Short time horizon solar power forecasting

Bennet Meyers[1] and Júlio Hoffimann[2]

[1]Stanford University, Electrical Engineering
[2]Stanford University, Energy Resources Engineering

December 14, 2017

1 Introduction

Solar photovoltaic (PV) systems show variability in output across all time scales, from seasonal changes in available sunlight to a cloud moving in front of the sun in a matter of seconds. This creates a challenge for grid operators who need to maintain the balance between load and generation within the energy distribution system.

We are exploring algorithms to predict the aggregate power output of many photovoltaic systems in a single geographic region on a 3-hour time horizon at 5-minute steps (a 36-step forecast) based only on observed system power. The goal is to correctly identify upcoming "ramp events," or large positive or negative deviations from a long-term trend over a short time period (Sevlian and Rajagopal, 2013). Identifying these ramp events before they occur will allow grid operators to plan for large changes in net load, thereby allowing for deeper penetration of solar power generation on the grid.

The input to the forecaster models is a window of observed historical data, culminating at the start of the forecast period. The historical window size is tuned for the specific model implementation, and ranged from 2 to 3 hours depending on the specific implementation (see methods section for additional discussion). We train models on a set of training data and then condition the forecasts on recently observed data. We are considering a "many-to-one" forecasting topology, where the individual site signals are used as feature data to predict the aggregate power, and comparing these results to a "one-to-one" approach, where only the aggregate signal is considered.

We explore ARIMA, k nearest-neighbors for functional regression, and fully connected neural networks to estimate the forecast window, and we compare these models to a simple "measure and hold" persistence model. Project code and Jupyter Notebooks are available at `https://github.com/bmeyers/SolarForecasting`.

2 Related Work

Forecasting photovoltaic solar power output has been well studied, as evidence by comprehensive review papers such as (Pelland et al., 2013) and (Inman et al., 2013). This domain has its roots in atmospheric and weather modeling and forecasting, which stretches back to the first half of the 20th century, and early numerical weather prediction approaches (see, for example (Smagorinsky, 1958)). All manor of forecasting approaches have been attempted by past researchers, including regressive methods (e.g. auto-regressive, integrative, moving average and related models) and machine learning approaches (e.g. neural networks, k-NN). Researchers have explored including other data sources such as satellite/remote sensing, numerical weather predictions (NWP) (Perez et al., 2010), and total sky imagers (Chow et al., 2011) to aid with generating better quality forecasts.

This work is differentiated from previous work in two important aspects: complexity of the forecast and limitation of the feature data. We are performing a 36-step forecast at 5-minute intervals for a total forecast horizon of 3 hours. Previous papers have typically only done multi-step forecasting with hourly or daily data, or one- or two-step forecasts on subhourly data (Inman et al., 2013). Because solar PV power exhibits increasing variance on smaller and smaller time scales, accurately predicting 5-minute power over multiple hours is a particularly difficult problem. In addition, we were interested in investigating how much information we could extract from the time-series signals directly, rather than relying on inputs from exogenous sources. Full sky imagers, for example, while providing unmatched time and spatial resolution of cloud moments in a particular area, are relatively expensive to manufacture and maintain and required significant computational resources to manage the output data stream. By focusing on endogenous data only, we hope to develop methods that scale better to many PV systems in many regions.

3 Dataset and Features

3.1 Data acquisition

The data for this project were obtained under NDA with SunPower Corporation as part of the DOE-funded Visualization and Data Analytics for Distributed Energy Resources (VADER) program[1]. The data were provided as uncleaned and unverified raw data files for all available systems monitored by SunPower in a specific geographic region in Southern California (exact location information is protected by NDA). The request included all data from all sites in the region going back to the beginning of 2014. Some of those sites came online some time later, or were shut down at some point between 2014 and now, or suffer from large amounts of missing

[1]`https://gismo.slac.stanford.edu/`

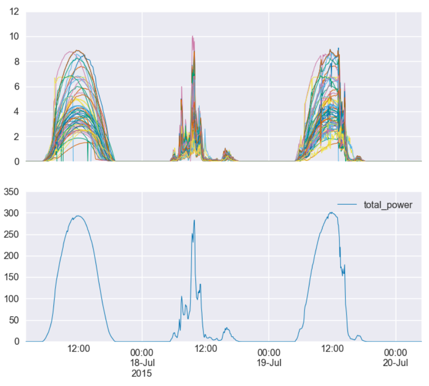


Figure 1: A selection of three days from the data set, showing the individual system power (top) and the aggregate regional power (bottom). Both y-axes are in kilowatts.

