Solar-TK: A Data-driven Toolkit for Solar PV Performance Modeling and Forecasting

Noman Bashir[1], Dong Chen[2], David Irwin[1], Prashant Shenoy[1]
[1]*University of Massachusetts Amherst,* [2]*Florida International University*

***Abstract*—Solar energy capacity is continuing to increase. The key challenge with integrating solar into buildings and the electric grid is its high power generation variability, which is a function of many factors, including a site's location, time, weather, and numerous physical attributes. There has been significant prior work on solar performance modeling and forecasting that infers a site's current and future solar generation based on these factors. Accurate solar performance models and forecasts are also a pre-requisite for conducting a wide range of building and grid energy-efficiency research. Unfortunately, much of the prior work is not accessible to researchers, either because it has not been released as open source, is time-consuming to re-implement, or requires access to proprietary data sources.**

To address the problem, we present Solar-TK, a data-driven toolkit for solar performance modeling and forecasting that is *simple*, *extensible*, and *publicly accessible*. Solar-TK's *simple* approach models and forecasts a site's solar output given only its location and a small amount of historical generation data. Solar-TK's *extensible* design includes a small collection of independent modules that connect together to implement basic modeling and forecasting, while also enabling users to implement new energy analytics. We plan to *release* Solar-TK as open source to enable research that requires realistic solar models and forecasts,and to serve as a baseline for comparing new solar modeling and forecasting techniques. We compare Solar-TK's simple approach with PVlib and show that it yields comparable accuracy. We present three case studies showing how Solar-TK can advance energy-efficiency research.

I. Introduction

The installed capacity of solar photovoltaic (PV) energy, which converts the Sun's light into electricity, has been growing at near an exponential rate since the early 1990s. Over this period, the average time required to double the world's installed solar capacity has been approximately 2.4 years [1]. At this rate of growth, solar is forecasted to generate more than half of the world's electricity by 2050 [2]. However, achieving such high rates of solar generation requires addressing numerous research challenges. In particular, the key complicating factor with integrating solar into buildings and the electric grid is its high power generation variability, as the grid must balance electricity's supply and demand in real time despite these variations. Balancing the grid under a high solar penetration may require buildings to shift their energy usage over time using demand response or energy storage. A high solar penetration also complicates the grid's scheduling of conventional generators to offset any deficits in solar generation. This scheduling typically occurs one day in advance based on highly accurate forecasts of next-day load, which are largely a function of the forecasted temperature and the resulting energy used for heating and cooling. Unfortunately, solar forecasts are much less accurate than load forecasts, primarily because cloud cover forecasts, which significantly affect solar output, are much less accurate than temperature forecasts.

Given its importance, there has been significant prior work on solar performance modeling [3], [4] and forecasting [5] that infers a site's current and future solar generation. Solar performance modeling infers a site's average solar power output at the current (or a past) time (over some interval) based on *known* conditions, while solar forecasting predicts a site's average solar output at a future time (over some interval) based on *forecasted* conditions. Prior work generally separates solar irradiance estimation and forecasting from solar power modeling. The focus of the former is accurately estimating the current and future ground-level solar irradiance. The latter then typically assumes these solar irradiance estimates as input, and focuses on determining how much of that irradiance a solar site converts to electrical power. There have been hundreds of papers written in both areas, as well as books [6], that develop highly detailed solar modeling and forecasting methods. Solar modeling and forecasting remains an active area of research.

Unfortunately, much of the research above is not accessible to researchers outside the area, either because it has not been implemented and released as open source, is too complex and time-consuming to re-implement, or requires access to proprietary data. A motivation for Solar-TK is that developing more accurate solar performance models and forecasts is not an end unto itself. Realistic site-specific solar performance models and forecasts are a pre-requisite for conducting a wide range of research in building and grid energy-efficiency that is only indirectly related to solar energy. For example, accurate solar models and forecasts are a pre-requisite for developing better control policies for home batteries [7], [8]. Similarly, solar models and forecasts can be used to improve building demand response by shifting its energy consumption to better align with solar generation [9]. Recent work also requires accurate solar models and forecasts to assess the potential for sharing surplus solar energy among neighbors in developing countries [10]. Importantly, in these cases, solar modeling and forecasting is not the primary focus of the research, but is instead simply a tool the researchers require to make forward progress.

To address the problem, we present Solar-TK, an open data-driven toolkit for solar performance modeling and forecasting that is *simple*, *extensible*, and *publicly accessible* at http://traces.cs.umass.edu/index.php/main/Solar-TK. A key goal of Solar-TK is to be *simple* to use by researchers that require realistic and accurate solar performance models and forecasts, but are not experts in these areas. As a result, to

generate its solar model and forecasts, Solar-TK requires users to supply only a solar site's location and a small amount of historical generation data. This approach differs from existing open source modeling frameworks, such as PVlib [3], which requires users to supply detailed information about a solar site to model it, e.g., type of inverter, number of modules, module tilt, orientation, wiring, etc. Researchers may not have access to such detailed information, and, even if they do, may not have the expertise or time to develop and validate an accurate model.

Solar-TK's *extensible* design includes a small collection of independent executables that connect together using UNIX pipes to implement basic modeling and forecasting, while also enabling users to interpose on them to implement new energy analytics. As we show, these modules are independently useful and pluggable, enabling users to compose and extend them to implement a wide range of new and existing energy analytics. We plan to *publicly release* Solar-TK as open source to advance energy-efficiency research that requires realistic solar models and forecasts. In addition, Solar-TK can also serve as a useful baseline for benchmarking new and existing solar modeling and forecasting techniques. The current state of solar modeling and forecasting is similar, in some sense, to non-intrusive load monitoring (NILM), in that there has been decades of research and many proposed techniques, but few open benchmark implementations and public datasets. This prevents quantitative algorithmic comparisons. Indeed, Solar-TK is motivated by NILM-TK, a recent open source implementation of standard NILM algorithms with accompanying datasets to enable quantitative benchmarking of new and existing NILM algorithms [11].

There have been some recent efforts to develop and release open solar modeling and forecasting software, particularly PVlib [3]. Solar-TK differs from PVlib in numerous respects, as we discuss. Most importantly, Solar-TK primarily focuses on data-driven modeling, while PVlib focuses on physical modeling. As a result, PVlib's approach has the potential for much more accurate models and forecasts, but generally requires both deep access to a solar site and significant expertise to configure and implement its physical models. In contrast, Solar-TK is accessible to non-experts, requiring only a site's location and a small amount of historical generation data to develop its models. Solar-TK leverages many of the same physical models as PVlib, but calibrates their parameters automatically from historical generation data, rather than manually.

As we discuss in §II, Solar-TK's modules leverage the same well-known physical models of solar generation as PVlib [3], as well as recent work on data-driven "black box" solar performance modeling [12], [13], [14]. Thus, Solar-TK's contribution lies not in its modeling and forecasting approach, but in its simple, modular, and extensible design. Our hypothesis is that this design can provide researchers a simple way to develop accurate solar models and forecasts, and enable the implementation of new and existing solar energy analytics that can advance energy-efficiency research. In evaluating our hypothesis, we make the following contributions.

Solar-TK Design. We present Solar-TK's data-driven design, which develops a custom solar performance model using only a site's location and small amount of historical generation data. The design includes a small collection of independent modules that connect to implement basic modeling and forecasting, while enabling users to interpose on them to implement new energy analytics.

