PVInsight Final Technical Report

Agency/Office/Program	DOE/EERE/Solar Energy Technology Office
Award Number	DE-EE00034368
Project Title	PVInsight Final Technical Report
Principal Investigator	Bennet E. Meyers, Project Scientist bennetm@slac.stanford.edu, 408-674-3908
Business Contact	Steve C. Chao, Financial Analyst scwchao@slac.stanford.edu, 650-926-3051
Submission Date	9/30/21
DUNS Number	
Recipient Organization	SLAC National Accelerator Laboratory
Project Period	10/1/2018–9/30/21
Project Budget	$1,390,000
Submitting Official Signature	

Acknowledgment. This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Solar Energy Technologies Office Award Number 34368.

Contents

1 Executive Summary

Motivation. Data generated from real-world photovoltaic (PV) systems represent significant opportunities for the industry—from digital operations and maintenance to real-time planning and forecasting. However, these data also come with substantial, unique challenges. A particular challenge is the analysis of *unlabeled* PV performance data, which we define as time-series measurements of real power production (or sometimes current or voltage) that do not have corresponding meteorological measurements (irradiance, temperature, etc.) or system configuration information (sometimes called system *metadata*). These challenges are amplified in the distributed rooftop sector, in which data quality, completeness, and metadata can be very poor.

At the same time, the distributed rooftop solar market is growing rapidly, with 3.2 GWdc of residential PV installed in the U.S. in 2020, the largest year on record [1]. While utility solar continues to contribute the majority of new installations, 46% of new installed capacity in 2020 (5.3 GWdc) were non-utility, distributed systems [1]. As such, it is critical for the industry and research community to develop analytical solutions that are robust to the unique data challenges for distributed PV data and that are scalable to fleet-sized analysis.

Goals. We seek to develop novel algorithms and software, which automate the performance and reliability analysis for photovoltaic (PV) systems with unlabeled performance data. This automated approach will drastically reduce the engineering work required to identify and quantify system loss factors. There are three major goals:

1. Data onboarding, cleaning, processing automation
2. Signal-processing-based loss factor identification and estimation
3. Estimation of missing system metadata (location and orientation)

Major accomplishments.

- Development of "optimization-based signal decomposition" (OSD) framework for estimating hidden signal components in time-indexed signals (possibly vector valued with missing data)
- Application of OSD to various performance and reliability subproblems (string failure analysis, clipping, analysis, soiling analysis, shade analysis...)
- Development of software tools (Solar Data Tools, Statistical Clear Sky, and PV System Profiler)
- Development of "statistical clear sky fitting" (SCSF) algorithm, that fits a clear sky baseline to unlabeled power data
- Peer-reviewed publication of methodology that uses SCSF to infer system degradation (or irradiance sensor drift) from unlabeled data
- Publication of latitude, longitude, tilt, and azimuth estimation methodology

2 Background

There are some similarities between this project and other open-source PV data analytics software projects such as RdTools [2] and Pecos [3]. In all three projects, the overarching goal is to streamline the extraction of useful information from time-series PV data. RdTools is more focused on automating certain performance analysis tasks, while Pecos is more focused on automating certain data onboarding, visualization, and cleaning tasks. PVInight covers both these topics. Below we discuss the simiarlities and differences between these two projects and PVInsight. Then, we also summarize recent research on aglorithms and statistical estimation that touch on similar topics to those in PVInsight. In all cases, PVInsight takes a unique approach to the analysis or estimation problem by only assuming access to *unlabeled* time-series data.

2.1 Software packages

RdTools. RdTools is an open-source library to support reproducible technical analysis of time series data from photovoltaic energy systems [2]. Originally focused on implementing a particular approach to analyzing long-term system degradation, the project has recently expanded into soiling rate analysis in addition to long-term degradation. Specifically, the software will assist the user with generating a *system performance index* [4] from labeled production data provided by the user. Having generated this performance index, the user may than request an analysis of degradation using a year-on-year analysis [5] or soiling using a recently proposed algorithm called "stochastic rate and recovery" [6]. In addition, the software automates some data onboarding and cleaning/filtering tasks.

The major difference between RdTools and the approach to degradation and soiling analysis taken in PVInsight is the requirement for labeled PV data, which is necessary to construct a performance index. The analytical approaches undertaken in RdTools all assume access to a (well-designed) performance index.

The PVInsight and RdTools teams have been collaborating closely over the project period; the teams co-authored a paper on degradation rate estimation that compares the PVInsight approach to the RdTools approach in 2020 [7]. As discussed in §7, the SLAC and NREL teams are joining forces in PVInsight Phase 2, to continue validate, compare, and contrast these differing methods to analyzing degradation and soiling.

Pecos. Pecos is an open source Python package designed to process large volumes of PV time-series data on a regular schedule and alert system operators when the system has changed [3]. The software can be used to automate a series of quality control tests and generate custom reports which include performance metrics, test results, and graphics. Similar to the PVInsight package Solar Data Tools, Pecos provides built in plotting functions that are useful for quickly visualizing PV power data. Both packages recommend the use of *heatmaps* to visualize multi-month (or longer) periods of sub-daily power measurements, as shown in figure 1. However, Solar Data Tools includes subroutines for automatically converting scalar time series into a format for viewing as a heatmap, including special consideration from missing data (see §4.2.1). As another similarity, both packages include functionality for detecting timestamp errors and have various methods for implementing data quality checks.

The functions in Pecos are intended to assist with automating user-defined data processing tasks; they do not directly provide insight into PV system health. No PV-specific

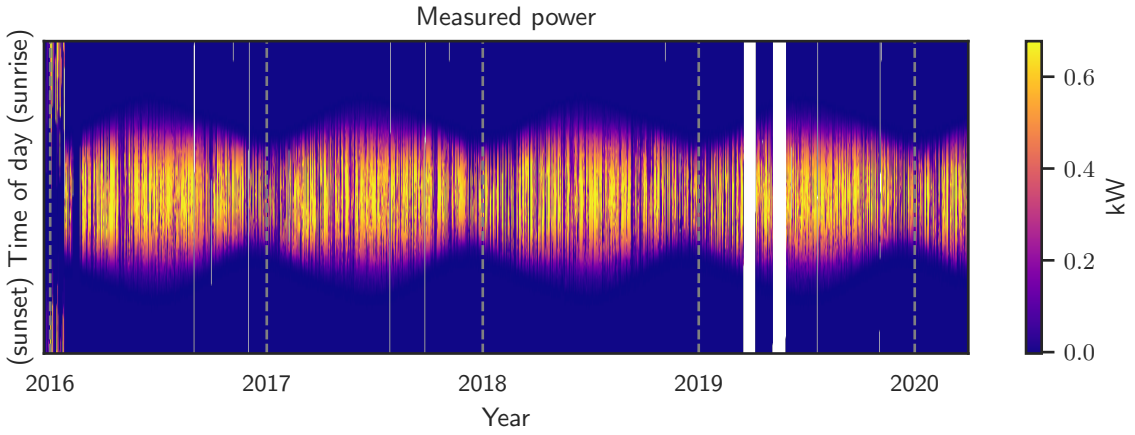

Figure 1: A heatmap view of power measurements over 4.28 years for a representative PV system, made with Solar Data Tools. White pixel represent missing data. Both Solar Data Tools and Pecos recommend the use of heatmaps for plotting long periods of power data.

performance analylsis algorithms are implemented (such as soiling loss estimation, for example); it would be up to the user to define such analytics in Pecos. Additionally, PVInsight is unique in its focus on developing and providing data onboarding and quality checking that are developed specifically for unlabeled PV data.

An illustrative example of the difference between Pecos and Solar Data Tools is the handling of data quality checks. Pecos provides an interface for the user to define their own quality check procedure (such as out-of-bounds check, or comparison to a model) which can be automated for running reports and generating dashboard. Solar Data Tools, in contrast, provides an automatic check in the form of an algorithm that fits a *data-driven baseline* that predicts the number of non-zero power measurements that are expected on a given day of the year (the baseline is 365-day periodic and smooth) and flags for user days that have too many or too few non-zero measurements, as shown in figure 2. (Solar Data tools also provides an interface for user-defined rules as well.) The smooth, orange trend in the figures is the estimated baseline for "daily signal density," or the fraction of non-zero values in a 24-hour period. The optimization-based signal decomposition (OSD) framework is used to estimate this baseline, which is discussed more in §4.1.

While there is some overlap in the general goals of the projects, PVInsight takes an approach that is more closely coupled to solving problems related to unlabeled PV data. So while both packages help to automate the data on-boarding, cleaning, and visualization process, PVInsight also presents *fundamentally novel algorithms and approaches* for providing automatic quality checks.

2.2 Algorithms and papers

Degradation analysis. A large number of researchers have proposed various approaches and methods for estimating the long-term degradation rate of a PV system from measured power data. During the course of the project, we participated in a large survey of such methods, led by the International Energy Agency Photovoltaic Power Systems Programme (IEA-

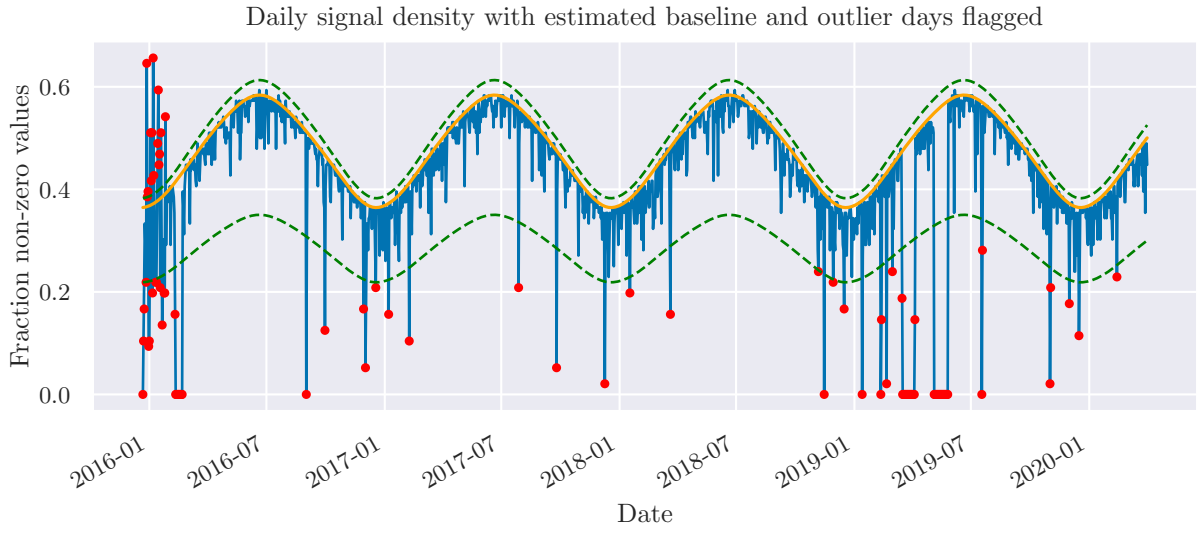

Figure 2: A visualization of the Solar Data Tools quality check procedure on the data set shown in figure 1. The orange line is the data-driven baseline, and the dashed lines show the outlier selection rules.

PVPS), Task 13.2, "Performance, Operation and Reliability of Photovoltaic Systems" [8]. The approach to analyzing degradation taken in PVInsight (the "SCSF apppoach") [7] differs from the other approaches considered in the IEA-PVPS review by not assuming access to any sort of irradiance data, which we consider to be a form of "labeled" data for PV systems. This allows our method to estimate the drift for irradiance sensors as well as degradation in PV system power.

There are some conceptual similarities between the SCSF approach to degradation analysis and more traditional PI-based analyses that substitute a well-designed clear sky model for the measured irradiance trend [9]. There is a basic underlying assumption in both approaches that the actual year-to-year variation in on-site clear sky irradiance is stable. Recent work has shown that this assumption is reasonable for more than 3 years of production data [10]. Again, the SCSF degradation estimator differs from these other published approaches in that no site information or model is required, and the "clear sky component" is inferred from the data itself, rather than being generated by an external clear sky model [11].

Soiling analysis. Recent work at estimating soiling from PV system production data are represented by [6, 12]. In these papers, periods of soiling and cleaning events are inferred from the data using various techniques, and then the periods between cleaning events are analyzed to extract soiling rates. These papers always operate on a performance index, which requires labeled PV data to generate. Additionally, the work by other researchers in this space can be best described as "collections of heuristics".

By applying the OSD framework to the soiling estimation problem, we achieve two improvements over the existing methods. The PVInisght formulation of soiling estimation (presented as a work-in-progress at 2021 PVSC [13] with full manuscript in process) is applicable to *both* labeled and unlabeled production data. Additionally, we formulate the entire

estimation process (from a time-series representation or either power or PI to segmented periods of soiling and recovery) as a single, formal mathematical optimization problem, providing a much stronger mathematical foundation than other methods. The PVInsight approach to soiling estimation will be described in more detail in §4.4.2.