data. Therefore, the first step of this project was understanding the quantity and quality of data provided by SunPower, and selecting specific sites and systems for use in the analysis. First, we converted the files from Gzipped CSV files to Python-native Pickle files with hard-enforced data types, null-value conversion, and timestamp indexing, which allows for rapid loading of the individual files. Next, we wrote a function to scan over all files and extract the following pieces of information:

- First time stamp
- Last time stamp
- Number of non-null data points in the `ac_power` column
- Maximum, minimum, average, and standard deviation of the `ac_power` column

The statistics on the AC power column identified a small subset of sites that were using "Watts" instead of "kW" as units of power, which we standardized from this point on. We filtered for sites that had data through the end of the requested time period and a start date at least two years earlier, and we confirmed that all sites in this subset had >100,000 non-null AC power data points during that time. 62 sites out of 218 were selected through this process. Because sites can have one or more system (solar inverter) associated with them, the total number of systems is 73. 8 systems were subsequently dropped from the data set due to data errors. The AC power data from these remaining 65 systems were considered the feature data for our forecaster models. In Figure 1, we present a selection of three days from the time series data for individual sites and the resulting aggregate power.

3.2 Train-dev-test splitting

Our final dataset spans two years from 7/15/2015 to 7/14/2017. We split the data chronologically into training, development, and test sets, with 500 days (7/15/2015–11/25/2016), 110 days (11/26/2016–3/15/2017), and 121 days (3/16/2017–7/14/2017) respectively.

3.3 Featurization

Discussion of how we featurize the data is complicated by the fact that different models interact with the data differently. In general at a given time t, we observe a "historical window" of data ending at t, and we try to map that to a "future window" starting at t. For all models, the future window is 3 hours of 5-minute data, so the predictions are vectors in $\mathbb{R}^{36}$. The historical window is as small as one single observation for the persistence model and three hours of observed data for the neural net model. This is further complicated by the fact that two of the models–persistence and ARIMA–take aggregate power as an input (one-to-one forecasting), while our implementations of k-nearest neighbors and neural nets use all individual system power signals as inputs. Table 1 summarizes the different featurization methods. Finally, for the neural network models, the feature data are centered and scaled to have a mean of zero and a variance of 1, and the feature set was extended to include the day-of-year (DoY) and time-of-day (ToD) of the beginning of the forecast.

Forecasting model	Historical Window	Feature Dimension
Persistence	5min	1
ARIMA	2.5hr	30
k-NN	3hr	2340
Neural networks	3hr	2340

Table 1: Feature selection for models considered in this study.

3.4 Clearsky detrending

After our initial attempts with the forecaster models described in the next section, we concluded that removing the periodic trends present in the observed data would likely improve forecast accuracy. We achieved this goal by estimating the "clearsky" performance of each system over the course of the year, and subtracting the observed power signal from the clearsky signal.

Briefly, the clearsky signal is estimated by forming the time-series signal into a data matrix, with individual days of data split into columns. Then singular value decomposition is performed on this matrix:

$$M = UDP \tag{1}$$

The left singular vectors in matrix U represent an orthonormal basis for every individual day of data in the dataset. Taken together, the singular values and the the right singular vectors give us the linear combination of left singular vectors needed to reconstruct any day in the dataset exactly. We treat the rows of matrix DP as daily time-series signals, each associated with a particular left singular vector. Finally, we do statistical fitting of these daily signals for the top 5 singular values. An example of the fits of the daily signals for the top two singular values are shown in Figure 2. More detail on this method, including specifics of the statistical fitting performed on the daily signals, will be published in the coming year. In Figure 3, we illustrate the clear sky fits obtained with this algorithm for two days in summer. These two days are from the same week, and they illustrate the successful fitting of a clearsky day and the reconstruction of the clearsky signal on a cloudy day. The detrended data is the difference between this clearsky signal and the observed power signal.