Implementation and Evaluation. We implement Solar-TK as a set of open-source software modules and sample data repository, and evaluate its accuracy by comparing against similar models developed using PVlib. We show that Solar-TK generally yields similar, and in some cases better, accuracy.

Case Studies. To demonstrate Solar-TK's flexibility, we leverage its extensible design to enable or advance recent research on solar energy analytics [15], disaggregation [12], and energy sharing [10].

II. Background

We briefly summarize many of the factors that affect solar power generation, and the physical modeling approach used by tools, such as PVlib [3] and PlantPredict [16],[1] to estimate and forecast it. Our summary is not exhaustive, as there are numerous physical models for many factors in the literature. Solar-TK's modules, which we discuss in §III, use some, but not all, of these same physical models.

Clear Sky Solar Irradiance. The clear sky solar irradiance—measured in W/m^2 represents the maximum power the Sun transmits to the Earth's surface under clear skies with no clouds. Clear sky irradiance is also referred to as global horizontal irradiance (GHI), which is the sum of direct normal irradiance (DNI) and diffuse horizontal irradiance (DHI). DNI is irradiance from direct rays of light, while DHI is irradiance from indirect light scattered by the Earth's atmosphere. Some physical models require DNI and DHI separately, since they can have different affects on solar output.

Clear sky solar irradiance is largely (but not entirely) a function of the Sun's position in the sky, which dictates the amount of the Earth's atmosphere sunlight must pass through to reach the ground. The more atmosphere sunlight must pass through, i.e., the lower the Sun's position in the sky, the less solar irradiance reaches the ground even under clear skies. Since the Sun's position in the sky is itself a function of location, i.e., latitude and longitude, and time (of the day and year), there are many simple models that accurately estimate clear sky solar irradiance for a location at any time [17]. Nearly all solar performance models and forecasts use clear sky solar irradiance models as their starting point, since solar power generation must be strictly less than the clear sky solar irradiance. There are numerous publicly-available software libraries that implement simple clear sky solar irradiance models [18].

Note that the simple models, while generally accurate, do ignore some important factors, such as the elevation of

[1]PlantPredict is proprietary solar modeling software.

the location and its surrounding landscape, and variations (and dynamic changes in) the Earth's atmosphere at different locations (even under clear skies). There are more advanced clear sky models that take these and other factors into account to improve accuracy, although they often require data sources that are not widely accessible as input, such as satellite measurements of the Earth's atmosphere at a location.

Weather Effects. Weather can reduce the solar irradiance that reaches the ground. The effect of weather on ground-level solar irradiance is typically quantified using the *clear sky index*, which represents the fraction $[0, 1]$ of the clear sky solar irradiance that reaches the ground. The primary weather metric that affects the clear sky index is the presence of clouds both overhead (which reduces DNI) and nearby (which reduces DHI). Other weather metrics, particularly temperature, also affect solar generation, but by decreasing power conversion efficiency and not ground-level solar irradiance. We discuss power conversion efficiency separately later.

Cloud cover is likely the most significant source of error in solar modeling and forecasting, largely because cloud cover is highly localized and difficult to measure. Unlike rain, ground-level radars cannot actively sense and track clouds. While there is work on precisely measuring cloud cover and forecasting short-term cloud movement using ground-level (sky) cameras [19], our focus is on solar modeling and forecasting that does not require physical access to a site. The clear sky index can be estimated from satellite data, but the methods are often proprietary [20], [21], and are actively changing due to upgrades in satellite sensing technology. As a result, satellite-based modeling is not widely accessible to researchers. Satellite-based models also have limitations in accuracy, as they can only measure the tops of clouds, and not their thickness, which significantly impacts the solar irradiance that reaches the ground.

The only public data on cloud cover that is widely available in the U.S. are ground-level cloud measurements taken at official weather stations. These measurements are in *oktas*, which represent how many eighths of the sky are covered in clouds, ranging from 0 oktas (completely clear sky) to 8 oktas (completely overcast). Okta measurements are typically made using a circular sky mirror divided into eight slices, such that when the mirror is placed on the ground, the oktas are equivalent to the number of slices with a cloud present [22]. Thus, oktas are a highly imprecise measure of cloud cover, and also do not account for cloud thickness or other properties. In addition, coarse okta measurements are typically translated to *even coarser* qualitative measurements when reported by weather stations, such as "Clear", "Partly Cloudy," or "Overcast." These qualitative measurements map directly to ranges of oktas [23]. For example, "Mostly Cloudy" is 5-7 oktas and "Overcast" is 8 oktas.

There are not many models for translating cloud cover measurements to a clear sky index. PVlib by default uses a simple linear transmittance model that assumes the cloud cover (in the range 0-100%) has a proportionate effect on the clear sky irradiance transmitted. It then combines this linear transmittance module with different models of GHI, DNI, and DHI to determine the clear sky index. Kasten and Czeplak proposed an empirical model based on data from hourly cloud cover observations and GHI measurements in Hamburg, Germany over a 10 year period (1964-1973) [24]. This model estimates the clear sky index as $(1 - 0.75n^{3.4})$ where n represents the fraction of cloud cover in the range $[0 - 1]$. Recent work refines this model by analyzing over 78k aggregate years of hourly data from over 11k sites [13]. The refined model empirically estimates the clear sky index as $(0.985 - 0.984n^{3.4})$.

Note that n in both models requires a fraction for cloud cover as input, while weather stations report an okta range. Solar-TK translates the reported range to a fraction by taking its average or by selecting a value randomly within the range, assuming any cloud cover is equally likely, and then dividing by 8. These inaccurate cloud cover measurements and models represent by far the largest source of error in solar performance modeling and forecasting. Note that PVlib only accounts for the effect of cloud cover in their solar forecasting model, and not their solar performance model, as some forecast data sources report cloud cover as a percentage.

Incident Irradiance. The tilt and orientation of a solar panel also affects the ground-level irradiance incident on it that can be converted to electrical power. The incident solar irradiance for a fixed panel is a well-known function of the Sun's azimuth and zenith angles, as well as the panel's tilt and orientation, as shown below. Here, I_{ghi} is the global horizontal irradiance, β is the tilt, ϕ is the orientation, Θ is the Sun's zenith angle, and α is the Sun's azimuth angle. The Sun's azimuth and zenith angles are themselves a well-known function of time for any location [25].

$$I_{incident} = I_{ghi} \times [\cos(90 - \Theta) \times \sin(\beta) \times \cos(\phi - \alpha) + \sin(90 - \Theta) \times \cos(\beta)] \quad (1)$$

Functions also exist for panels that track the Sun in one or both dimensions, although Solar-TK currently only supports fixed panels.

Solar Panel Type. Different types of solar panels have different designs and use different materials that affect their power conversion efficiency. For example, polycrystalline panels are slightly less efficient than monocrystalline panels. In addition, different designs also have different sensitivities to temperature and light conditions. While the power conversion efficiency generally decreases linearly with increases in cell temperature, different panel designs have different temperature coefficients, which dictate the slope of the efficiency decrease. Wind speed can also affect the cell temperature, since higher speeds dissipate the panel's heat more effectively (and can slightly reduce the cell temperature). PVlib includes detailed physical models for hundreds of solar panels, which capture their specific conversion efficiency under different light, temperature, and wind conditions. Of course, this library is not complete as new solar panels are frequently being released.

