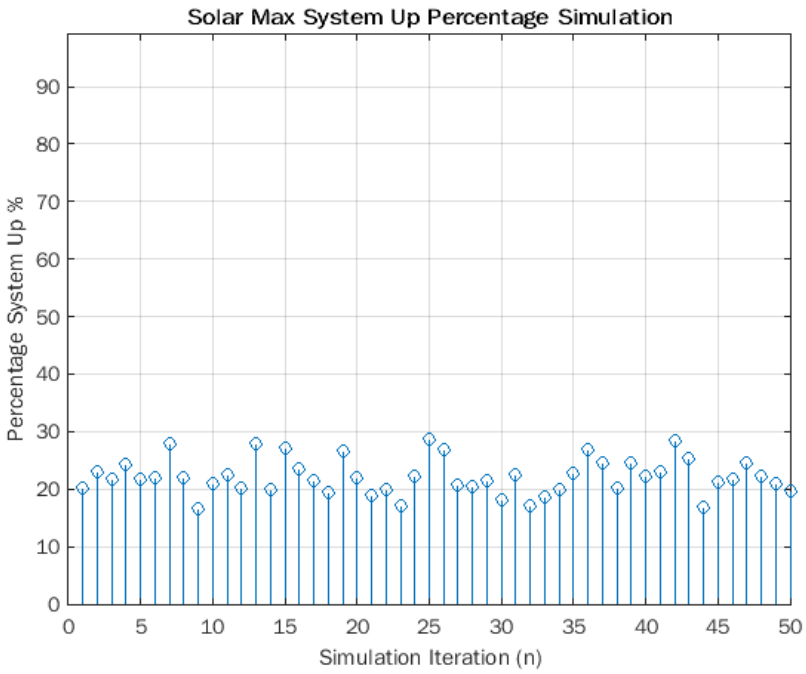

Figure 14: System Reliability for Maximum Number of Panels, Minimum Number of Turbines at n

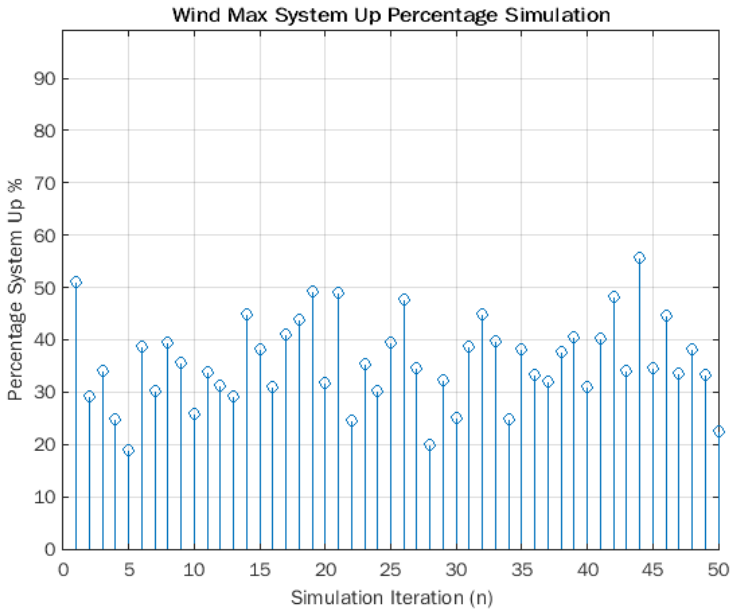

Figure 15: System Reliability for Maximum Number of Turbines, Minimum Number of Panels at n

SYSTEM RELIABILITY STATISTICS

	Mean	Standard Deviation
Maximum Turbines, Panels	0.4890	0.0513
Minimum Turbines, Panels	0.0074	0.0086
Maximum Panels, Min Turbines	0.2216	0.0311
Maximum Turbines, Min Panels	0.3573	0.0823

The system reliability statistics show how unlikely it is that any of these systems will meet peak demand in the month of August. It's important to note that the maximum system has low variance that is most likely due to the fact that solar panels are more reliable in the month of August. The standard deviation of the Maximum Panel system is lower than the Maximum Turbine system. As stated in the beginning of the paper, it has been proven that turbines do introduce a higher rate of intermittency and risk to an electric grid power system.