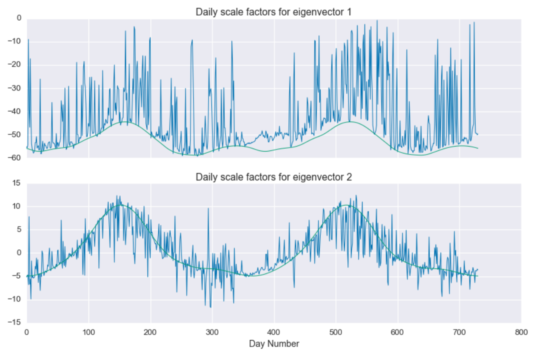


Figure 2: Estimating the clearsky signal by fitting the daily scale factors for the left singular vectors associated with the top two singular values for a single system in the dataset. This process was repeated for all systems and for the summed power signal.

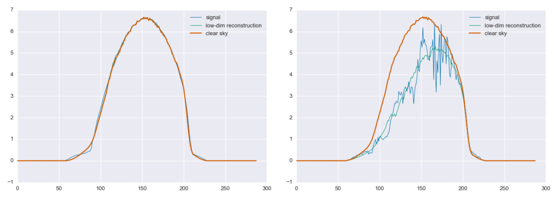


Figure 3: The statistically derived clearsky signal for a single system on a clear day (left) and a cloudy day (right) in the summer.

4 Methods

4.1 Persistence model

We implemented a persistence forecast model to provide a baseline for all other forecasters. In this case, a persistence forecast means selecting the last measured value and assuming all future values in the forecast horizon are exactly the same. We implemented the persistence forecast using an ARI (autoregressive integrative) model of order (1,1). This was implemented using the `SARIMAX` function from the Statsmodels Python package (Seabold and Perktold, 2010), setting the order to (1,1,0), with no seasonal or exogenous factors. The persistence model takes the summed power as an input and gives summed power as an output; it does not utilize any information from the constituent system AC power signals.

4.2 Auto-regressive models

Auto-regressive (AR) models express the constraint that the current value of a process can be expressed as a finite, linear combination of previous values, as shown in Equation 2. $\theta_1, \ldots, \theta_p$ are the parameters of the model, c is the constant or bias term, and ϵ_t is white noise.

$$X_t = c + \sum_{i=1}^{p} \theta_i X_{t-i} + \epsilon_t \tag{2}$$

AR models are a special case of the more general class of ARIMA models, which include moving-average and integrative terms as well. The moving average component models the regression error as a linear combination of past error terms, while the integrative component replaces the data values with differences between pairs of values, whic accounts for non-stationary processes.

Also implemented using `SARIMAX`, we explored a parameter space of autoregressive parameters between 1 and 30 (the number of lagged terms in the model), moving average parameters between 0 and 2 (the number of lagged error terms in the squared model), and integrative parameters between 0 and 2 (the number of differencing steps, deals with non-stationary data). The data are stationary (Sulaiman et al., 1997), but integrative parameters were explored for completeness and to observe the effect on forecasts. In the end, a simple AR model performed the best on the development set, and the best AR parameter p was emprically found to be 24 for the original data and 30 for the detrended data. Seasonal and exogenous parameters were also attempted, though not systematically, with no observed model improvement and significant increase in parameter space and therefore train time. Like the persistence model, this classical statistical time-series analysis takes the summed power as an input and gives summed power as an output.

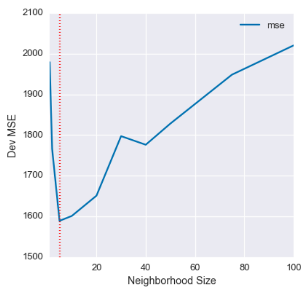
4.3 k-NN functional regression

To utilize the individual AC power signals in a predictive model, we implemented a k-nearest neighbors method for functional regression that takes a window of data from all systems as features, and returns the summed power as an output. This model was inspired by the work of (Ciollaro et al., 2014) that was covered in problem set 1. Rather than predict one half of a spectrum from the other half, however, we predict a forecast horizon signal based on the historical window across all systems. This is a nonparametric method that uses the Euclidian distance between an observed data point and the points in the training data set to select a neighborhood of similar observations. Then, a prediction is made by averaging the associated future windows for each data point in the neighborhood. We performed a parameter search over the neighborhood size which led us to choose 5 neighbors based on the mean squared error metric for the original data and 40 neighbors for the detrended data, as shown in Figure 4. No special software was used for this model besides Numpy (van der Walt et al., 2011) and Pandas (McKinney, 2010).