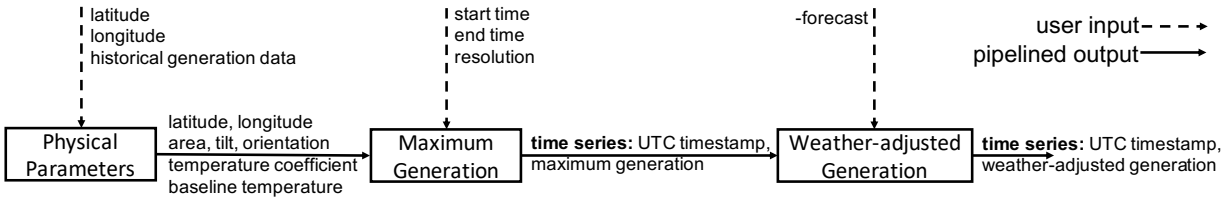


Fig. 1: A depiction of the pipeline for Solar-TK's three primary modules.

Electronics and Wiring. The electronics and wiring of a solar array also impact its power conversion efficiency. The power generated by a solar panel is a function of its operating voltage, as dictated by its I-V curve. The current the panel generates remains largely steady at low voltages, but begins to decrease sharply at some higher voltage. Since DC power is the product of current and voltage, there is a voltage that generates the maximum power. The shape of the I-V curve changes dynamically based on the incident irradiance and temperature, which are captured in the detailed solar panel models mentioned above. To maximize power, solar panels often include electronics that implement maximum power point tracking (MPPT) that dynamically adjusts the voltage to search for the maximum power point as the I-V curve changes. MPPT can occur in DC optimizers connected to panels or solar inverters.

Inverters convert DC solar power to grid-synchronous AC power, and incur some efficiency loss. These efficiency losses can vary as a function of the inverter's rated capacity and the solar power generation (in general, inverters are more efficient when operating near their rated capacity). Inverter capacity varies widely: small "micro" inverters may be connected to each panel, or large "string" inverters may be connected to an entire solar array wired in series or parallel. The wiring of solar panels into an array also affects output, as all panels wired in series are limited to generating the current of the panel with the minimum current. Note that when many solar panels are wired to a string inverter that implements MPPT, the tracking operates on a combination of the I-V curves of the individual panels based on the wiring. I-V curves of individual panels wired in series are additive in only the current dimension, while those wired in parallel are additive in only the voltage dimension.

PVlib includes physical models that capture the losses of many specific types of inverters under different conditions, and enables users to specify the wiring of solar panels to inverters that account for the relationships above. However, as above, this library is not complete, since new inverters are continuously being released.

Shading, Soiling, and Snow. In addition to clouds, shading from nearby objects, such as trees, buildings, and mountains, soiling from dust and mud, and snow can also prevent solar irradiance from reaching the panel. Physical models can capture shading effects given enough detail of a site based on the position of the objects, the Sun (at a particular time), and the solar panels. Offline solar models, such as PVlib and PlantPredict, generally estimate losses due to soiling from historical data. Online data-driven models, such as Solar-TK, can learn these effects from the data.

III. SolarTK Design

A solar performance model that solely uses physical models in the previous section, such as PVlib and PlantPredict, requires users to not only specify a site's location, i.e., its latitude and longitude, but also virtually configure the site. This includes identifying the specific type and number of solar panels and inverters, i.e., the specific brand and model, the tilt and orientation of each panel, the wiring of the panels together to one or more inverters (or DC optimizers), and potentially the characteristics of nearby objects that may shade the panels. Virtual configurations are time-consuming even for experts, and generally not possible without physical site access.

Virtually configuring a site is also challenging because any library of physical models for specific solar panels, inverters, and DC optimizers is necessarily incomplete given rapidly advancing technology. The newest components are often not in the library, which can decrease model accuracy for new (or proposed) sites that typically use newer components. Of course, users can always select components in the library that closely match their actual components. However, this is not necessarily as simple as selecting a component with a similar capacity rating, as the underlying physical model and behavior may be substantially different. As a result, to accurately model existing sites, users may need to try multiple components and select the ones that yield the highest accuracy.

Solar-TK entirely eliminates such virtual configurations, and instead *automatically* creates custom solar performance and forecast models for a site given its location and a small amount of historical data as input. Thus, users need not be experts to develop a solar performance and forecast model, and they do not require physical site access. Solar-TK's approach uses some of the simple physical models from §II, but calibrates their parameters based on historical data rather than requiring users to manually supply them. Solar-TK does not use many of the advanced physical modeling techniques supported by physical modeling frameworks, such as PVlib and PlantPredict. These advanced models are often too complex to calibrate from historical data, and, in many cases, the input data they require is not widely accessible. In addition, the inaccuracy of solar performance and forecast models is dominated by the inaccuracy of cloud cover measurements and models. Advanced physical models generally do not improve enough on the accuracy of the simple models to significantly improve overall accuracy.

```
$ parameters.py 42 -72 historical_data.csv
#latitude(°),longitude(°),area(m^2),tilt(°)
,orientation(°),c,T_base(°C)
42,-72,68,37,186,0.56,0.55
```

Fig. 2: Terminal output for physical parameters module.

Figure 1 depicts Solar-TK's design, discussed below, which consists of 3 primary modules that connect together via pipes to implement basic solar performance modeling and forecasting. These modules leverage both the physical models from the previous section, as well as recent work on "black box" solar modeling [12], [14], [13].

A. Physical Parameters Module

Figure 2 shows the terminal command-line and output of Solar-TK's physical parameter module. This module takes as input a site's latitude and longitude, as well as a csv file that includes some historical generation data. The format of the csv file is simple: the first column is a UNIX timestamp and the second column is the average power generation, in watts (W), from the corresponding timestamp to the next timestamp. The module can support any time resolution, and simply derives it automatically from the timestamps. However, the timestamps should represent the same resolution and be equally spaced in time. The module's output, as shown, is an estimate of a solar site's physical parameters, including its orientation, tilt, area, temperature coefficient, and a baseline temperature. The estimated array size (in m^2) simply assumes a 16% power conversion efficiency at 25C, which is an average efficiency for current solar panels. As discussed below, the approach can only accurately learn the product of size and efficiency, and not each individually.

The module leverages an algorithm from recent work to determine the parameters above [12], [14], which we summarize below. The algorithm computes the average clear sky irradiance (in W/m^2) at each timestamp (for the given resolution) of the historical data. The algorithm then uses the physical models from §II that dictate the amount of clear sky irradiance that is converted to electrical power based on various physical parameters. Specifically, it adapts Equation 1 from §II to include an unknown constant k that captures the product of a solar site's size and efficiency. This equation assumes that power conversion efficiency is constant (at a given temperature), and that all panels in an array have the same tilt and orientation. While these assumptions are not strictly true, as mentioned above, the resulting inaccuracy is not significant compared to the inaccuracy of cloud cover measurements and forecasts.

$$P(t) = I_{ghi}(t) \times k \times [\cos(90 - \Theta(t)) \times \sin(\beta) \times \cos(\phi - \alpha(t)) + \sin(90 - \Theta(t)) \times \cos(\beta)] \quad (2)$$

All the known parameters on the right side of the equation are a function of time, as labeled. The module includes functions to compute both the clear sky irradiance I_{ghi} (at the given location and time) and the Sun's azimuth and zenith angles (at the given time). The unknown parameters are k, β (or tilt angle), and ϕ (or orientation angle). The algorithm searches for values of these physical parameters such that the resulting output of the physical model at every timestamp is greater than or equal to the corresponding historical data. The algorithm searches for this tight upper bound because the solar generation should follow this function over time such that its output never exceeds some (unknown) maximum dictated by the clear sky irradiance. Ultimately, a single point dictates the tight upper bound and parameter values across the dataset (regardless of the amount of data). This point corresponds to the time with the lowest temperature (yielding the highest conversion efficiency) when the skies were clearest.