Location and orientation estimation. Previous research on estimating the location, tilt, and azimuth of fixed-tilt PV systems from their power data can be found in [14, 15]. In PVInsight, we took a fresh look at the problem, greatly simplifying the estimation steps and governing equations and validating the new approach on a much larger data set. [16]. Furthermore, we implemented the algorithms in the open-source software package, PV System Profiler. Previously published papers did not provide code implementations of their proposed methods, leaving the implementation up to others.

3 Project Objectives

Our goal is to make sure that distributed, smaller-scale PV systems do not become stranded assets. There are a rapidly increasing number of rooftop PV systems that have internet connection and are generating time series data sets of power generation, sometimes with related measurements like DC current and voltage. However, it is difficult for industry to make use of this data as the well-known methods for analyzing PV system performance typically rely on calculating a performance index [4], which requires access to what we refer to as *labeled* PV data. In a PV performance context, labeled data are measurements of power combined with the information necessary to model the system (sometimes called "system metadata") and correlated environmental measurements, typically some aspect of irradiance and temperature.

Obtaining labeled data for smaller scale, distributed PV systems is often difficult or impossible. For this reason, we have sought to develop tools to manage and analyze *unlabeled* PV data, primarily by modeling and analyzing the time-dependent statistics of the signals, rather than by applying a performance index approach[1]. We model these time-dependent statistics to define signal decomposition problems, which we solve through an optimization-based framework (OSD) that has been developed under this project and will be discussed in §4.1.

In addition to developing a statistical framework suitable to analyzing unlabeled PV data, we have applied to framework to numerous PV performance analysis tasks, as will be detailed in §4. We have also taken these algorithms and implemented them in open-source software, making them available to researchers in both academia and industry. Our primary piece of software, Solar Data Tools, has received hundreds of downloads and is being tested and applied in a number of industry settings.

We sought to develop algorithms and software that enable the quick analysis of unlabeled PV power data. The tools are intended to reduce the *time to analysis* for new data sets and to enable the automation of analysis of fleets of heterogenous PV systems. We hope that the software created through this project will assist industry, academia, and system owners

[1]It is worth noting that there are, in fact, time-dependent statistics in performance index signals that may be modeled, it is simply the case that in the majority of published literature and well-known practices, no time-dependencies are utilized. A notable exception is the year-on-year methodology [17] underpinning RdTools, which models the yearly periodicity in PI signals that often occurs due to imperfect normalization.

to make use of and gain insight from PV production data.

Finally, we conclude by noting that while the approaches in PVInsight are motivated by the needs of distributed PV with respect to unlabeled data, the approaches and tools developed under this project are also applicable to the large-scale utility PV systems. One area of note is in the area of soiling analysis, in which special sensors are needed to carry out a standard analysis (typically a reference cell, module, or string that is regularly cleaned). These sensors are often not present at utility PV plants, and so they required a solution for unlabeled data as well. Another application is the use of SCSF-degradation analysis to evaluate irradiance sensor drift.

3.1 Task list.

Task 1 Form a technical advisory committee (TAC) and collect feedback on technical scope

Task 2 Data set collection, on-boarding, and baselining

- **Task 2.1** Obtain fleet-scale data set(s) from industry partners to support algorithm development and software testing
- **Task 2.2** Research dimensionality reduction techniques for PV signals and develop baselining algorithm
- **Task 2.3** Publish methods for on-boarding, cleaning, pre-processing, and baselining data

Task 3 Develop algorithms and software to estimate PV system location and orientation

- **Task 3.1** Obtain ground truth for system parameter estimation for validation of algorithms
- **Task 3.2** Development and validation of algorithm(s) to estimate the latitude, longitude, tilt, and azimuth of fixed-tilt PV systems

Task 4 System loss factor disaggregation using signal processing based techniques

- **Task 4.1** Develop formulation for optimization-based signal decomposition (OSD) framework
- **Task 4.2** Develop methods for estimating shade losses, soiling lossing, inverter clipping, long-term degradation, inverter faults, and data acquisition sytsem errors in fixed-tilt and tracking PV systems.
- **Task 4.3** Validation of scalability and accuracy of loss factor disaggregation

Task 5 Disseminate results through publishing software, publishing papers, attending conferences/tradeshows, and communicating with the TAC

Task 6 Obtain validation data set of >500 unique PV systems and fully validate PVInsight algorithms

Task 7 Release version 1.0 of Solar Data Tools and integrate algorithms into analylsis system with convenient object-oriented API suitable for applications such as Jupyter notebooks and large-scale scripting

3.2 Milestone list

Milestone 1 (1/1/19) Write white paper describing the identified research gaps, adjusted project scope and specific feedback from the participants.

Milestone 2 (4/1/2019) Present initial results of solar clear sky baselining methodology to TAC as draft journal paper.
Milestone 3 (7/1/2019) Draft paper and present initial results on the fixed parameter estimation methodology.
Milestone 4 (10/1/2019, *yearly*) Targeting to achieve the following errors in Task 3: azimuth of less than 10, errors in tilt of less than 5, errors in latitude of less than 5, and errors in longitude of less than 3 for 85% of systems and demonstrate the completed system fingerprinting and ideal signal generation for a minimum of 150 systems.
Milestone 5 (1/1/2020) Present initial results on loss factor disaggregation in a white paper to be reviewed by stakeholders and the TAC, with approaches for estimating shading, soiling, inverter clipping, long-term degradation, inverter faults, and data acquisition system errors.
Milestone 6 (4/1/2020) Validate and evaluate limits of performance disaggregation algorithms, targeting less than one hour of processing time to apply algorithms to a system with at least 2 years of data at 5-minute intervals
Milestone 7 (7/1/2020) Publish documented PVInsight code to a publicly available online repository.
Milestone 8 (10/1/2020, *yearly*) Submit papers on loss factor disaggregation and capacity factor estimation for peer review in academic journals or conference proceedings.
Milestone 9 (1/1/2021) Have plan in place for validating all algorithms, including validation procedures and identifying necessary data sets
Milestone 10 (4/1/2021) Complete validation on data set of >500 unique PV systems. Develop methods for obtaining ground truth on all PVInsight algorithms and perform statistical validation as described in Task6.
Milestone 11 (7/1/2021) Finalize software to integrate PVInsight algorithms into streamlined, automated process
Milestone 12 (10/1/2021, *yearly*) Finalize documentation and industry feedback

4 Project Results and Discussion

In this section, we present the major technical results of the project. We found that the task strcture we planned for ourselves in 2018 was an effective tool for guiding the progress of the research over the past three years. However, it is not the best organizational structure to communicate the achievements and products of the project. So, we briefly describe how the manuscript will proceed and how the topics relate to the tasks and milestones defined in §3.

In §4.1, we will discuss the optimalization-based signal decomposition (OSD) framework, which is fundamental to much of the rest of the project and officially fulfills subtask 4.1, but actually ended up supporting work throughout tasks 2, 3, and 4. We will spend a bit more time on this topic then may otherwise seem typical for a single subtask, as the methods and results became so fundamental to all aspects of the technical work.

In §4.2, we will describe data onboarding and processing methods encompassed in Solar Data Tools, which officially supports subtask 2.3. This software itself has be come a front-end entry point to all analysis we do on PV data in the project. Then in §4.3, we will describe the research on statistical clear sky baselining and the associated software package (which is now integrated as a component of Solar Data Tools), supporting subtask 2.2. In §4.4 we

describe the procedure for estimating systems losses, which relates to subtask 4.2. Finally, §4.5 explains the methodology for estimating latitude, longitude, tilt, and azimuth, relating to task 3. Discussion of the software development, conferences, and papers that supported tasks 1, 5, and 7 will be in §5.

4.1 Optimization-based signal decomposition (OSD)

In support of analyzing unlabeled time-series data, and in collaboration with Prof. Stephen Boyd from the Electrical Engineering department at Stanford University, we developed a framework for decomposing vector-valued time-series or signals, possibly with missing entries, into hidden components. A monograph of this topic, written by Bennet Meyers and Stephen Boyd in nearing publication. A public lecture hosted by SLAC is available online, which is a preview of the coming manuscript and explains the major aspects of the research [18].

We briefly note that we did not set out with the intention of developing a new framework for signal decomposition at the beginning of the project. We expected to "develop a signal processing based approach" for "decomposing PV loss factors." We mentioned in the proposal and the statement of projects objectives that we expected to apply a method known a *contextually supervised source separation* (CSSS) [19] to this problem. As the project developed and through conversations with Prof. Boyd, it became clear that OSD, which we will define shortly, would be better fit for analysis problems we wished to tackle in PVInsight. Briefly, OSD addressed the needs of the project as follows:

- OSD is a more general, extensible framework than CSSS, allowing for the expression of a larger class of decomposition problems.
- OSD is designed from the ground up to handle missing values in a mathematically rigorous way.
- The OSD problem structure yields a scalable algorithm for solving the estimation problems (CSSS proposed no such specialized algorithm and instead used a commercial interior-point solver [19]).

The forthcoming monograph will go into great detail on the theory and application of OSD. For this report, we summarize the basic concepts of OSD.

4.1.1 Definition of vector signal decomposition over missing data

Consider a vector time series or signal, possibly with missing entries, $y_1, \ldots, y_T \in (\mathbf{R} \cup \{?\})^p$. We use the value ? to denote a missing entry in the signal, and say that entry $y_{t,i}$ is known if $y_{t,i} \in \mathbf{R}$, and unknown if $y_{t,i} =?$. We define $\mathcal{K}$ as the set of indices corresponding to known values, *i.e.*, $\mathcal{K} = \{(t,i) \mid y_{t,i} \in \mathbf{R}\}$. We define $\mathcal{U}$ as the set of indices corresponding to unknown or missing values, *i.e.*, $\mathcal{U} = \{(t,i) \mid y_{t,i} =?\}$. We represent the signal compactly as a $T \times p$ matrix $y \in (\mathbf{R} \cup \{?\})^{T \times p}$, with rows $y_1^T, \ldots, y_T^T$.

We will model the given signal y as a sum (or decomposition) of K components $x^1, \ldots, x^K \in \mathbf{R}^{T \times p}$,

$$y_{t,i} = (x^1)_{t,i} + \cdots + (x^K)_{t,i}, \quad (t,i) \in \mathcal{K}.$$

We refer to this constraint, that the sum of the components matches the given signal at its known values, as the consistency constraint. We introduce the notation $\overset{\mathcal{K}}{=}$ to mean that the

left and righthand sides agree on the known entries, so the consistency constraint above can be written as

$$y \overset{\mathcal{K}}{=} x^1 + \cdots + x^K. \tag{1}$$

Note that the components $x^1, \ldots, x^K$ do not have missing values. Indeed, we can interpret the values

$$\widehat{y}_{t,i} = x^1_{t,i} + \cdots + x^K_{t,i}, \quad (t,i) \in \mathcal{U}, \tag{2}$$

as estimates of the missing values in the original signal y. (This will be the basis of a validation method described in the monograph.)

4.1.2 Component classes

The K components are characterized by functions $\phi_k : \mathbf{R}^{T\times p} \to \mathbf{R} \cup \{\infty\}$, $k = 1, \ldots, K$. We interpret $\phi_k(x)$ as the loss of or implausibility that $x^k = x$. In some cases we can interpret the classes statistically, with $\phi_k(x)$ the negative log-likelihood of x for signal class k, but this does not have to be the case. Roughly speaking, the smaller $\phi_k(x)$ is, the more plausible it is. Infinite values of $\phi_k(x)$ are used to encode constraints on components. We refer to x as feasible for component class k if $\phi_k(x) < \infty$, and we refer to $\{x \mid \phi_k(x) < \infty\}$ as the set of feasible signals for component class k. When a component class takes on the value ∞ for some x, we say that it contains or encodes constraints; when ϕ_k does not take on the value ∞, we say the component class has no constraints, or has full domain. We will assume that every component class has at least one feasible point, *i.e.*, a point with finite loss.

The monograph contains many examples of component class losses, but for now we mention a few simple examples.

Mean-square small class. One simple component class has the mean-square loss

$$\phi(x) = \frac{1}{Tp}\sum_{t,i}(x_{t,i})^2 = \frac{1}{Tp}\|x\|_F^2, \tag{3}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, the squareroot of the sum of squares of the entries. (To lighten the notation, we drop the subscript k when describing a general component class.) All signals are feasible for this class; roughly speaking, smaller signals are more plausible than larger signals. We call this the component class of mean-square small signals.

We will assume that the first class is always mean-square small, with loss function (3). We interpret x^1 as a residual in the approximation

$$y \approx x^2 + \cdots + x^K,$$

and $\phi_1(x^1)$ as the mean-square error.