4.4 Neural networks

We have explored various fully connected deep neural networks from a few hundreds to a couple of thounsands of neurons with the help of TensorFlow™ by Abadi et al. (2015) and Keras by Chollet et al. (2015). The exploration space ranged from networks with 2 hidden layers up to 5 hidden layers, 100 neurons up to 4000 neurons per layer, learning rates from 1e-6 up to 1e-1, and finally levels of regularization from 1e-5 up to 1e0 with both L_1 and L_2 norms.

By comparing the different learning curves obtained for each



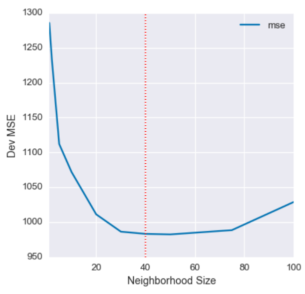


Figure 4: Selecting the neighborhood size for k-NN functional regression. Original data on the left, clearsky detrended data on the right.

architecture + learning rate + regularization combination on both training and validation sets, we selected a fully connected architecture in the middle range with two hidden layers. The first and second hidden layers had 2000 and 1000 neurons, respectively, and both used ReLU activation. The output layer with linear activation was used to compute a mean squared error loss. Also from the exploration, we selected L_2 regularization ($\lambda = 0.0001$) and trained with a learning rate of 1e-2.

The neural network digests training examples in the forward pass to produce an estimate of the form $\hat{y} = W^{[3]} f\left(W^{[2]}\ f\left(W^{[1]}x + b^{[1]}\right) + b^{[2]}\right) + b^{[3]}$ with f the ReLU activation and $W^{[l]}$ and $b^{[l]}$ the weights and biases. It then performs backpropagation of gradients based on the mean squared error loss $l = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{36}\left(y_{ij} - \hat{y}_{ij}\right)^2$ computed on batches of size $B = 500$. Mini-batches were used during training for both computational reasons and because the training data in this study does not fit in computer memory[2].

5 Results and Discussion

5.1 MSE analysis

For computational efficiency, a small test set consisting of 8 hand-picked days (from the larger test set decribed in section 3.2) with a mix of sunny and cloudy weather is used to evaluate and compare model performance. We created the three hour forecasts at the top of each hour in the small test set. We used the mean squared error (MSE) between the forecasts and the observed total power signal during daylight hours as our error metric, with a smaller value indicating overall more accurate forecasts for that model type. A summary of MSE values on the small test set is given in Table 2. Note that the MSE on the detrended data is calculated after transforming the detrended predictions back into actual power, so the numbers are directly comparable.

After detrending the power signals, all forecasters saw improvements in test error. Most striking is the improvement for the persistence model, which saw a 71% reduction in MSE and a lower error using detrended data than either the AR or the k-NN models were able to achieve on the original data.

[2]While the time-series data fits in memory, the featurization process causes a large expansion on the number of stored values. This is because a vector in $\mathbb{R}^{2340}$ is generated for each time step in the test set.