While there are many ways to conduct and optimize the search, we use an iterative brute force approach. The search first sets ϕ and β near their optimal values—180° (south) orientation and β equal to the latitude—and then iteratively searches for the value of k that yields the tightest upper bound while minimizing the root mean squared error (RMSE) with the data. We increase k by some amount (2 in our implementation) to provide some slack in the function when searching for the orientation and tilt. The result yields a function that is strictly "wider" and "higher" than the actual function due to a higher value of k and the optimistic setting of orientation and tilt. We then conduct a brute force search for the orientation angle that yields a tight upper bound but minimizes the RMSE. We next do the same for the tilt angle. We repeat this process 10 times or until the RMSE at each step does not change.

The algorithm above does not take into account the effect of temperature on power conversion efficiency. Prior work models this effect as being linear with the the ambient air temperature [12]. In reality, conversion efficiency is actually a function of the solar cell temperature, which is typically much higher than the ambient air temperature. However, the model assumes the relationship between the cell and air temperature is also linear, although the constant may differ between solar panels based on their design and materials. To adjust for temperature, we replace k above with a linear function $k'(t)$, which changes over time based on the temperature.

$$k'(t) = k \times (1 + c \times (T_{baseline} - T_{air}(t))) \quad (3)$$

Here, c is the constant linear temperature coefficient and $T_{air}(t)$ is the ambient air temperature at time t. The value of k is from above, while $T_{baseline}$ is the corresponding temperature at the datapoint that dictated the tightest upper bound, and thus k. Again, as above, we automatically search for the value of c that yields the tightest upper bound on the data. Ultimately, the tightest bounding c is dictated by two datapoints, the bounding datapoint above representing the clear sky period with the lowest temperature and another datapoint under clear skies with a different temperature. The module automatically fetches temperature at each location at the given time from the Internet, in this case Weather Underground. Note that the temperature data is hourly, so the module assumes that the temperature is constant within each hour when the given data resolution is higher. The module does not account for

the effects of wind speed, which dissipate heat that reduces the cell temperature. Again, we find the resulting inaccuracy small compared to the inaccuracy of cloud measurements and forecasts.

We evaluate the accuracy of the physical parameter module in §V as a function of the amount of historical data.

B. Maximum Generation Module

Figure 3 shows the terminal command-line and output of Solar-TK's maximum generation model. This modules takes as input the output of the physical parameter module above, specifically the site's location (latitude and longitude), k, $T_{baseline}$, c, ϕ, and β, as well as a start time, end time, and time resolution. The output, which is written to standard output, is a time-series (formatted the same as the csv file above) that estimates the maximum clear sky solar generation at those times. The module can support any time resolution. Note that output, by default, is UTC time.

The standard output of the physical parameter module can be piped directly to the maximum generation module as shown in Figure 3, or these parameters can be specified manually on the command-line. In the latter case, the parameters need not be generated by the physical parameter module based on historical data, and could just as well be derived from a physical site inspection (similar to a purely physical modeling approach). Size, tilt, and orientation parameters could also be inferred from satellite imagery, either manually as we do in §V or automatically [26], [27]. Alternatively, researchers could also supply their own parameter values to generate realistic, albeit synthetic, solar generation data for experiments.

The start time, end time, and time resolution must be specified on the command-line. The maximum generation module also supports a couple of options. Since the maximum generation is dependent on temperature, by default the module assumes the ambient temperature is always 25C. The module also supports a "-temperature" option that enables users to set a different static temperature. In this case, users may set the start time to any past or future value.

The module also supports using historical and forecasted temperature data. To use historical data, the "-real" option must be set, and the start time must be equal to or before the current time. In this case, the module fetches historical temperature data at each respective time from the Internet, i.e., Weather Underground. The module also accepts a "-forecast" option that instead fetches the temperature data that was forecasted at the start time from the National Digital Forecast Database (NDFD) [28]. If the start time is in the past, the module still retrieves the forecasted temperature data at that past start time and not the historical temperature data. NDFD forecasts only up to 7 days in advance. As a result, if the end time is more than 7 days past the start time, the module will only generate 7 days of forecasted values. Note that the resolution of forecasted weather data is every 3 hours. As above, we assume temperature is constant within each interval at higher resolutions.

```
$ parameters.py 42 -72 historical_data.csv |
 maxgen.py '2015-01-02 00:00:00' '2015-01-02
 23:00:00' '1h'
#latitude(°),longitude(°)
#42,-72
#time,max_generation
2015-01-02 05:00:00,0.0
2015-01-02 06:00:00,0.0
2015-01-02 07:00:00,0.0
2015-01-02 08:00:00,0.0
2015-01-02 09:00:00,0.0
2015-01-02 10:00:00,0.0
2015-01-02 11:00:00,0.0
2015-01-02 12:00:00,0.0
2015-01-02 13:00:00,0.0
2015-01-02 14:00:00,3167.490670450223
2015-01-02 15:00:00,6329.773863309609
2015-01-02 16:00:00,8285.330977204278
2015-01-02 17:00:00,9099.006104253323
2015-01-02 18:00:00,8887.269659227184
2015-01-02 19:00:00,7629.933318453599
2015-01-02 20:00:00,5506.951432379013
2015-01-02 21:00:00,2606.1217794935164
2015-01-02 22:00:00,8.27981536762782
2015-01-02 23:00:00,0.0
2015-01-03 00:00:00,0.0
2015-01-03 01:00:00,0.0
2015-01-03 02:00:00,0.0
2015-01-03 03:00:00,0.0
2015-01-03 04:00:00,0.0
```

Fig. 3: Terminal output for the max generation module.

The module's implementation is straightforward, as it simply uses the combination of Equations 2 and 3 below to directly compute an estimate of the maximum generation at each time t, given the parameters passed to it by the physical parameter module.

$$P(t) = I_{ghi}(t) \times k \times (1 + c \times (T_{baseline} - T_{air}(t))) \times [\cos(90 - \Theta(t)) \times \sin(\beta \times \cos(\phi - \alpha(t)) + \sin(90 - \Theta(t)) \times \cos(\beta)] \quad (4)$$

As we discuss in §VI, the maximum generation module is useful independently of the weather-adjusted module we discuss below.

C. Weather-adjusted Generation Module

Figure 4 shows the terminal command-line and output of the Solar-TK's weather-adjusted generation module. This module takes as input the time-series output of the maximum generation module (via a pipe) and adjusts it based on the cloud cover. To make this adjustment, the module simply multiplies each of the power values provided as input by the clear sky index. We estimate the clear sky index C using the reported cloud cover n measured in oktas as $(0.985 - 0.984n^{3.4})$ based on recent work [13].