Mean-square smooth class. The component class of mean-square smooth signals has loss

$$\phi(x) = \frac{1}{(T-1)p}\sum_{t=1}^{T-1}\|x_{t+1} - x_t\|_2^2, \tag{4}$$

the mean-square value of the first difference. Here too all signals are feasible, but smooth ones, *i.e.*, ones with small mean-square first difference, are more plausible.

Boolean signal class. As one more simple example, consider the component class with loss function

$$\phi(x) = \begin{cases} 0 & x_{t,i} \in \{0,1\} \text{ for all } t,i \\ \infty & \text{otherwise.} \end{cases} \tag{5}$$

This component class consists only of constraints, specifically that each entry is either 0 or 1. It has a finite number, 2^{Tp}, of feasible signals, with no difference in plausibility among them. We refer to this class as the Boolean component class.

4.1.3 Signal decomposition problem

We will estimate the components $x^1, \ldots, x^K$ by solving the optimization problem

$$\begin{array}{ll} \text{minimize} & \phi_1(x^1) + \cdots + \phi_K(x^K) \\ \text{subject to} & y \overset{K}{=} x^1 + \cdots + x^K, \end{array} \tag{6}$$

with variables $x^1, \ldots, x^K$. We refer to this problem as the *signal decomposition (SD) problem*. Roughly speaking, we decompose the given signal y into components so as to minimize the total implausibility. The SD problem is always feasible, but it need not have a unique solution. We refer to a solution of the SD problem as an optimal signal decomposition.

Solving the signal decomposition problem. If the class losses ϕ^k are all convex functions, the SD problem (6) is convex, and can be efficiently solved globally [20]. In other cases it can be very hard to find a globally optimal solution, and we settle for an approximate solution. In the OSD monograph, we will describe a method that solves the SD problem when it is convex (and has a solution), and approximately solves it when it is not. Our method is based on ADMM [21], an operator splitting method that handles each of the component classes seperately. This allows us to parallelize the method, gives a very convenient software architecture, and makes it easy to modify or extend it to many component classes.

4.1.4 OSD in PVInsight

We utilize the OSD framework extensively throughout PVInisght analytics. One such example has already been shown in figure 2, which shows the use of OSD to flag days with operational issues with Solar Data Tools. Other uses of OSD will be explained in more detail in the rest of this report.

Currently, the software implementations of OSD in PVInsight are not using the algorithm to be proposed in the coming monograph. Instead, the applications in PVInsight have been prototyped using CVXPY [22, 23] and Mosek [24]. These implementations may be found in `solardatatools.signal_decompositions`. As described in §7, a major aspect of PVInsight phase 2 will be the implementation of the proposed algorithm throughout PVInsight software, removing the software dependency on Mosek.

4.2 Data onboarding

The ability to quickly onboard, clean, and standardize unlabeled PV data sets is not just an opperational problem in the industry, but it is also a practical problem for carrying out the research and analysis proposed in PVInsight. Solar Data Tools [25, 26] is an automatic data processing pipeline application, written in open source Python. This software takes as an input generic, tabular PV performance time series data, typically a power signal at some

```
1 import pandas as pd
2 from solardatatools import DataHandler
3 df = pd.read_csv("PVDataExample.csv")
4 dh = DataHandler(df)
5 dh.run_pipeline(power_col="ac_power_01")
```

Figure 3: A basic instantiation of the `DataHandler` class from Solar Data Tools.

Table 1: Summary of algorithms in the Solar Data Tools processing pipeline

Tool	Standardize	Clean	Label	OSD
Time stamp cleanup	✓			
Sampling rate detection	✓			
Matrix embedding	✓			
Missing data		✓		
Time shift		✓		✓
Data quality			✓	✓
Clear/cloudy			✓	✓
Inverter clipping			✓	✓
Capacity change			✓	✓

sampling frequency, possibly with some missing or corrupted data. No site model, system metadata, or reference irradiance or temperature data is required, so the software can work with distributed rooftop data as well as centralized power plant data. The software can process correlated data columns along with power data, when available. A typical usage is shown in figure 3. The `DataHandler` class is instatiated on a Pandas DataFrame [28, 29] containing PV data in tabular form. The example shown in figure 3 assumes "wide-form" data, with each row corresponding to a unique time stamp and each column representing a different measurement (power signals, voltage signals, temperature signas, etc). Solar Data Tools can also accept "long-form" data, with each row representing a single measurement from one sensor so that time stamps are repeated in subsequent rows to record values from different sensors[2]. The software will convert long-form data to wide-form before proceeding.

When the `run_pipeline` method in line 5 of figure 3 is run, the software initiates a serial data processing pipeline that *standardizes*, *cleans*, and *labels* the input data. This process is summarized in table 1 and typically takes on the order of 10-30 seconds for typical data sets on a standard laptop. The column labeled "OSD" inidicates subroutines in the pipeline that make use of optimization-based signal decomposition (see §4.1). After running the pipeline, the `DataHandler` object may be used as an entry point to further investigation and analysis, including data set summarization, visualization/plotting, and loss factor analysis.

In the following sections, we document each of the items described in table 1. Two of these subroutines—data quality labeling and inverter clipping labeling—are loss factors that

[2]Long-form data is less common in the solar industry, but it is often the form returned by queries to key-value databases such as Apache Cassandra or Google Bigtable.

are identified in task 4.2.

4.2.1 Standardize

Time stamp cleanup. As described in §4.1.1, we define time-series signals mathematically as collections of vectors that are indexed by time. So, our data must occur on regular intervals, such as one value every minute or every hour, for example. This is generally a safe assumption for PV performance data sets, which are typically generated by data aqcuisition systems (DAS) installed at the point of power generation. These DAS are usually programmed to record the average measured power over a set interval, with the most common choices for PV data being 1-minute, 5-minute, 15-minute, and 60-minute intervals. Unfortunately, this idealized description is rarely experiences in practice. In reality, data sets have various time-index inconsistencies. Time stamps may be missing or repeated. Sometimes, records are recorded at odd intervals, such as every 5 minutes plus or minus 60 seconds or so.

So, the first thing the pipeline does is rebuild the time axis of the input data frame. This code can be found in the definition of the `standardize_time_axis` function in the `solardatatools.time_axis_manipulation` module [25]. The software infers the scan rate or rates from the data by looking at the time between consecutive stamps, picking the most common value if there are multiple rates. Then, a new time axis is generated starting at midnight on the first day and ending at one timestamp before midnight on the last day, using the standard frequency. We then utlitize the Pandas reindexing function using the keyword arguments `method="nearest"` and `limit=1`. This forces the data into the regular time index, using the closest available value for each time stamp and not filling in any gaps, which are left as `NaN` values and are treated as ? in OSD.

Matrix embedding. We find that a useful and compact way to represent a multi-year PV power data set is by embedding it in a matrix $P \in (\mathbf{R} \cup \{?\})^{m \times n}$ where m is the number of measurements on each day (24-hour period starting and ending at midnight) and n is the number of days in a data set. For example, the heatmap shown in figure 1 is just an image representation of a a 96×1562 matrix. The data has a 15-minute scan rate, so there are 96 measurements per day, and there are 1562 days in the data set. When other data columns are to be analyzed (such as correlated irradiance and temperature), the software generates matrices of equivalent shape for the other measurements. Having first standardized the time axis of the data, reshaping the scalar signal into the appropriate shape is straight forward because the new time axis is designed to have exactly mn entries. This is easily achieved with the `numpy.reshape` function [30]. This code may be found in `solardatatools.matrix_embedding.make_2d`. This matrix embedding of the power data is our "standardized form" of the data. Data cleaning and labeling operations in Solar Data Tools assume this matrix form, as do subsequent analyses like Statistical Clear Sky.

4.2.2 Clean.

Data filling Some data operations expect the measured signal to be real-valued everywhere, with no missing values or `NaN` values (while OSD is designed to directly handle missing values, see §4.1). So, we generate a copy of the data with the missing values filled in, $P_{filled} \in \mathbf{R}^{m \times n}$. We infill the missing values by the following rules:

- night time values are filled with zeros

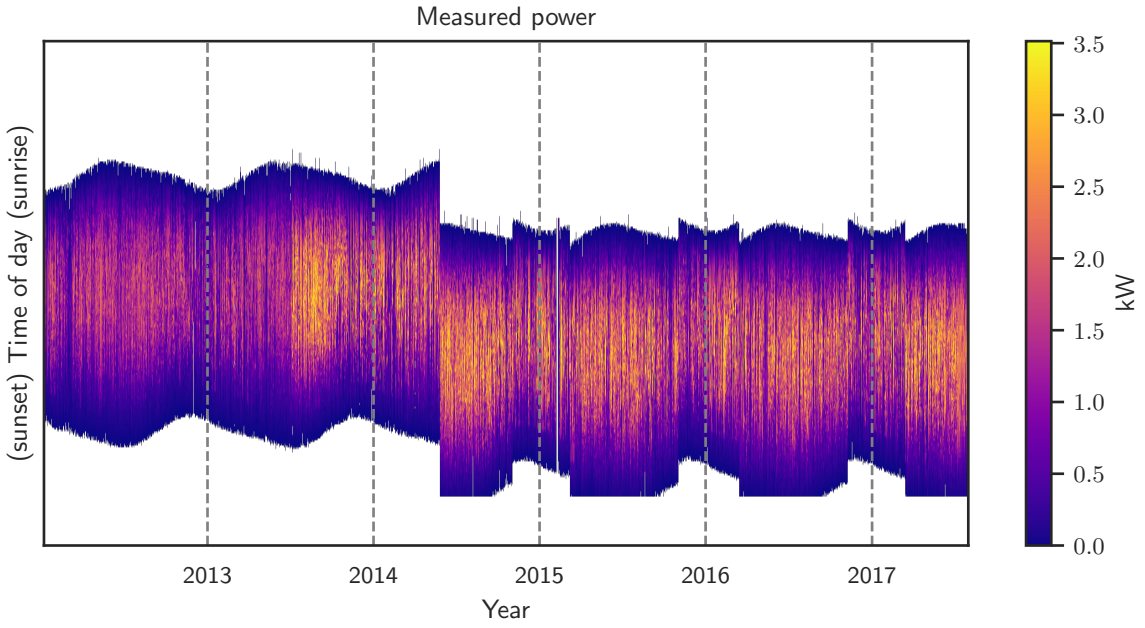

Figure 4: The raw matrix embedding of a PV power data set with multiple time shift errors. There is a "one off" error in mid-2014 followed by daylight savings shifts twice a year for each of the following years. This is generated with the `solardatatools.DataHandler.plot_heatmap` function.

- daytime values are filled by linear interpolation
- days that are completely missing are filled with zeros.

This is implemented in solardatatools.data_filling. After the pipeline is complete, the `DataHandler` class instance maintains two attributes, `raw_data_matrix` and `filled_data_matrix` corresponding to the original matrix embedding and the infilled version respectively. An example of the raw data matrix and the filled data matrix is shown in figures 4 and 5 respectively.

Time shift detection and correction. *Time shifts* are a common error in PV data sets that occur when the local DAS clock time is changed for some reason. This often occurs due to local daylight savings time, where the local clock time is changed by an hour twice a year, but it can occur for other reasons as well. The Solar Data Tools pipeline includes an algorithm that automatically detects when these shifts occur and "corrects" them by adjusting the time stamps so that the shifts are removed.