Conclusion

During the month of August relying on only renewable energy would be a risk for the Washington D.C. area. Even at a maximum system size which would be financially impossible, the system would be very unlikely to meet peak demand. The system has a 58.32% likelihood of meeting 50% power demand. Even so, renewable energy integration is a critical part of the future. A statewide solution to this issue would be to share electricity across the MIDA region or even across regions creating a mesh system of renewable energy microgrids. Microgrids can trade energy and calculate the risk of trading energy based on their predicted peak demand and production. Oftentimes regions in the US produce too much power and the generated energy is either wasted or stored. During these times if a region like Washington D.C. is not meeting peak energy demand, energy can be traded to save energy and meet demand. Further research may include setting up a Game Theory Framework to analyze how and when cities can trade energy.

If Washington D.C. implemented the minimum turbine and maximum panel system at a lower cost they would meet 20% energy demand 75% of the time. However this system would be weak in the winter time, where a maximum turbine and minimum panel system would best serve demand. Further research is required to calculate the risk of integrating a certain number of panels and turbines. Defining an objective function that minimizes risk and cost and maximizes reliability can take this research forward to analyze the optimal size of the system.

References

1. Bauer, L. (n.d.). *Gamesa g128-5.0MW*. Gamesa G128-5.0MW - 5,00 MW - Wind turbine. Retrieved December 9, 2021, from https://en.wind-turbine-models.com/turbines/767-gamesa-g128-5.0mw.

2. Constante-Flores, G., & Illindala, M. (n.d.). *Data-driven probabilistic power flow analysis for a distribution system with renewable energy sources using Monte Carlo Simulation*. IEEE Xplore. Retrieved December 9, 2021, from https://ieeexplore.ieee.org/document/7945118.

3. Degeilh, Y., & Gross, G. (2014, August 15). *Stochastic simulation of power systems with integrated intermittent renewable resources*. International Journal of Electrical Power & Energy Systems. Retrieved December 9, 2021, from https://www.sciencedirect.com/science/article/pii/S0142061514004840.

4. Ganesh, P. (2019, October 18). *Types of convolution kernels : Simplified*. Medium. Retrieved December 9, 2021, from https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37.

5. *Global wind atlas*. Global Wind Atlas. (n.d.). Retrieved December 9, 2021, from https://globalwindatlas.info/.

6. Lopes, V. S., & Borges, C. (n.d.). *Impact of the combined integration of wind generation and small hydropower plants on the system reliability*. IEEE Xplore. Retrieved December 9, 2021, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6867368.

7. *Real-time operating grid - U.S. Energy Information Administration (EIA)*. Real-time Operating Grid - U.S. Energy Information Administration (EIA). (n.d.). Retrieved December 9, 2021, from https://www.eia.gov/electricity/gridmonitor/dashboard/electric_overview/regional/REG-MIDA.

8. *U.S. Energy Information Administration - EIA - independent statistics and analysis*. District of Columbia - State Energy Profile Overview - U.S. Energy Information Administration (EIA). (n.d.). Retrieved December 9, 2021, from https://www.eia.gov/state/?sid=DC#tabs-2.

9. *US/vertex 670W+ module*. Trina Solar. (2021, September 30). Retrieved December 9, 2021, from https://www.trinasolar.com/us/product/VERTEX-DEG21C.20.

10. Vallée, F., Lobry, J., & Deblecker, O. (n.d.). *(PDF) System Reliability Assessment Method for wind power ...* Retrieved December 9, 2021, from https://www.researchgate.net/publication/3268561_System_Reliability_Assessment_Method_for_Wind_Power_Integration.

11. Zakaria, A., Ismail, F. B., Lipu, M. S. H., & Hannan, M. A. (2019, July 16). *Uncertainty*

models for stochastic optimization in renewable energy applications. Renewable Energy. Retrieved December 9, 2021, from https://www.sciencedirect.com/science/article/pii/S0960148119311012.