The module's output is similar to Figure 3 with the first column being the UNIX timestamp, but with the second column being the weather-adjusted estimate of solar generation at the corresponding time. As above, the module also supports some optional parameters. The "-index" option adds a third column that is the clear sky index used to adjust the data. The "-forecast" option is the same as above, and outputs the forecasted weather-adjusted solar generation at the start time.

```
$ parameters.py 42 -72 historical_data.csv |
 maxgen.py '2015-01-02 00:00:00' '2015-01-02
 23:00:00' '1h' | weather.py
#time,adjusted_generation
2015-01-02 05:00:00,0.0
2015-01-02 06:00:00,0.0
2015-01-02 07:00:00,0.0
2015-01-02 08:00:00,0.0
2015-01-02 09:00:00,0.0
2015-01-02 10:00:00,0.0
2015-01-02 11:00:00,0.0
2015-01-02 12:00:00,0.0
2015-01-02 13:00:00,0.0
2015-01-02 14:00:00,75.84828650106284
2015-01-02 15:00:00,454.9143394032858
2015-01-02 16:00:00,722.4368854360703
2015-01-02 17:00:00,1044.2585273864622
2015-01-02 18:00:00,553.3818215991138
2015-01-02 19:00:00,6663.886529266351
2015-01-02 20:00:00,5424.343097076968
2015-01-02 21:00:00,2567.0299204095663
2015-01-02 22:00:00,8.155526468940131
2015-01-02 23:00:00,0.0
2015-01-03 00:00:00,0.0
2015-01-03 01:00:00,0.0
2015-01-03 02:00:00,0.0
2015-01-03 03:00:00,0.0
2015-01-03 04:00:00,0.0
```

Fig. 4: Terminal output for the weather-adjusted module.

Since cloud cover readings translate to an okta range, by default, the module use the average of this range. However, the module supports a "-seed" parameter that, if specified, selects an okta value within the range uniformly randomly. This is useful for supporting Monte Carlo simulations. Setting the seed to 0 causes the module to use a dynamic seed value based on the current time.

D. Shade-adjustment Modules

Solar-TK's current implementation includes the 3 primary modules above. As we show in §V, these modules have comparable accuracy to existing physical modeling approaches. However, they do not account for artificial shading from surrounding buildings, trees, and hills. These effects are unique to each site, and thus cannot be estimated using general one-size-fits-all physical models. As a result, we have a preliminary implementation of shade-adjustment modules that use machine learning (ML) to learn the unique shading effects of each site from historical data. We base our approach on one from recent work, which we summarize below [13]. This approach requires 2 modules: one for training and one for testing.

The shade training module takes as input the output of the weather-adjusted module above, as well as the site's location and a csv file with historical generation data covering the same time periods at the same resolution (similar to the physical parameter model above). The module checks to ensure the timestamps and resolution are consistent, and then transforms the data into a set of input features and a dependent output variable for model learning.

The input features are the Sun's azimuth and zenith angle at each datapoint, which, as discussed above, are a well-known function of location and time. The dependent output variable is the shading ratio of the observed solar generation at each time to the solar generation estimated by the weather-adjusted module. Thus, if there is no shading and the weather-adjusted module is accurate, the ratio should be 1. The more artificial shading, the lower the ratio. Assuming fixed objects, the degree of shading should be a function of the Sun's position. Unlike the previous modules, the training module requires a significant amount of data—up to a year or more—to learn an accurate model. Based on prior work [13], we use a support vector machine (SVM) with a radial basis function kernel to train the model. The training module writes the ML model's parameters to standard output, which can be redirected to a file.

The shade testing module takes as input the output of the weather-adjusted module, as well as a command-line parameter specifying a file that includes the model parameters (generated from the output of the training module). For each datapoint in the output of the weather-adjusted module, the module computes the Sun's azimuth and zenith angles at the respective time and feeds them to the ML model, which returns the shading ratio above. The module then multiplies the estimated average solar power at the datapoint with the shading ratio, and writes to standard output the UNIX timestamp and the shading-adjusted average solar generation.

E. Dynamic Factors

Many dynamic factors, such as dust and snow, can also affect solar generation that are not accounted for in the modules above. Since Solar-TK's modules are not computationally-intensive, they can be periodically re-run based on recent data to account for changing conditions. Note that the shade-adjustment module above is intended for semi-permanent shading from objects, and not highly dynamic shading from snow and dust. Thus, users should only need to train the shade-adjustment module once. Dynamic factors instead would affect the estimate of a site's size/efficiency, as Solar-TK would interpret dust and snow as a less efficient/smaller site.

IV. SolarTK Implementation

We implement Solar-TK in python. The implementation includesfive binary executables—parameters.py, maxgen.py, weather.py, shadetraining.py, and shade.py—that implement the modules discussed in the previous section. Our implementation extensively uses the pandas python data analysis library [29], and the NumPy scientific computing python library [30]. We use the clear sky irradiance library from the Pysolar python library [18], as well as an implementation of the PSA algorithm that computes the Sun's azimuth and zenith angles given a location and time [25]. We use weather data from Weather Underground and forecast data from the NDFD [28]. Finally, we use the pytz [31] and tzwhere [32] python libraries to adjust timestamps to the same timezone.

Reference Dataset. In addition to open-source code, SolarTK also includes a sample data repository, which include sample datasets for use with the software. A key aspect of Solar-TK is standardizing the data formats for our modules: this enables users to easily convert their internal datasets into formats

suitable for data ingestion by our modules and also enables other researchers to contribute open datasets to Solar-TK's data repository. Presently, Solar-TK's sample data repository contains historical solar generation for 150 solar sites in North America as well as historical weather and cloud cover for hundreds of locations. We plan to release the code and our sample data repository prior to publication.

V. Evaluation

We evaluate the accuracy of the physical parameter, maximum generation, and weather-adjusted generation module by comparing with both ground truth and PVlib [3]. Given its preliminary implementation, and that PVlib does not support shading, we do not evaluate the shade-adjustment modules. Our evaluation also focuses on solar performance modeling accuracy based on known conditions, and not forecasting. Both Solar-TK and PVlib use the same approach for modeling and forecasting, just that the forecasts use forecasted weather data rather than historical weather data. Thus, the solar forecast accuracy is largely a function of this forecasted weather's accuracy, and not the underlying approach.

We use data from 14 solar sites from a variety of regions, including in California, Colorado, Florida, Hawaii, Massachusetts, Texas, and Washington. For all the graphs, we use a one hour data resolution, since our weather and forecast data sources report an hourly resolution. Since we have physical access to only one solar site, only this site's virtual configuration in PVlib is accurate. Since PVlib did not have the solar panel type for this site in its library, we tried multiple different panels and selected the one that yielded the best results. For the other sites, we performed "best effort" PVlib modeling using satellite imagery and inferences from the data. Note that SolarTK's goal is not to improve upon PVlib's accuracy, but to achieve comparable accuracy with much less information.