The matrix-embedding of the power signals makes these issues quite clear. An example of a data set with both types of time shift errors is shown in figure 4. In figure 5, we show how the algorithm corrects the time shift errors for this example. (Note that this data set also has a capacity change in 2013, which will be discussed in more detail in §4.2.3.) There is a large one-time shift in the first half of 2014. After that event, the data set starts to exhibit yearly daylight savings shifts as well, seen as a shift up (earlier) in the fall and a shift down (later) in the spring. The algorithm correctly identifies all 7 time shift events in this

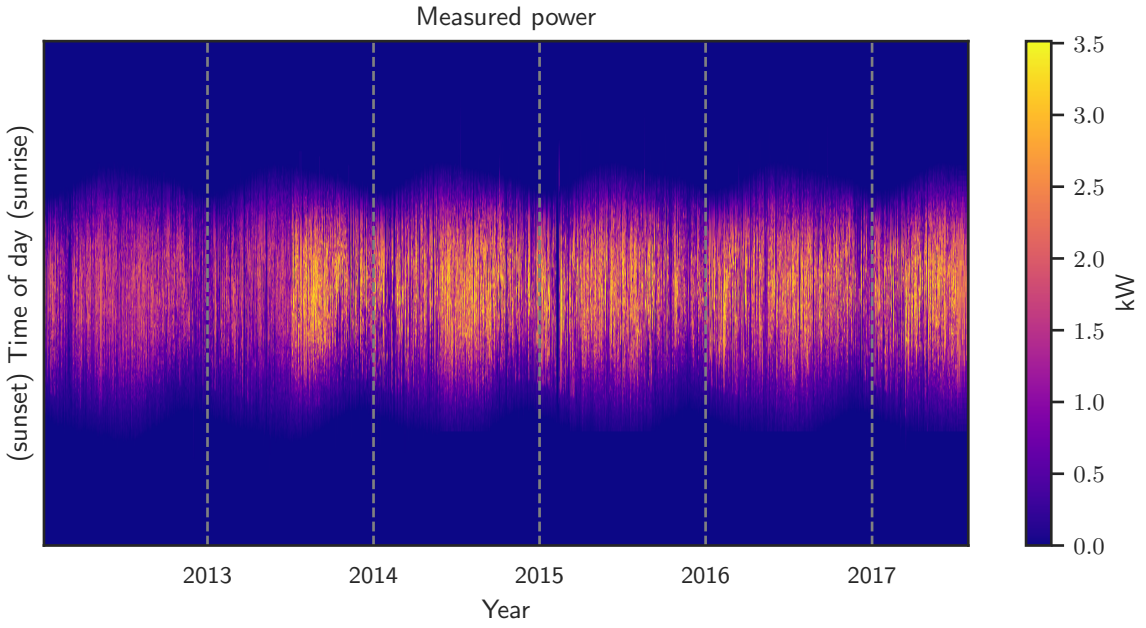

Figure 5: The filled and time-shift-corrected matrix embedding of the PV power data set shown in figure 4 after the application of the Solar Data Tools data cleaning subroutines. The time shifts have been eliminated. This is generated with the `solardatatools.DataHandler.plot_heatmap` function.

data set and removes them.

The algorithm is an application of OSD, and the implementation can be found in `solardatatools.algorithms.time_shifts`. First, the measured power signal is converted into a daily estimation of solar noon time. This can be done by inferring sunrise and sunset times in the data using a simple threshold and taking the average ("sunrise-sunset method" or "srss") or by calculating the energy-weighted mid-point of the day ("energy center-of-mass" or "com"). The user may select either method, and the default is "srss". This daily solar noon signal is then used as an input to OSD. We dicompose the signal into residual, seasonal baseline, and piecewise constant (PWC) terms, as shown in figure 6. The PWC component uses a cost function defined as

$$\phi(x) = \|Dx\|_1 \,, \tag{7}$$

where $D \in \mathbf{R}^{(m-1)\times m}$ is the first-order difference matrix. The ℓ_1-norm encourages sparsity in the first difference of the component, which results in a signal that is piecewise constant. When the PWC component changes value, a time shift is detected. The magnitude of that change is used by the algorithm to correct the time stamps.

4.2.3 Label

Data quality. The "data quality" quality label is a Boolean value assigned to each day in the data set (each column of the embedded power data matrix) that is `True` when the day has not experienced an inverter or DAS outage and is `False` if an inverter or DAS outage has

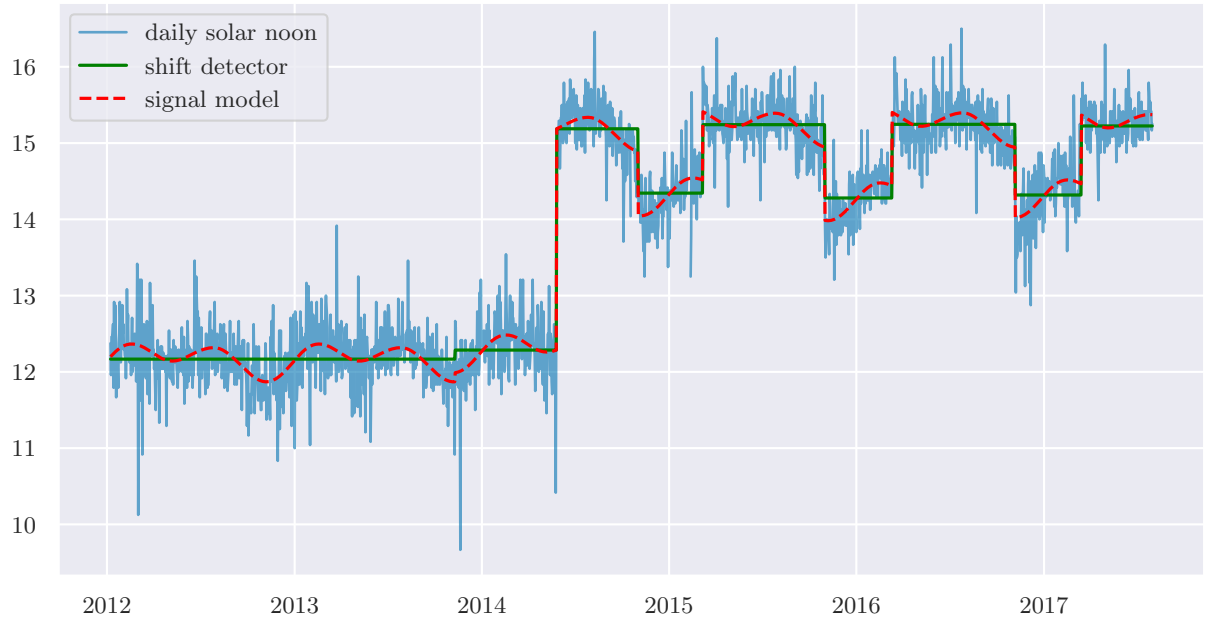

Figure 6: Illustration of OSD applied to the problem of finding time shifts in PV power data sets. This is generated with the `solardatatools.DataHandler.plot_time_shift_analysis_results` function.

occured. We define a `True` value to correspond to one or both of the following conditions:

1. the day is an outlier in the fraction on non-zero power measurements, relative to a seasonal baseline
2. the day is an outlier in the fraction of the daylight hours during which the power followed a linear trend with respect to time

The first condition catches when the inverter is off for longer than it should be during a certain season, indicative of an inverter outage. It also catches where there are too many non-zero values, which typically occurs when there has been a DAS malfunction has caused a "hold value" to be recorded. This occurs when the measurements are not updating but the DAS thinks the last measured power is still the current power. The second condition catches when there has been a linear interpolation infill, which is a common choice in many DAS and database systems. We utilize linear interpolation in our own data infill procedure, as described in §4.2.2. The process for generating the final Boolean label is represented as a process diagram in figure 7. The code implementation is in `solardatatools.data_quality`.

The first condition is checked by fitting a seasonal baseline to a daily signal representing the fraction of non-zero values in each day ("signal density," as in the inverse of sparsity), as shown in figure 2. We use an asymmetric cost function for the residuals, to model the fact that clouds tend to reduce the signal density, not not incraese it, relative to the clear sky baseline. See `solardatatools.data_quality.make_density_scores`.

The second condition is checked by taking the first-order difference of the power matrix down each column, which can be efficiently implemented in vectorized numpy. This is done

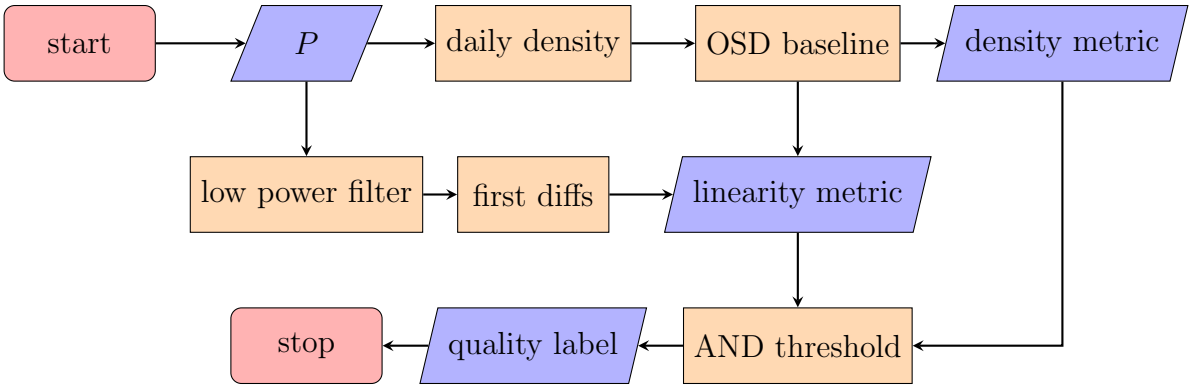

Figure 7: Process flow diagram for generating the data quality metric in Solar Data Tools. Red rounded rectangles are start/stop nodes. Blue parallelograms are input/output nodes. Yellow rectangles represent operations on the data. The input P is a power signal embedded in a matrix.

after first filtering the measured power data to remove low power values below a threshold (0.5% of the estimated system capacity). Then the first-order differences are rounded to four significant digits and the mode of the first differences is calculated for each day. Finally, a "linearity metric" is generated by calcuating what fraction of the non-zero values correspond to first order differences that are equal to the daily mode. This calculate the fraction of the day spend on a single linear trend. See `solardatatools.data_quality.make_linearity_scores`.

We finally define simple thresholding rules on these two metrics, as shown in figure 8. Days with a linearity score less than 0.1 and a density score between 0.6 and 1.05 receive a `True` label, while the remaining points receive a `False` label. The points the scatter plot are colored according to the clusters found with DBSCAN [31] as implemented in Scikit Learn [32]. We generally expect the main cluster (label 0, blue in the plot) to be within the decision boundaries. If it is not, the software will raise a warning for the user. In this example, the user may want to manually inspect the orangle cluster (label 1) as that grouping of days has an abnormally large linearity score.

Clear/cloudy. The second labeling operation performed by the pipeline is another Boolean label; it is `True` if the day is (mostly) clear, and it is `False` if the day is mostly cloudy. Only days that have `quality=True` are assigned a clearness label. The code implementation may be found at `solardatatools.clear_day_detection`.

Clear days are identified as those days that both

- have a high energy content relative to a seasonal baseline
- are smaller with respect to the smoothness measure $\|D^2 x\|_2^2$, where $D^2 \in \mathbf{R}^{(m-2)\times m}$ is the second-order finite difference matrix.

This is based on the observation that cloudy days can be smooth (if it is highly overcast) or high-energy (if the clouds are intermittent or localized in the morning or afternoon), but they generally are not both high energy and smooth. Days that are both close to the energy baseline and are small with respect to the smoothness metric are given the flag `clear=True`.

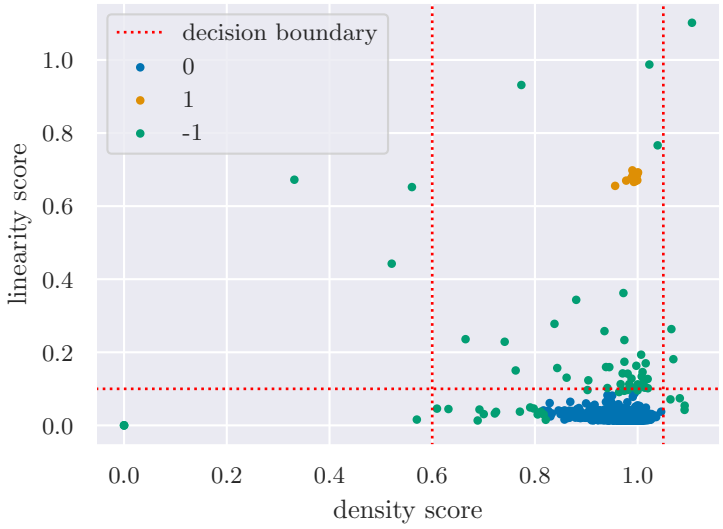

Figure 8: A scatter plot of linearity scores versus density scores for the data set shown in figure 5. This is generated with the `solardatatools.DataHandler.plot_data_quality_scatter` function.

The procedure for estimating the seasonal energy baseline is based on OSD and is very similar to the procedure for estimating the signal density baseline described in §4.2.3. The daily signal that is derived from the the mesaured power, in this case, is simply the energy. The OSD problem formulation is identical. Both subroutines use `solardatatoos.signal_decompositions.tl1_l2d2p365`.

Inverter clipping. The third labeling operation also sets a Boolean label; it is `True` if the day has experienced inverter clipping, and it is `False` if the day has not experienced inverter clipping. As in §4.2.3, only days that have `quality=True` are assigned a clipping label. The code implementation may be found at `solardatatools.algorithms.clipping`.

We identify clipping points by their tendency to generate many power data points at nearly the same value. This creates one or more "point-masses" in the distribution of power values for that system. We emphasize the "or more," as our experience has shown us that re-configuring the system size and clipping point during operation is not uncommon in distributed PV systems. An example is shown in figure 9. Here we show the cumulative density function (CDF) and histogram for the power measurements (normalized to $[0, 1]$) for a PV system that was reconfigured over it's five-and-a-half year history with 4 different clipping set points. In addition, there were also periods of time where the inverter clipping was turned off, and the system experienced no clipping.