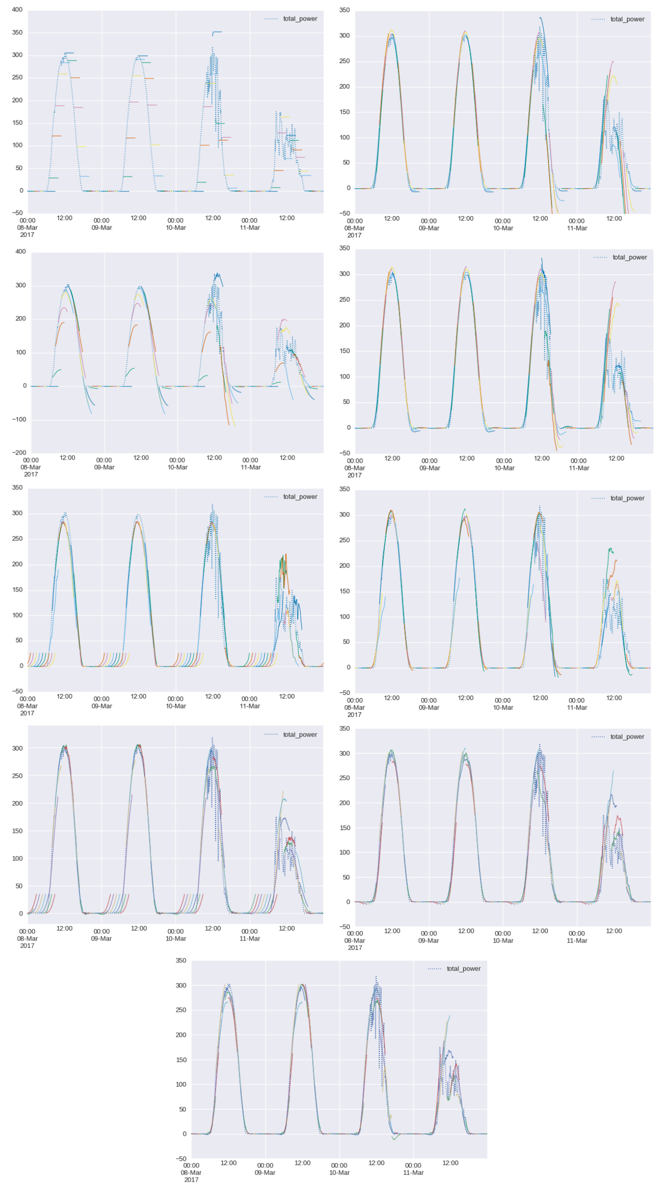


Figure 5: Comparison of forecasts with persistence model (1st row), AR (2nd row), k-NN (3rd row), and NN (last two rows). The left column shows the results for the original data, while the right columns shows the results for the detrended data. The last row corresponds to including DoY and ToD features.

We also observe that the addition of DoY and ToD for the NN model resulted in an additional 24% reduction in MSE over working with the detrended data.

5.2 Visual inspection of forecasts

As described in the previous subsection, 3-hour forecasts were executed every hour over the test set. We developed code to plot these forecasts, overlaid on the actual observed total power signal. This technique allows us to qualitatively assess the quality of our forecasters, as shown in Figure 5.

All models had difficulty dealing with the day/night and night/day transitions, when working with the original data. For k-NN and NN, the models are not able to distinguish between nighttime data that preceeds a forecast window that

Table 2: Mean Squared Error (MSE) on small test set of the best model of each type.

Model	Original Data	Detrended Data	Detrended +DoY,ToD
Persistence	5635.0	1592.3	-
AR	3021.9	1227.5	-
k-NN	1658.2	1434.1	-
Fully Connected NN	1045.5	672.4	510.7

is also nighttime, versus preceeding a forecast window that includes sunrise. This results in the "standing hairs" pattern observed before and during sunrise. This issue is eliminated by working with the detrended data.

It is notable that a simple k-NN functional regression model works as well as it does. The input features to this model are of a fairly high dimension, so we would expect the curse of dimensionality (Bellman, 1957) to make the task of selecting an appropriate neighborhood difficult. However, we can see qualitatively that the functional regression model is generally able to distinguish between upcoming clear and cloudy behavior.

The best performing NN model with DoY and ToD features is able to predict the upcoming 3-hour time series quite well. During times of variable power, the model tends to predict the time averaged value up the upcoming data, which is a very positive result.

6 Conclusions and Next Steps

We have shown that it is possible to produce reasonable forecasts of 5-minute aggregate PV power data on a 3-hour horizon, using only endogenous power data. This work is unique compared to other similar work in the complexity of the forecast and the limiting of exogenous data features.

Moving forward, there are three areas we will address: One, do a better job of predicting the time-series power during highly variable time-frames. Two, include an estimate of prediction confidence along with the expected power signal. Three, rigorously compare the methods developed during this project with other approaches.

To continue to improve prediction accuracy, we will look at forecaster model improvements as well as cost function improvements. ARIMA been shown to outperform NNs on some timescales (Reikard, 2009), so we will investigate refinements there, as well as ARIMA-NN hybrids. In addition, we will explore convolutional and recurrent NN topologies; long short-term memory (LSTM) NNs in particular look very promising for multi-step time-series forecasting (Olah, 2015). We will also investigate the potential to improve our forecaster model by including the spatial location of the inverters into account, and exploit the fact that the clouds approach the region from a certain angle and with a certain velocity. We hope to capture this velocity information based on lag analysis and geostatistics. Finally, we will explore alternate cost functions than overall MSE that give more credit for accurately identifying upcoming ramp events.