Physical Parameter Module. Figure 5 shows the accuracy of the size (a), tilt angle (b), and orientation angle (c) estimated by Solar-TK's physical parameter module from one year of historical generation data. Note that we have physical access to Site 1 and thus directly measured its ground truth size, tilt, and orientation. For the other sites, the ground truth is an estimate of the size, orientation, and tilt from satellite and street view imagery. We do not compare with PVlib here, since it requires these physical parameters as input. These solar sites are single planes that do not connect panels with different tilts and orientations. Solar sites on complex roofs with different angles may decrease accuracy. As the graph shows, the estimated size, tilt, and orientation are highly accurate to within a few m^2 for size and to within a few degrees for the angles.

We also evaluate the accuracy of the estimates as the amount of historical generation data used as input increases, in this case only for Site 1 where we have ground truth. Figure 6 shows the number of days included in the input historical data (starting from July 1st) on the x-axis, and the Mean Absolute Percentage Error (MAPE) between the inferred metric and the ground truth for that metric on the y-axis. The graph has three lines, one for the size, tilt, and orientation. The graph shows that the parameters converge quickly within a few days, which is due to the first few days of July that year being sunny. The estimated size remains nearly unchanged when using more than a few days data. The tilt and orientation angles also converge to low MAPEs after using a few days of data, and continue to improve slightly as more data is used.

Maximum Generation Module. Figure 7 evaluates the accuracy of the maximum generation module for each site using both Solar-TK and PVlib. The y-axis is the MAPE between the site's estimated clear sky generation and the site's actual clear sky generation only during daytime hours. In computing the MAPE, we remove the sunrise and sunset hours, as these are effected the most by shade. In general, MAPE is a harsh metric when evaluating solar modeling accuracy, as it can result in high percentages for low absolute errors during periods of minimal solar generation. As a result, much of the prior work in this space does not use MAPE.

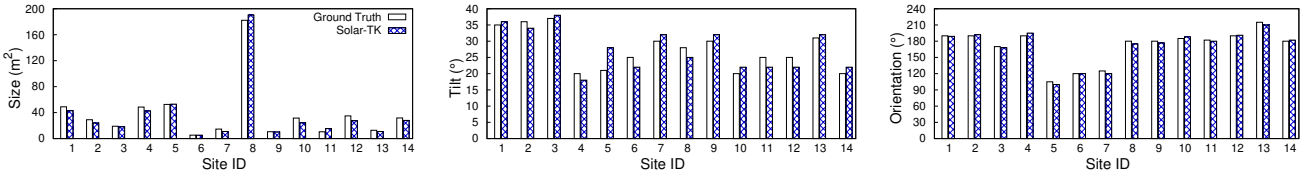
To derive the actual clear sky generation, we filtered out all time periods that were not under a clear sky. In this case, we defined a clear sky as those datapoints each month where the generation was within 90% of the maximum generation observed for that month at that time. This criteria assumes that within each month, every time of day observes at least one clear sky period, which is generally true for each of these sites. We also filter out the first and last hour of sunlight each day, as these hours only receive partial sunlight. After this filtering, each site had an average of 430 hourly datapoints.

The graph shows that, for each site, the MAPE under clear skies varies between 2% to, in the worst case, 18%. In most cases, the MAPE is well under 10%. Solar-TK's accuracy is comparable, and in many cases better than, PVlib's, although we reiterate that our PVlib virtual configurations for Sites 2-14 are best effort and not informed by physical access to the site. For Site 1, where we have physical access, the accuracy is comparable at $\sim$2%.

Weather-adjusted Module. Figure 8 evaluates the accuracy of our weather-adjusted generation module for each site using both Solar-TK and PVlib. For PVlib, we configured it to use its clearsky-scaling function, which does a linear adjustment of the clear sky irradiance to account for cloud cover. We evaluate the accuracy over one year of hourly data. In this case, we compute the MAPE between the actual solar generation each hour and the inferred generation from Solar-TK and PVlib's modules. The figure shows that the presence of clouds significantly decreases model accuracy in both cases compared to the maximum generation module. Again, Solar-TK and PVlib generate results with comparable accuracy, and in some cases, Solar-TK's accuracy is actually better.

VI. Case Studies

Since Solar-TK's underlying physical models and approaches are well-known and have been proposed in prior work [12], [14], [13], its primary contribution lies in its ease of use and modular design that researchers can interpose on or extend to implement new applications. In this section, we



(a) Size (m^2) (b) Tilt (°) (c) Orientation (°)

Fig. 5: Accuracy of the physical parameter module at estimating 14 solar sites' size, tilt angle, and orientation angle.

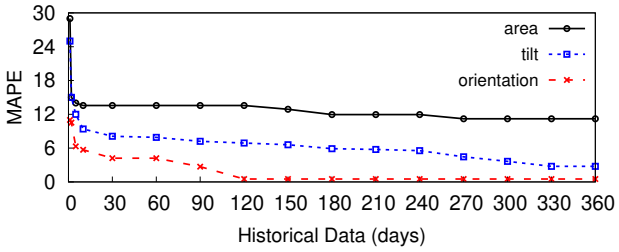


Fig. 6: The effect of data size on physical parameters.

describe 3 case studies of using different elements of Solar-TK to advance energy-efficiency research.

A. Computing the Clear Sky Index

Recent work proposes a clear sky photovoltaic (PV) index (K_{pv}) that uses solar generation data to estimate the clear sky index, which captures the fraction of the clear sky solar irradiance that reaches the ground (due to cloud cover) [15]. Recall that Solar-TK's weather-adjusted module simply multiplies the power estimated by the maximum generation module by this clear sky index, which it estimates from coarse cloud cover measurements. As Figure 8 shows, cloud cover introduces significant inaccuracy to the model.

Due to this inaccuracy, another recent approach has been to estimate a solar site's power based on the power generated by a nearby solar proxy site. Since the sites are nearby, their clear sky index should be similar. Thus, in this case study, we replace Solar-TK's weather-adjusted module with another similar module, which has the same input and output, but adjusts the maximum generation based on the K_{pv} computed at a nearby solar site. Computing a solar site's K_{pv} at any time using Solar-TK is simple, as it is just a solar site's actual solar output each time period divided by the output estimated by the maximum generation module.

Figure 9 shows the MAPE of this approach for three of our sites where we could find data from a nearby solar site within 50 miles. As shown, the MAPEs are significantly lower than those reported in Figure 8 and generally under 10%, indicating (unsurprisingly) that solar sites are a much better cloud sensor than cloud cover measurements in oktas. The accuracy is near that of the maximum generation model under clear skies. Recent work also uses data from nearby sites to detect solar anomalies [33]. Solar-TK with this approach could also be used for anomaly detection by detecting when the clear sky index (K_{pv}) at nearby sites was substantially different. This may indicate a fault at the site with lower K_{pv}.

B. Disaggregating Solar from Net Meter Data

There have been multiple different methods recently proposed that "disaggregate" solar generation from net meter data that is the sum of energy consumption and solar generation data [12], [34], [35]. These proposed methods are complex and generally require a significant amount of training data from either the solar site being disaggregated or other solar sites. Solar-TK's modeling approach leverages aspects of some prior work on solar disaggregation and thus enables a straightforward implementation using Solar-TK given some historical net meter data from a site [12].