The point masses are detected by fitting an OSD model with a piecewise linear component to the empirical CDF. This component is modeled using the cost

$$\phi(x) = \left\| D^2 x \right\|_1, \tag{8}$$

which is the ℓ_1-norm of the second-order differences of the signal. This cost selects for components that are sparse in the second derivative, or piecewise linear (PWL). The OSD

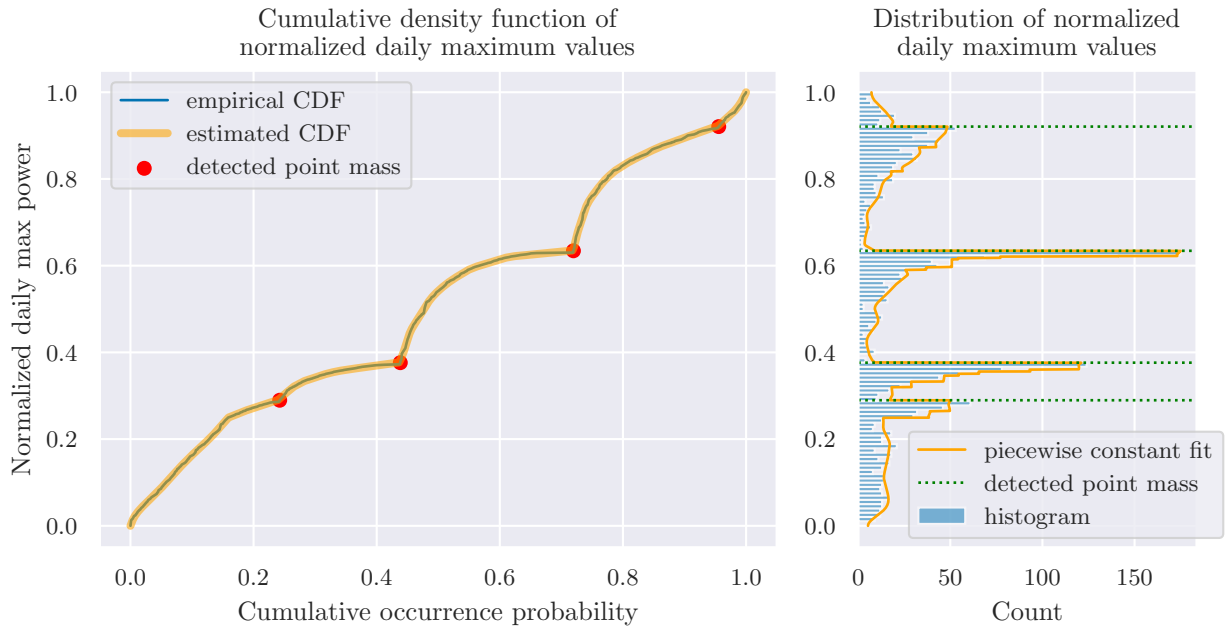

Figure 9: Four point masses in the power data from a site with four clipping set points, as seen in the cumulative density function (left) and the data histogram (right). The orange lines were estimated using OSD. This is generated with the `solardatatools.DataHandler.plot_daily_max_cdf_and_pdf` function.

formulation is implemented in `solardatatools.signal_decompositions.make_l2_l1d2`.

After fitting the OSD model, the PWL component estimate is analyzed, and the breakpoints for the linear models are identified as point-masses, if they are larger than a threshold. Finally, days that generate more than 10% of their energy at or near a power point-mass are identified as `clipping=True` days.

After the pipeline is complete, the user may wish to run the `solardatatools.DataHandler.find_clipped_times` function, which goes back and labels each individual power value as clipped or not, rather than entire days. This generates a matrix of Boolean values of the same shape as the power data matrix that is `True` if the corresponding power measurement was impacted by inverter clipping.

Capacity changes. The fourth and final label assigned by the Solar Data Tools pipeline is a categorical assignment into capacity clusters, or periods of time during which the system appeared to be at a single capacity value. The apparent capacity of a system can change for many reasons, *e.g.* from installing additional panels (an increase in capacity) or a string failure (a decrease in capacity). This process is distinct from other changes in apparent output, such as long-term degradation or soiling. In this subroutine, we are specifically looking for "step-changes" in apparent capacity, which we model in OSD as a PWC component. The code implementation may be found in `solardatatools.algorithms.capacity_changes`, and the OSD formulation is in `solardatatools.signal_decomposition.tl1_l1d1_l2d2p365`.

As with the other OSD implementations discussed so far (with the exception of clipping),

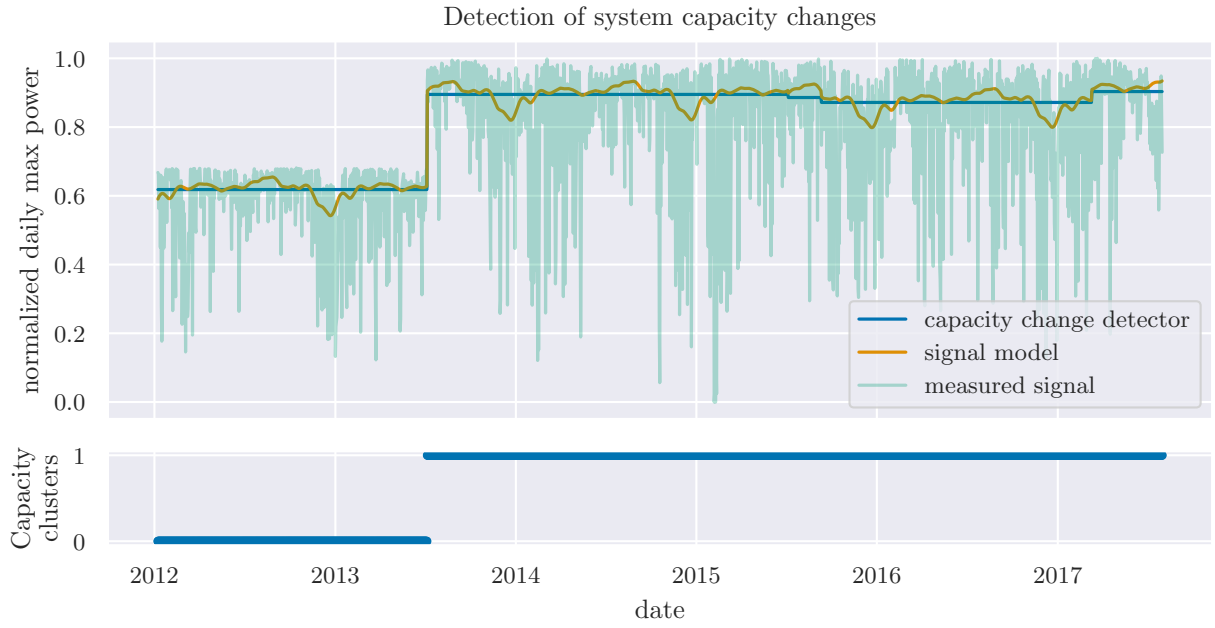

Figure 10: Capacity clustering analysis on the data presented in figures 4 and 5. Two capacity clusters were found in this data, labeled with `0` and `1`. This is generated with the `solardatatools.DataHandler.plot_capacity_change_analysis` function.

the input signal y in this subroutine is a scalar daily signal, calculated from the measured power. In this case, the daily signal the maximum power measured each day. We note that each of the daily calculations discussed so far can be expressed as a (pseduo-)norm of the columns of the power data matrix. The fraction of non-zero values (§4.2.3), the daily energy (§4.2.3), and the daily maximum values correspond to the (appropriately scaled) ℓ_0-, ℓ_1-, and ℓ_∞-norms respectively[3].

After calculating the daily maximum power, we fit an OSD model with asymmetrical residuals, a PWC component, and a smooth component that is 365-day periodic. The PWC component provides the detection of capacity changes, as shown in figure 10. After applying OSD, labels are assigned to each day identifying the day as belonging to a particular capacity cluster. For many basic loss factor analyses (*e.g.* degradation or soiling) having any capacity changes in the data is problematic. So, the purpose of this labeling operation is two-fold: (1) changes in capacity are flagged for an analyst as potential operational issues and (2) data sets *without* capacity changes may be further analyzed for more subtle operational and under-performance issues.

4.3 Statistical clear sky power baseline

Statistical clear sky fitting (SCSF) is an algorithm that estimates a clear sky performance signal from the measured power of a PV system [33, 7]. The algorithm uses only observed

[3]The ℓ_p-norm of a vector $x \in \mathbf{R}^m$ is defined, for $p \geq 1$, by $\|x\|_p = \left(\sum_{i=1}^m |x_i|^p\right)^{1/p}$, with the appropriate limit taken for $p = \infty$. Following tradition, we define the ℓ_0-pseudo-norm as the number of nonzero elements of the vector x. This measure of sparsity is not a true norm.

power output, and assumes no knowledge of weather, irradiance data, or system configuration metadata. This is a novel approach to understanding the clear sky behavior of an installed PV system, that does not rely on traditional atmospheric and geometric modeling techniques. The algorithm is contained in a stand-alone software package called Statistical Clear Sky [34]. The algorithm may be invoked from Solar Data Tools by running the `.fit_statistical_clear_sky_model` method on a `DataHandler` class instance after the pipeline has completed.

The signal model used in SCSF is not an OSD formulation. As detailed in [33], the data model is derived from a *generalized low rank modeling* framework [35]. An example of a statistical clear sky model and the input data are shown in figure 11. We draw attention to the fact that this model of system clear sky behavior includes the effects of nearby object shading, seen clearly in this example during the winter season. Note how the estimated clear sky baseline recreates the features in the data induced by the shade. This example also illustrates how SCSF may be used as a data imputation method. The dark bands seen in the top plot (measured power) are caused by periods of missing data (see §4.2.2). We see in the bottom plot that the SCSF model estimates what the clear sky power should have been during these times.

Applications. There are many applications of SCSF. The most immediate, for the purposes of the PVInsight project is the estimation of long-term degradation (see §4.4.1). SCSF is also useful for establishing a baseline for statistical power forecasting [36]. The data imputation nature of the method may also be exploited to make forecasts of future expected clear sky system response, which may be used to build systems for detecting outages and operational issues. We are continuing to look at ways to improve and apply this methodology.

4.4 Loss factor estimation

In this section, we describe the statistical signal processing approach we have taken to identifying and estimating the loss factors listed in task 4.2. Each loss factor estimation algorithm is a stand-alone analysis. For example, the long-term degradation may be analyzed for a system without analyzing soiling, and vice versa. All methods except long-term degradation involve OSD in the estimation process. The last two methods have already been discussed in §4.2 and are included in this section for the sake of completeness.

4.4.1 Long-term degradation

The long-term degradation rate of a PV system is generally defined as a single value, given as a percentage change in energy output of the system each year, typically on the order of 0.5–1.5% per year [9]. As discussed in §2.2, the approach taken in PVInsight differs from other published work in the focus on estimating the quantity when it is not possible to construct a performance index from the power data.

We estimate the year-over-year percent change in energy output of the system by including that quantity as a parameter in the SCSF model [7]. By using a holdout methodology, we are able to generate empirical uncertainty bounds on the degradation estimate that have been shown to be in good agreement with established methods [8]. For example, the degradation rate for the data set shown in figure 11 is $-0.87 \pm 0.15\%$. The uncertainty analysis is controlled by the `bootstrap=N` keyword arguement in `statistical_clear_sky.SCSF.execute`.