We are interested in exploring methods for capturing the uncertainty of the forecasts. One option is to include a metric like rolling total variation (the sum of the absolute value of all one-step differences in a window) to capture information about cloudiness and have a forecaster predict that value in addition to expected power. Another option is to track forecast stability over a sliding window.

Additionally, we will develop rigorous side-by-side tests to vet the performance of our best performing model. Specifically, we will implement the one-to-one forecasting approach with the NN models and compare to the accuracy of the many-to-one approach. In addition, we would like to compare the performance of our net to other authors' work on the simpler problem of 1- and 2-step forecasting, which, as mentioned in section 2, is much more common in the literature.

The methods developed during this project, in particular the application of clearsky detrending and the neural net implementation, show very promising results on the difficult problem of forecasting 36 values over three hours. We look forward to continuing this research and further developing highly accurate multi-hour forecasters for aiding in the large-scale integration of PV power in our electrical distribution system.

Contributions

Júlio: Implementation of the neural network model in TensorFlow™ and Keras, code development for submitting various architectures and hyperparameter configurations to TensorBoard™ for live assessment of learning curves and parameter pruning, code development of a reusable class for managing TensorFlow graph and organizing neural net training and predictions.

Bennet: PV performance background, statistical periodic detrending via SVD, data set selection and preprocessing, persistance model implementation, ARIMA model implementation, functional k-NN model implementation, standardized MSE analysis implementation, plotting function implementation, development of class for managing data loading, splitting, detrending, and retrending.

References

Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

R.E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

Chi Wai Chow, Bryan Urquhart, Matthew Lave, Anthony Dominguez, Jan Kleissl, Janet Shields, and Byron Washom. Intra-hour forecasting with a total sky imager at the UC San Diego solar energy testbed. *Solar Energy*, 85(11):2881–2893, 2011. ISSN 0038092X. doi: 10.1016/j.solener.2011.08.025.

Mattia Ciollaro, Jessi Cisewski, Peter Freeman, Christopher Genovese, Jing Lei, Ross O'Connell, and Larry Wasserman. Functional regression for quasar spectra. *Annal of Applied Statistics*, 2014.

Rich H. Inman, Hugo T.C. Pedro, and Carlos F.M. Coimbra. Solar forecasting methods for renewable energy integration. *Progress in Energy and Combustion Science*, 39(6):535–576, 2013. ISSN 03601285. doi: 10.1016/j.pecs.2013.06.002.

Wes McKinney. Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, pages 51–56, 2010.

Christopher Olah. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. Accessed: 12/14/2017.

Sophie Pelland, Jan Remund, Jan Kleissl, Takashi Oozeki, and Karel De Brabandere. Photovoltaic and Solar Forecasting: State of the Art. IEA PVPS Task 14, Subtask 3.1 Report IEA-PVPS T14-01: 2013. *International Energy Agency: Photovoltaic Power Systems Programme*, 2013.

Richard Perez, Sergey Kivalov, James Schlemmer, Karl Hemker, David Renné, and Thomas E. Hoff. Validation of short and medium term operational solar radiation forecasts in the US. *Solar Energy*, 2010.

Gordon Reikard. Predicting solar radiation at high resolutions: A comparison of time series forecasts. *Solar Energy*, 83(3):342–349, 2009. ISSN 0038092X. doi: 10.1016/j.solener.2008.08.007.

Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

Raffi Sevlian and Ram Rajagopal. Detection and statistics of wind power ramps. *IEEE Transactions on Power Systems*, 2013.

J. Smagorinsky. On the numerical integration of the primitive equations of motion for baroclinic flow in a closed region. *Monthly Weather Review*, (January 1958):3–8, 1958.

M. Yusof Sulaiman, W. M.Hlaing Oo, Mahdi Abd Wahab, and Z. Abidin Sulaiman. Analysis of residuals in daily solar radiation time series. *Renewable Energy*, 11(1):97–105, 1997. ISSN 09601481. doi: 10.1016/S0960-1481(96)00110-3.

Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13:22–30, 2011. doi: DOI:10.1109/MCSE.2011.37.