In the historical net meter data, we first find the minimum power consumption at night and assume this is a home's base power consumption. Assuming power generation is positive, and power consumption is negative, we add this minimum power consumption to the net meter data to offset the effect of the base power consumption on the maximum solar generation that dictates our model. We then simply run the physical parameter module on this modified historical net meter data. The physical parameter module is able to derive accurate parameters as long as there is at least two points in the data under clear skies with a temperature difference, and where consumption is near the base consumption. The extra condition increases the historical data required to derive an accurate model.

After deriving the physical parameters, we simply use the maximum generation and weather-adjusted models as normal to estimate the site's solar generation, and then subtract it from the net meter data to estimate energy consumption. We could also replace the weather-adjusted module with the module above that estimates solar output based on a nearby site to improve accuracy, as done by a recent approach to solar disaggregation [35]. Figure 10 shows the MAPE of the solar power disaggregated from the 14 solar sites (which are all homes that also consume power) using our weather-adjusted module. As the graph shows, the MAPE using net meter data is similar to when using pure solar data across the 14 sites.

C. Estimating Solar Energy Sharing Potential

Our last case study focuses on recent work that analyzes the potential for solar energy sharing between neighbors in developing regions [10]. In these regions, small-scale solar-battery kits that include a single solar panel and a small car-sized battery have become popular for providing supplemental power, especially given the instability (or lack) of grid infrastructure. Unfortunately, given its small size, the battery often fills to capacity early in the day, at which point it

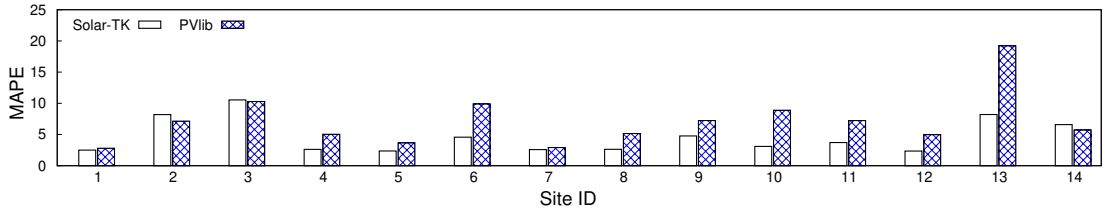


Fig. 7: MAPE for Solar-TK and PVlib maximum generation modeling modules for the 14 solar sites.

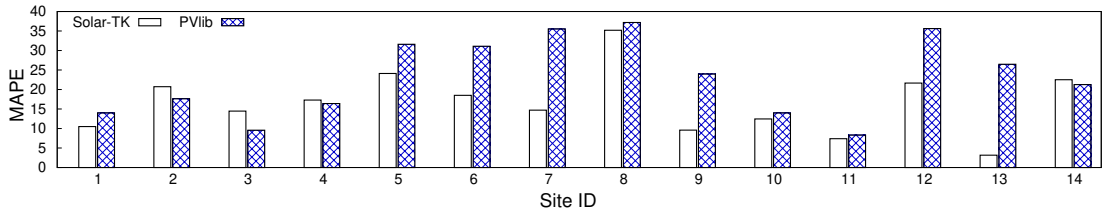


Fig. 8: MAPE for Solar-TK and PVlib weather-adjusted generation modeling modules for the 14 solar sites.

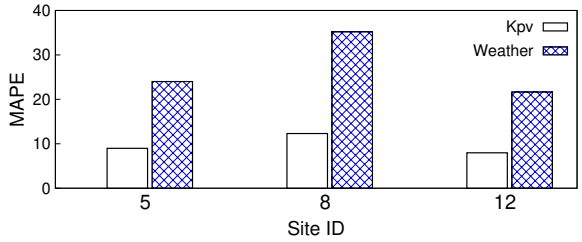


Fig. 9: Generation accuracy using K_{pv} and weather data

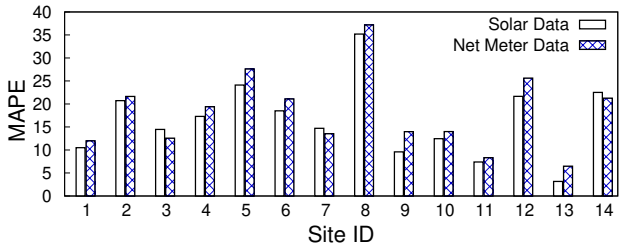


Fig. 10: Solar disaggregation accuracy of 14 sites.

wastes the additional solar power. The work explores the potential benefits of being able to share, rather than waste, this surplus energy with nearby neighbors by analyzing a large solar generation dataset from a solar-battery kit manufacturer.

The problem is that the data only reports the battery charging rate: when the battery charge is below capacity, it reveals solar generation, but when the battery is at capacity its charging rate is 0 (and thus does not reveal solar generation). As a result, the dataset does not convey the unrealized solar potential available for sharing. Thus, the work estimated the solar potential at each time period based on the average historical weather at that time period (over the past 20 years). Instead, Solar-TK can accurately model the incomplete data to enable an estimate of the actual solar potential at each time period. Doing so is straightforward, and simply requires feeding each site's data through the physical parameter, maximum generation, and weather-adjusted generation modules.

Figure 11 shows the MAPE between the ground truth data, and using Solar-TK to estimate potential, as well as using the 20-year average historical potential at that time

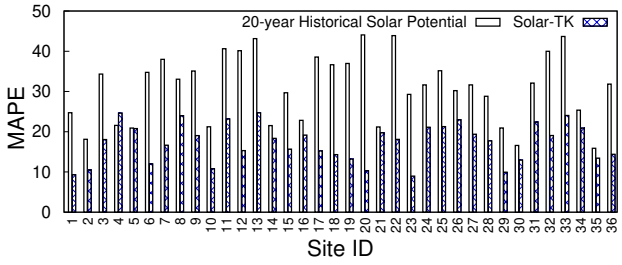


Fig. 11: MAPE using 20-year historical potential and Solar-TK for the 36 solar sites in Western Kenya.

(regardless of the actual real-time weather) for 36 solar-battery kit deployments. The ground truth only includes periods where the battery is not at capacity and we know the actual solar generation. As expected, the figure shows that using Solar-TK significantly reduces the MAPE and improves the accuracy of this data analysis. Since the dataset includes many nearby sites, the accuracy could be further improved by computing the K_{pv}, as described above, across all sites for periods when the battery is not at capacity. The weather-adjusted module could then be replaced by one that used that estimated potential when the battery was full based on the K_{pv} of the nearest site that did not have a full battery.

VII. Related Work

There have been many decades of research on solar modeling and forecasting using physical and empirical models. Solar-TK builds on and leverages much of this work. The underlying insights and algorithms that it uses are not novel. In particular, it implements elements of a variety of data-driven solar modeling approaches described in recent work [12], [13], [14], as well as uses many of the same physical models as PVlib [3] and PlantPredict [16]. SolarTK's novelty lies in both its simple design and decomposition of its basic modules, which, as we show in our case study, enables researchers to interpose on or extend them in useful ways. Solar-TK is usable by non-experts, and intended to advance energy-efficiency research that requires solar modeling and forecasting.

Solar-TK does not utilize as many of the physical models used by PVlib and PlantPredict. As a result, its accuracy potential is likely not as high. However, one insight of our work is that the inaccuracy is dominated by coarse and imprecise cloud cover measurements. Of course, SolarTK's weather-adjusted generation module uses very simple cloud cover reports. In §VI, we show how to replace this module with one that uses nearby solar sites to more accurately infer cloud cover. This module could also be replaced with one that infers the clear sky index using solar radiation data gathered from ground-level sensors or satellite measurements [6].