Recent, unpublished work has found that correcting for soiling trends in the data before

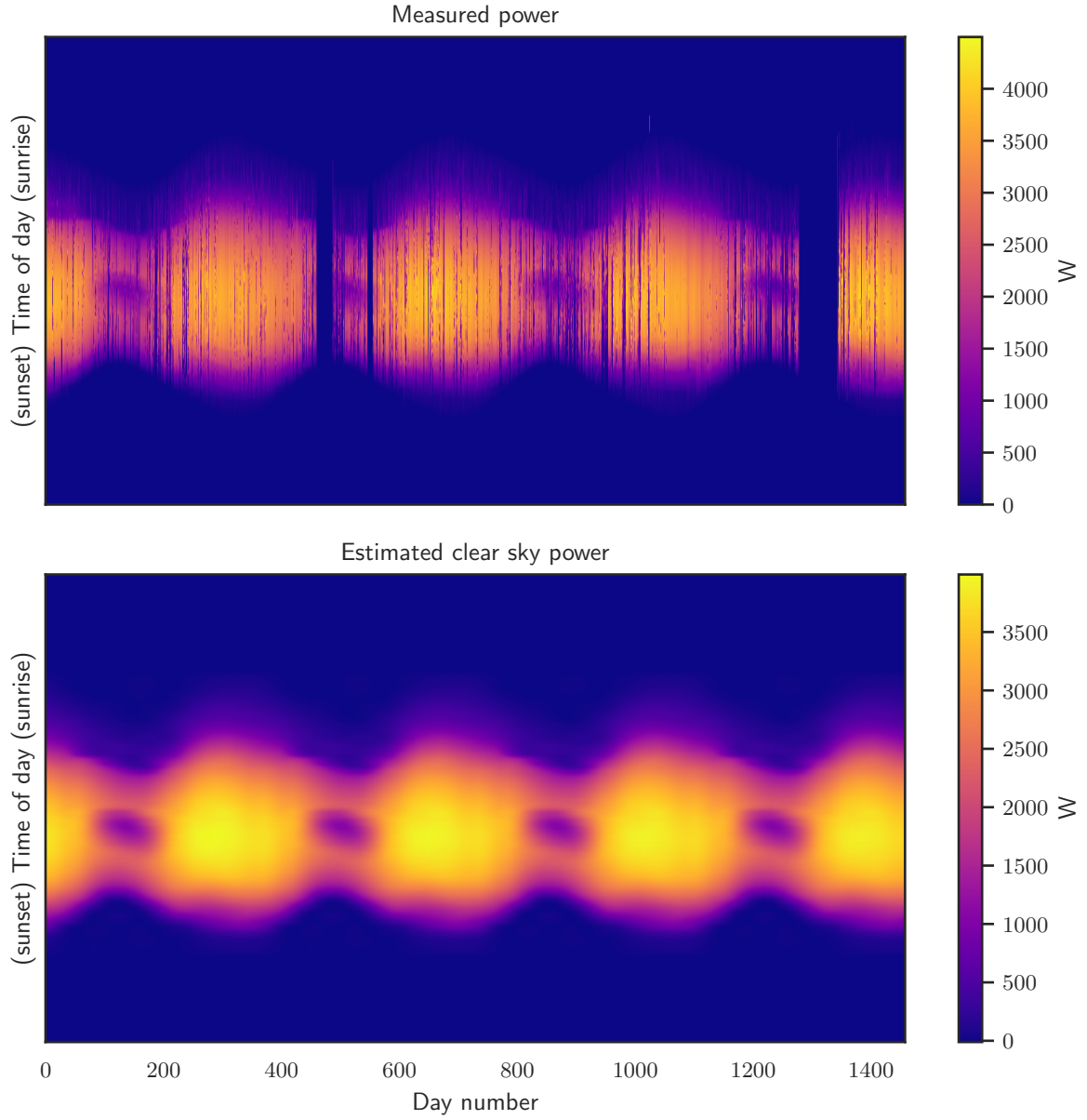

Figure 11: An example of a statistical clear sky model. This is generated with the `statistical_clear_sky.SCSF.plot_measured_clear_matrices` function.

running the SCSF degradation analysis improves the results by lowering the empirical uncertainty. The implementation of the soiling detection algorithm described in 4.4.2 includes helper functions for correcting data by removing the effects of soiling.

4.4.2 Soiling

We have designed an algorithm based on OSD that identifying soiling in daily performance data. The algorithm generates as estimate of the reduction in energy output by the system over time due to soiling, expressed as a fraction between 0 and 1. Notably, the algorithm works on both unlabeled power data as well as on performance index data, which is more commonly analyzed for soiling trends (see §2.2). We use an OSD model with the following components:

- asymmetric residuals
- piecewise constant, non-positive, with an absolute value penalty
- smooth, 365-day periodic
- linear

A full paper on this topic is still in progress and will completed after the OSD monograph, which it will require as a reference. Preliminary results were presented at the 2021 Photovoltaic Specialists Conference [13]. In that manuscript, we present data from a soiling test site in Qatar, which has two identical PV systems, one of which was cleaned regularly and the other of which was allowed to soil. We present in [13] that the OSD analysis of the PI generated by normalizing the soiled system by the cleaned system matches the results calculated by a human analyst (in a fraction of the time). This estimation of the soiling trend from the soiling-PI data is shown in figure 12. This was not presented at the conference, but we have also applied the same OSD formulation to the unlabeled daily energy signal from the soiled test system, shown in figure 13. We find that the estimate of the soiling trend from the energy data closely matches the trend estimated from the soiling-PI, as shown in figure 14. The code implementation of the soiling algorithm may be found in `solardatatools.algorithms.soiling`.

4.4.3 Shade

We have developed an algorithm for identifying shade in unlabeled PV power signals, and estimating the amount of energy lost to shade relative to an unshaded baseline. The algorithm may be found in `solardatatools.algorithms.shade`, but the approach has not been published in a paper or presented at a conference, yet. The core of the algorithm is again OSD, but unlike in all the other examples mentioned so far, this is the first example to have a vector-valued signal (all other examples have been on scalar signals). The formulation of the OSD model is significantly more complex than other examples discussed in this report, and we will decline to explain the details in this report. Instead, we will focus on presenting compelling results that illustrate the capabilities of the algorithm.

In figure 15, we annotate a heatmap of power production using the shade algorithm to identify periods of signicant shade losses. Qualitatively, we find that the annotation matches the patterns of power loss visible to the eye in the heatmap, and we have found that this holds true for many examples we have evaluated so far. In figure 16, we present an analysis

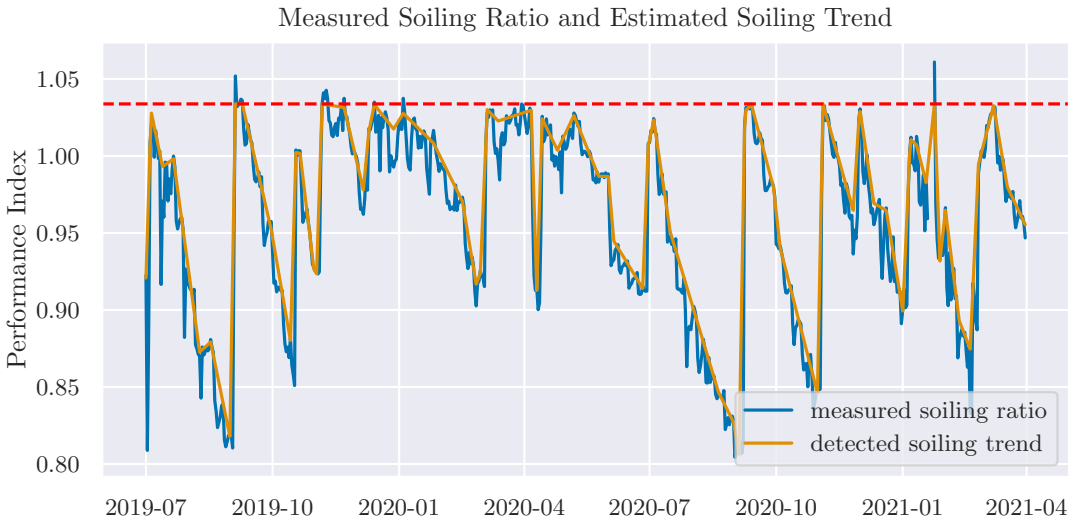

Figure 12: An extraction of a piecewise linear soiling trend from a soiling performance index, generated from the labeled data available from the Qatar test site. The periods of soiling and recovery, as well as the estimated soiling rates, closely match the results calculated by a human analyst.

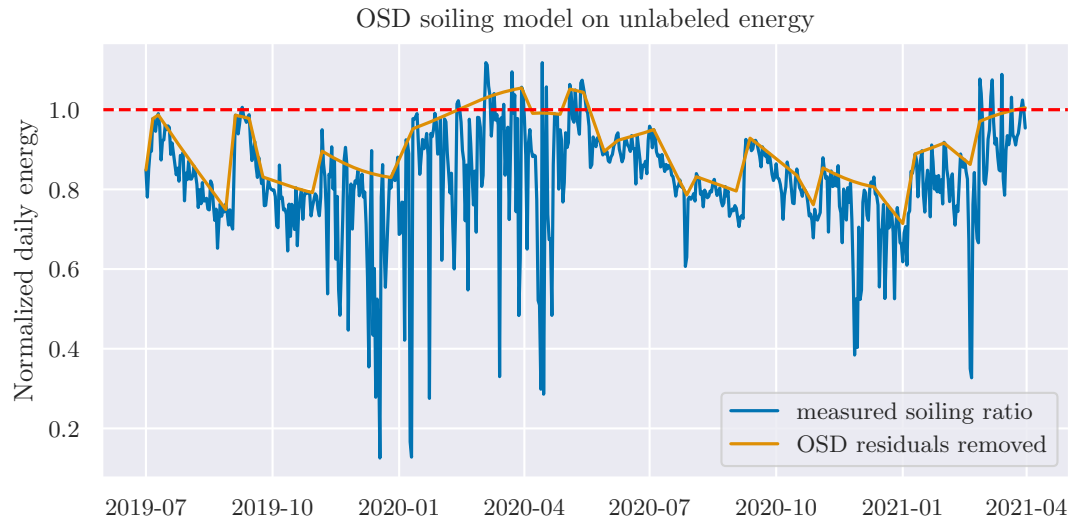

Figure 13: The dailly energy recorded from the soiled system at the Qatar test site. The fit OSD model with the residuals removed is shown in orange (soiling component, seasonal component, and linear component).

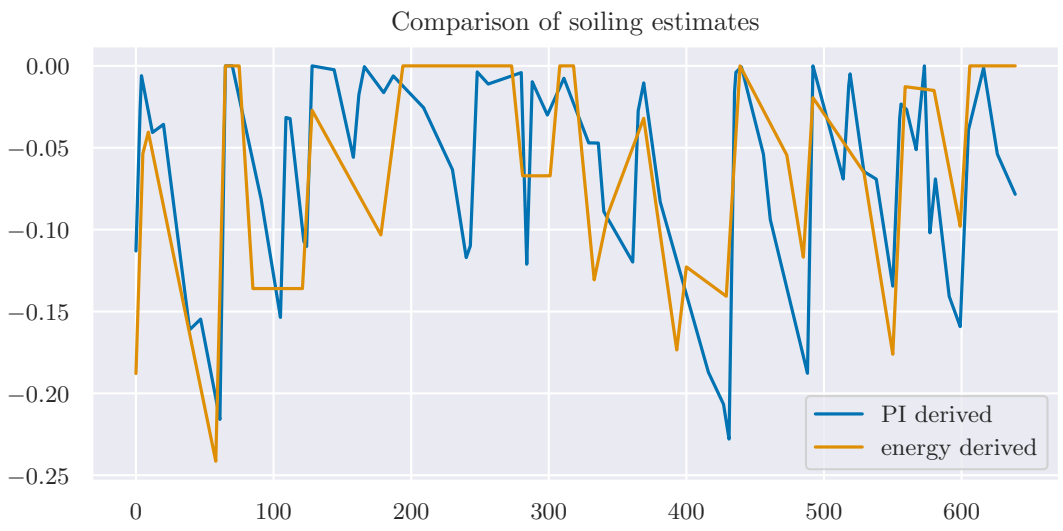

Figure 14: Comparing the estimation of the soiling trend estimated from the unlabeled energy data to the trend derived from the soiling performance index.

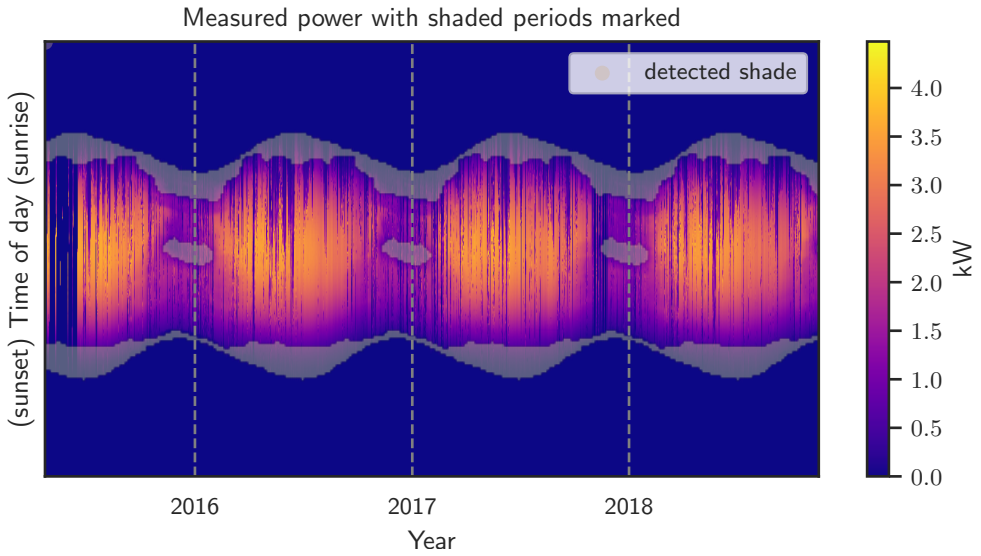

Figure 15: A heatmap of PV power annotated to mark periods of time that are significantly shaded.

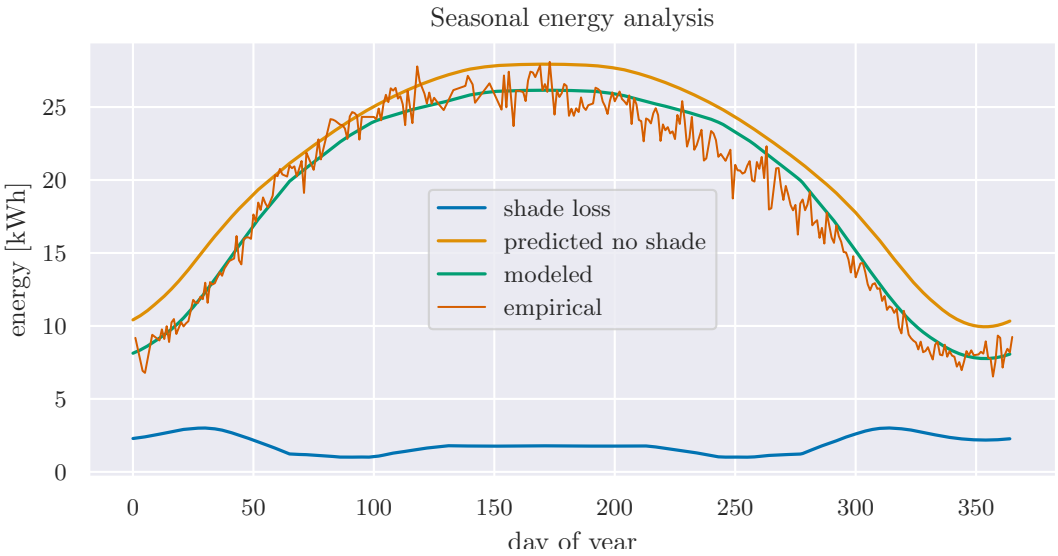

Figure 16: Analysis of the shade OSD components that estimates the seasonal energy lost to shade.

of the signal decomposition that shows the energy lost to shade through different times of year. We are currently working on a manuscript documenting this work and on validating the algorithm against PV sites with known model configurations.

4.4.4 Inverter clipping

See §4.2.3.

4.4.5 Inverter and data aquisition faults

See §4.2.3.

4.5 System location and orientation estimation

We have developed three algorithms for estimating latitude, longitude, and system orientation for fixed-tilt systems, based only on measured power data (and knowledge of the local times zone where the time stamps for the measurements were generated). These algorithms have been published in [16], and the code implementation is in a standalone software package called PV System Profiler [37]. Validation results are availbe in the paper. The algorithms may be invoked directly from Solar Data Tools by through the `DataHandler` methods `estimate_latitude`, `estimate_longitude`, `estimate_orientation`, and `estimate_location_and_orientation`. The latitude and longitude estimators are completely stand alone; they can be estimated indepenently and do not depend on anything other than the power data. The location estimator (tilt and azimuth) requires latitude as an input. The software allows the user to either estimate location based on the latitude derived from the data or based on a known latitude.

4.5.1 Sunrise and sunset estimation

Our main contribution to this area of research is establishing the fundamental importance of accurately estimating sunrise and sunset times and providing a robust algorithm based on

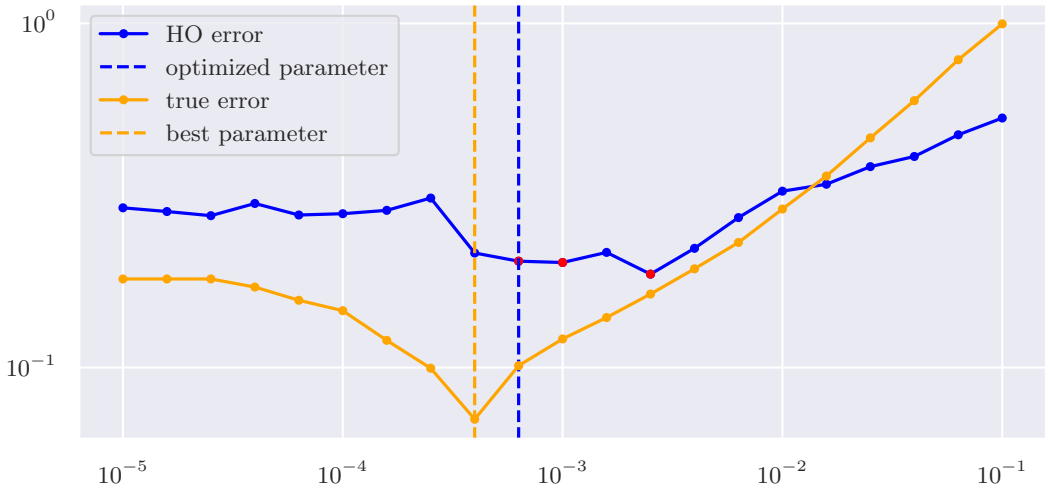

Figure 17: Using holdout validation to selection a turn-on threshold for a data set. The blue line represents the holdout validation used in OSD. The orange line shows what the actual error is versus the true sunrise and sunset times (only possible to calculate if actual latitude and longitude are known). The dashed lines show the parameter value picked through holdout validation (blue) and the best parameter value if the true values were known ahead of time (orange).

OSD for doing so. The number of daylight hours a location sees on a given day of the year only depends on its latitude, while the time of solar noon on a given day of year depends only on the longitude. Both these quantities are calculated from sunrise and sunset times (the difference and the average respectively).

Other researchers have used a simple threshold to estimate the sunrise and sunset times each day, *e.g.* 1% of the maximum value or 50 watts-per-square-meter of irradiance. Because we assume unlabeled data, we do not assume access to irradiance data, and so we must define a threshold based on the power signal along. However, practical experience has shown that different systems have different turn on characteristics, meaning that different data instances would have a different optimal turn-on threshold.

Therefore, we use OSD with holdout validation to test the prediction error of different turn on thresholds, and chose the one that minimized the holdout error. After selecting a threshold value, we run the OSD model with all available data to obtain a robust estimate of sunrise and sunset times that is smooth and 365-day periodic. This algorithm is implemented in `solardatatools.algorithms.sunrise_sunset_estimation`. When we have access to the true location of a site, the software will automatically display the true value in addition to the estimated values.

In figure 17, we show holdout validation results for selecting the turn-on threshold for the data set shown in 15. We see that the holdout validation provides a noisy estimate of the true error versus turn-on value, and that the value selected through holdout validation (blue dashed line) is close to the best possible value (orange dashed line). In figure 18, we

show result of running the final OSD estimation after selecting an optimal threshold. The RMS error in estimated sunrise and sunset times for this example over the entire data set is about 0.1, or about 6 minutes.

4.5.2 Latitude

The latitude of a system on a given day is related to the number of daylight hours and the declination angle by

$$N = \frac{2}{15}\cos^{-1}\left(-\tan\phi\tan\delta\right), \tag{9}$$

where N is the number of daylight hours, ϕ is the latitude, and δ is the declination angle [38]. The declination angle is a function of the day of the year [39, 40] and is therefore known. An estimate of N is generated for each day in the data set according to the procedure described in §4.5.1. We then solve (9) for ϕ on each day, and the final estimate of system latitude is taken as the median of these values.

4.5.3 Longitude

The longitude of a system on a given day is related to the difference between solar time and standard time by

$$\text{solar time} - \text{standard time} = 4\left(\psi_{ST} - \psi\right) + E, \tag{10}$$

where ψ_{ST} is the standard meridian for the local time zone, ψ is the longitude and parameter E is the equation of time. The equation of time is related to parameter B by

$$\begin{aligned} E = 229.2(0.000075 + 0.001868\cos B - 0.032077\sin B \\ -0.014615\cos 2B - 0.04089\sin 2B), \end{aligned} \tag{11}$$

where B is defined as

$$B = \frac{n-1}{360-365}, \tag{12}$$

and n is the day of the year [38]. The difference between solar time and standard time on a given day is equivalent to the time at which solar noon occurs on that day (by definition). So, we estimate solar noon each day as described in §4.5.1, and then we solve (10) for ψ on each day. The final estimate of system longitude is the median of these daily values.

4.5.4 Tilt and azimuth

The angle of incidence of a system is related to its latitude, tilt, azimuth, declination angle, and hour angle by

$$\begin{aligned} \cos\theta = \sin\delta\sin\phi\cos\beta - \sin\delta\cos\phi\sin\beta\cos\gamma + \\ \cos\delta\cos\phi\cos\beta\cos\omega + \cos\delta\sin\phi\sin\beta\cos\gamma\cos\omega + \\ \cos\delta\sin\beta\sin\gamma\sin\omega, \end{aligned} \tag{13}$$

where θ is the angle of incidence, β is the tilt, γ is the azimuth, ϕ is the latitude, and ω is the hour angle (defined as $15^{\circ}\times$ number of hours before/after noon) [38]. The unknowns in this equation are therefore θ, β and γ. We provide a method for estimating $\cos(\theta)$ from the observed power, then allowing us to estimate β and γ through non-linear least squares.

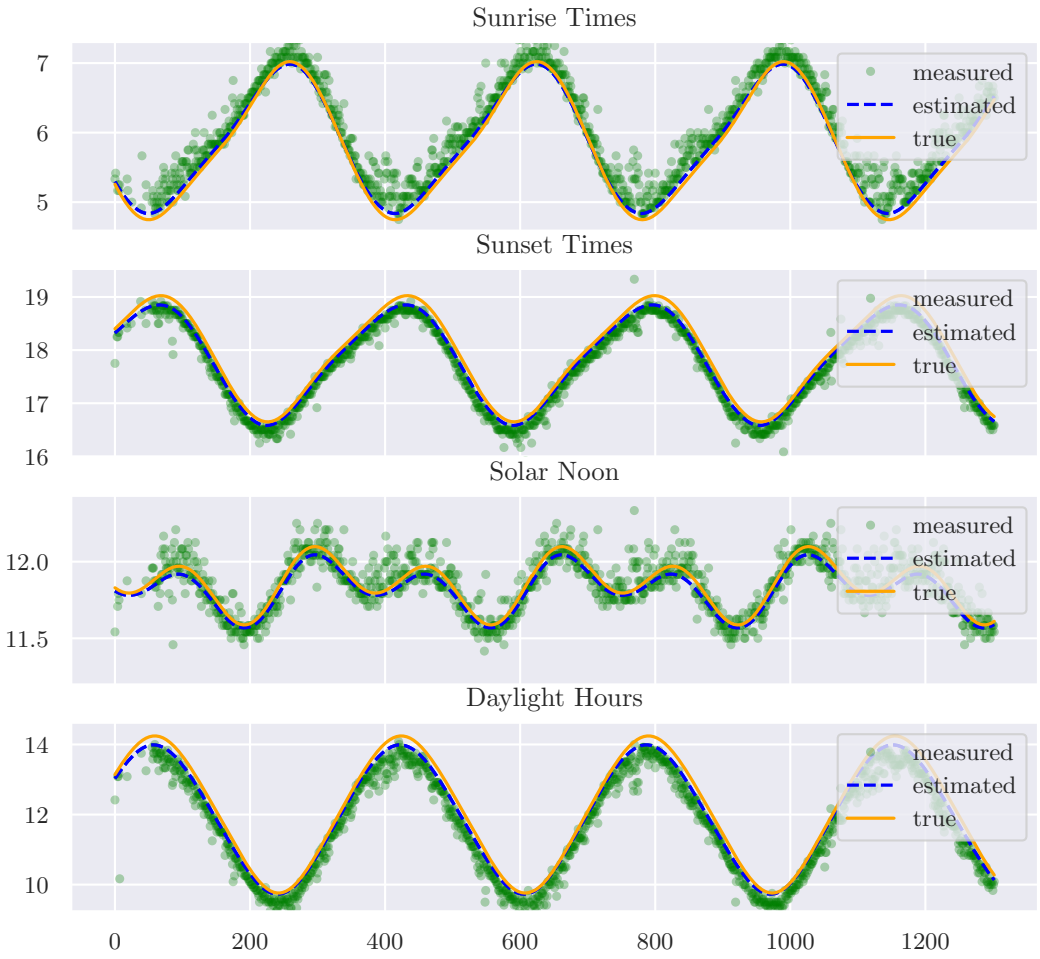

Figure 18: Application of OSD to sunrise and sunset time estimation, and the derived values of solar noon and number of daylight hours. The green dots are the values obtained through applying the optimized threshold to the data. The blue lines are the estimate generated by OSD. The orange lines are the true values for this site location.