Solar-TK also supports solar forecasting using the same model, but using forecasted temperatures and cloud covers. We do not evaluate weather forecast accuracy here, as it can vary widely at different locations, but note that it is often another major source of inaccuracy (along with cloud cover measurements) that dwarf the inaccuracy from many advanced physical models. Of course, there is also a significant body of work on solar forecasting, which uses a wide variety of techniques at different time resolutions and horizons [5]. We do not claim that Solar-TK's forecasts are more accurate than these prior techniques. Our evaluation only shows that, given perfect weather forecasts, Solar-TK's accuracy should be no worse than the model accuracy presented in §V.

VIII. Conclusions

This paper presents Solar-TK, a data-driven toolkit for solar performance modeling and forecasting that is simple, extensible, and publicly accessible. Solar-TK's goal is to advance energy-efficiency research that requires accurate solar models and forecasts. We describe Solar-TK's design, which consists of independent modules that connect together to implement basic solar modeling and forecasting. We compare Solar-TK's approach with PVlib's pure physical modeling approach both qualitatively and quantitatively, and show that it yields comparable, and sometimes better, accuracy. We also use Solar-TK as part of three case studies to show how it could be used and extended by researchers. We plan to publicly release Solar-TK in the near future.

Acknowledgement

This research is supported by NSF grants CNS-1645952, IIP-1534080, CNS-1405826, CNS-1505422, and the Massachusetts Department of Energy Resources.

References

[1] "Growth of Photovoltaics," https://en.wikipedia.org/wiki/Growth_of_photovoltaics, Accessed June 24th 2019.
[2] "Technology Roadmap Solar Photovoltaic Energy," https://www.iea.org/publications/freepublications/publication/TechnologyRoadmapSolarPhotovoltaicEnergy_2014edition.pdf, International Energy Agency, Tech. Rep., 2014.
[3] W. Holmgren, C. Hansen, and M. Mikofski, "PVlib Python: A Python Package for Modeling Solar Energy Systems," *Journal of Open Source Software*, vol. 3, no. 29, 2018.
[4] "PV Performance Modeling Collaborative," https://pvpmc.sandia.gov/, Accesssed June 24th 2019.
[5] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. M. de Pison, and F. Antonanzas-Torres, "Review of Photovoltaic Power Forecasting," *Solar Energy*, vol. 136, October 2016.
[6] J. Kleissl, *Solar Energy Forecasting and Resource Assessment*. Academic Press, July 2013.
[7] B. Qi, M. Rashedi, and O. Ardakanian, "EnergyBoost: Learning-based Control of Home Batteries," in *e-Energy*, June 2019.
[8] A. Mishra, D. Irwin, P. Shenoy, J. Kurose, and T. Zhu, "GreenCharge: Managing Renewable Energy in Smart Homes," *JSAC*, vol. 31, 2013.
[9] J. Taneja, D. Culler, and P. Dutta, "Towards Cooperative Grids: Sensor/Actuator Networks for Renewables Integration," in *SmartGridComm*, October 2010.
[10] S. Correa, N. Bashir, J. Iglesias, C. Saffery, and J. Taneja, "Like a Good Neighbor, Solar is There," in *e-Energy*, June 2019.
[11] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava, "NILMTK: An Open Source Toolkit for Non-intrusive Load Monitoring," in *e-Energy*, June 2014.
[12] D. Chen and D. Irwin, "SunDance: Black-box Behind-the-Meter Solar Disaggregation," in *e-Energy*, June 2017.
[13] D. Chen, J. Breda, and D. Irwin, "Staring at the Sun: A Physical Black-box Solar Performance Model," in *BuildSys*, November 2018.
[14] D. Chen and D. Irwin, "Black-box Solar Performance Modeling: Comparing Physical, Machine Learning, and Hybrid Approaches," in *Greenmetrics*, June 2017.
[15] N. Engerer and F. Mills, "Kpv: A Clear-sky Index for Photovoltaics," *Solar Energy*, vol. 105, July 2014.
[16] "PlantPredict," https://plantpredict.com/, June 2019.
[17] J. S. S. Matthew J. Reno, Clifford W. Hansen, "Global Horizontal Irradiance Clear Sky Models: Implementation and Analysis," Sandia National Laboratories, Tech. Rep., March 2012.
[18] "PySolar," https://pysolar.readthedocs.io/en/latest/, Accessed June 2019.
[19] B. Urguhart, B. Kurtz, E. Dahlin, M. Ghonima, J. Shields, and J. Kleissl, "Development of a Sky Imaging System for Short-term Solar Power Forecasting," *Atmospheric Measurement Techniques*, vol. 8, 2015.
[20] "SolarAnywhere," https://www.solaranywhere.com/, Accessed June 2019.
[21] "SoDa - Solar Radiation Data," http://www.soda-pro.com/, Accessed June 2019.
[22] "NASA, Make a Sky Mirror to Observe Clouds and Contrails," https://mynasadata.larc.nasa.gov/science_projects/make-a-sky-mirror-to-observe-clouds-and-contrails/, June 2019.
[23] "Weather.gov Terms," http://www.weather.gov/bgm/forecast_terms, 2018.
[24] F. Kasten and G. Czeplak, "Solar and Terrestrial Radiation Dependent on the Amount and Type of Cloud," *Solar Energy*, vol. 24, no. 2, October 1980.
[25] M. Blanco-Muriel, D. Alarcon-Padilla, D. Lopez-Mratalla, and M. Lara-Coira, "Computing the Solar Vector," *Solar Energy*, vol. 70, no. 5, 2001.
[26] J. Yu, Z. Wang, A. Majumdar, and R. Rajagopal, "DeepSolar: A Machine Learning Framework to Efficiently Construct a Solar Deployment Database in the United States," *Joule*, vol. 2, no. 12, pp. 2605–2617, December 2018.
[27] S. Lee, S. Iyengar, M. Feng, P. Shenoy, and S. Maji, "DeepRoof: A Data-driven Approach For Solar Potential Estimation Using Rooftop Imagery," in *KDD*, August 2019.
[28] "National Digital Forecast Database (ndfd)," 2018. [Online]. Available: https://www.weather.gov/mdl/ndfd_home
[29] "Python Data Analysis Library," https://pandas.pydata.org/, Accessed June 2019.
[30] "NumPy," https://www.numpy.org/, Accessed June 2019.
[31] "pytz - World Timezone Definitions for Python," http://pytz.sourceforge.net/, Accessed June 2019.
[32] "tzwhere," https://pypi.org/project/tzwhere/, Accessed June 2019.
[33] S. Iyengar, S. Lee, D. Sheldon, and P. Shenoy, "SolarClique: Detecting Anomalies in Residential Solar Arrays," in *COMPASS*, June 2018.
[34] R. Mohan, T. Cheng, A. Gupta, V. Garud, and Y. He, "Solar Energy Disaggregation using Whole-House Consumption Signals," in *NILM Workshop*, June 2014.
[35] M. Tabone, S. Kiliccote, and E. Kara, "Disaggregating Solar Generation Behind Individual Meters in Real Time," in *BuildSys*, November 2018.