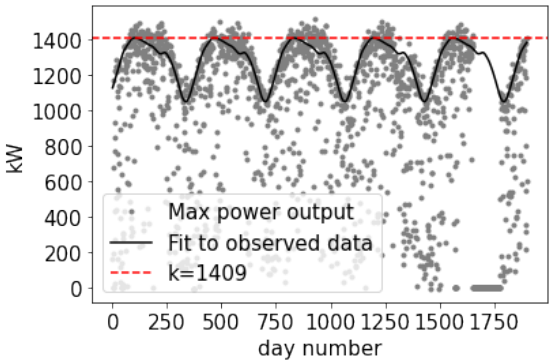

Figure 19: Visualization of the method for estimating k, the parameter that relates measured power and angle of incidence. The black line is estimated using signal decomposition.

To estimate $\cos\theta$ from the measured power data, we assume a very simple model relating angle-of-incidence and power

$$p = k\cos\theta, \tag{14}$$

where p is the system power and k is a parameter we'd like to estimate. Given an estimate for k, we can plug (14) into (13), and solve for the remaining parameters. We estimate k as follows:

1. calculate the daily maximum power output
2. utilize OSD to generate a smoothed, robust estimate of daily maximum power
3. the maximum value of the smoothed estimate is taken to be k

This processes is visualized for a representative site in Figure 19.

The explicit assumption in this process is that the angle-of-incidence is equal to zero ($\cos\theta = 1$) at least one time over the course of an entire year. This is a reasonable assumption for many PV systems located in North America. To support this assertion, we show here an analysis of two representative locations, Miami, FL and Olympia, WA. Figures 20 and 21 show the maximum value of $\cos\theta$ for various tilts and azimuths in Miami and Olympia respectively. In Figures 22 and 23, the tilts and azimuths corresponding to $\mathbf{max}(\cos\theta) \geq 0.99$ are marked in white, representing the space of tilts and azimuths at these locations for which the assumption is valid.

Finally, the parameters β and γ are estimated by applying non-linear least-squares to (13), using the algorithm implemented in the Scipy Optimize package [41].

5 Significant Accomplishments and Conclusions

This project lies at the intersection of three domains: PV system performance, signal processing/data analysis, and coding/software development. We have made significant contributions to the area of signal processing and optimization-based data analytics. These contributions have been motivated by a particular data challenge that is practically impacting the PV industry today and by solving particular estimations problems of interest to digital operations and maintenance of PV fleets. To make these contributions generally useful to industry and

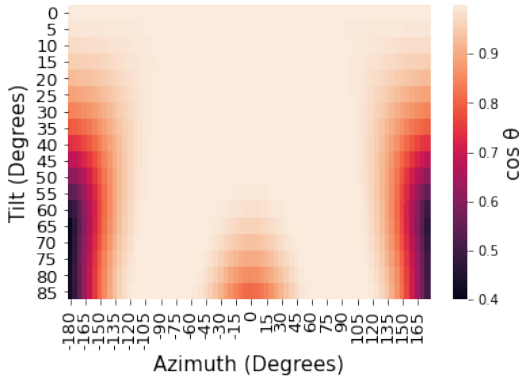

Figure 20: Maximum value $\cos\theta$ observed in a year as a function of tilt and azimuth for a PV system located in Miami, FL.

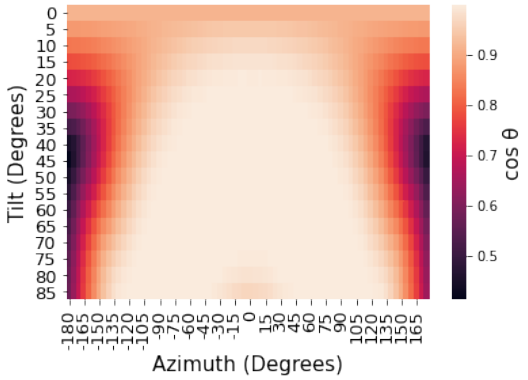

Figure 21: Maximum value $\cos\theta$ observed in a year as a function of tilt and azimuth for a PV system located in Olympia, WA.

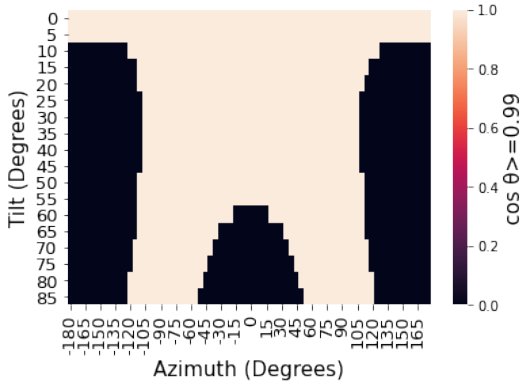

Figure 22: Configurations of tilt and azimuth that produce $\cos\theta \approx 1$, as a function of tilt and azimuth for a PV system located in Miami, FL.

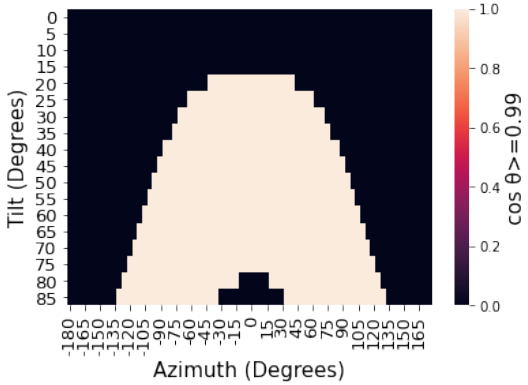

Figure 23: Configurations of tilt and azimuth that produce $\cos\theta \approx 1$, as a function of tilt and azimuth for a PV system located in Olympia, WA.

the research community, we have make code implementation and software development a priority of the project, rather than an afterthought or something left to future researchers and engineers.

Optimization-based signal decomposition. The development of optimization-based signal decomposition (OSD) and the forthcoming monograph are highly significant developments that have enabled the majority of the algorithms developed in this project and are expected to have wide-ranging impacts in many areas in addition to PV data science. This research is related to many generalized methods that are applied throughout numerous domains such as trend filtering, seasonal-trend decomposition, regularized regression, sparse dictionary methods, contextually supervised source separation and more. These connections and relationships are explained in the monograph, and OSD may be thought as a general framework that includes all these specific methods as subtypes. That is, these methods may be themselves expressed as OSD problems. In this way, OSD generalizes and ties together many previously proposed analytical methods, while providing an extensible language for designing new analytics as well as a dedicated algorithm for solving the induced optimization problem.

Applications of OSD to PV data science. OSD has enabled numerous approaches that are well-suited for analyzing the performance and reliability of PV systems based on unlabeled power data, as detailed in §4.2 and §4.4. We are still in the process of finalizing the manuscripts for the soiling and shade loss factor estimation algorithms, but the algorithms themselves are published in open-source code and have been internally validated.

Development of open-source software. The project has produces three open-source software packages: Solar Data Tools (SDT), Statistical Clear Sky, and PV System Profilier. SDT is the main piece of software, which contains the `DataHandler` object, for managing the data processing pipeline and for invoking the other analyses. As of September 2021, SDT had about 300 downloads per month on PyPI and 179 downloads in total on Anaconda. The project has also been starred 23 times on GitHub.

Statistical clear sky fitting algorithm. The development of the statistical clear sky fitting (SCSF) algorithm, and it's related application to estimating PV system degradation has made a significant impact on the PV data science community. The paper on applying SCSF to degradation analysis [7] has already been cited 17 times as of September 2021.

Estimation of system location and orientation from data. While other researchers have shown that it is possible to estimate location and orientation of a PV system from measured power data, the existing methods were complex to implement and had not been released in software (open-source or otherwise). We made a significant contribution to this particular area of research by presenting a novel approach, based on OSD, for inferring daily sunrise and sunset times from power data using OSD (see § 4.5.1) that is far more accurate than previously established methods. Additionally, we have implemented our approaches in the PV System Profiler Python package, and streamlined the integration of the algoriothms with the SDT DataHandler, allowing the methods to be easily used by others.

6 Budget and Schedule

The following numbers are subect to change due to pending Year End closing. Travel, supplies, and other were underspend because of COVID-19. Most spending were on personnel and fringe that we were able to cover from areas of underspending.

Table 2: Budget and spending

Categories	*Budget*	*Spent*	*Unspent*
Personnel	499,338	523,148	(23,810)
Fringe	126,144	148,454	(22,310)
Travel	30,000	9,560	20,440
Equipment	-	-	-
Supplies	110,000	72,940	37,060
Contractual	50,000	11,000	39,000
Construction	-	-	-
Other	57,171	-	57,171
Total direct	872,653	765,103	107,551
Indirect	517,347	581,279	(63,932)
Total	**1,390,000**	**1,346,382**	**43,618**

Table 3: Schedule

Categories	*10/1/2018–9/30/2019*	*10/1/2019–9/30/2020*	*10/1/2020–9/30/2021*
Personnel	217,981	221,371	59,986
Fringe	53,776	54,972	17,396
Travel	15,000	15,000	-
Equipment	-	-	-
Supplies	55,000	55,000	-
Contractual	25,000	25,000	-
Construction	-	-	-
Other	28,094	29,077	-
Total direct	394,852	400,420	77,382
Indirect	225,580	229,149	62,618
Total	**620,431**	**629,569**	**140,000**

7 Path Forward

We are very pleased to report that PVInsight Phase 2, covering FY'22-24, has been approved by the U.S. Department of Energy's Solar Energy Technologies Office. In Phase

2 of PVInsight, we will be building on the work and experience of PVInsight, developing algorithms and tools to solve modern data challenges in the residential, commercial, and utility sectors of the photovoltaic (PV) solar industry. We will develop data science tools to enable cost-effective, fleet-scale operations and maintenance for all PV systems, inclusive of those systems that have lower data quality, are not well modeled, and are lacking reliable environmental data. We will continue to develop and improve the OSD framework for analyzing PV performance signals, which enables the analysis of unlabeled time-series data. Additionally, we are bringing the NREL RdTools developers in as project partners, and we will be working together to solve problems about scaling PV data science software to large-scale cloud deployments and develop standardized methods and metrics for comparing and contrasting the accuracy, performance, and data requirements of algorithms in the PV data science community.

We anticipate three outcomes from the next phase of research: (1) develop a novel approach to fleet-scale probabilistic modeling of power production, suitable for large scale fault detection and forecasting, (2) build the software and deployment packages to run algorithms developed under PVInsight (SETO #34368) and RdTools (SETO #30311 and #34348) on modern, large-scale cloud computing environments, and (3) building a public-facing validation hub, for testing and comparing statistical algorithms specifically related to retrospective performance analysis and real-time fault detection. Of critical importance to the second goal is the removal of MOSEK from the PVInsight software requirements. This work will focus on implementing the proposed ADMM-based algorithm for solving OSD problems discussed in §4.1.3.

8 Publications and Other Results

Papers.

- Peer-reviewed journal articles: [7]
- Conference proceedings: [33, 26, 16, 13, 16, 42, 43]

Talks and workshops.

- B. Meyers, "Tools for PV Data Science: Applied math, statistics, and signal processing for gaining insight from PV data," February 2019, PV Reliability Workshop
- B. Meyers, M. Deceglie, C. Deline, and D. Jordan, "Signal Processing on PV Time-Series Data: Robust Degradation Analysis without Physical Models," June 2019, 46th IEEE Photovoltaic Specialists Conference
- B. Meyers, E. Apostolaki-Iosifidou, L. Schelhas, "Solar Data Tools: Automatic Solar Data Processing Pipeline," June 2020, 47th IEEE Photovoltaic Specialists Conference
- B. Meyers, "PVInsight: Data Driven Aproaches for Analyzing PV System Performance and Reliability," October 2020, SPI Smart Energy Week Virtual Microconference: Is Data the New Bacon?
- B. Meyers and E. Kam-Lum, "Preliminary Application of PVInsight Automated Machine Learning for Soiling Detection in Desert Climate," February 2021, PV Reliability Workshop

- B. Meyers and S. Boyd, "Signal Decomposition via Distributed Optimization," April 2021, Invited talk at SLAC AI Seminar Series (online: `https://ml.slac.stanford.edu/ai-seminar`)
- B. Meyers, "Identification of Best Plane-of-Array Irradiance Sensor for PV System Performance Analytics," June 2021, 48th IEEE Photovoltaic Specialists Conference

Software.

- Solar Data Tools, "Automatic handling of PV data preprocessing, cleaning, and filtering, as well as tools for executing other PVInsight algorithms," `https://github.com/slacgismo/solar-data-tools`, DOI: 10.5281/zenodo.5534917
- Statistical Clear Sky, "Implements statistical clear sky fitting (SCSF) algorithm," `https://github.com/slacgismo/StatisticalClearSky`, DOI: 10.5281/zenodo.5056944
- PV System Profiler, "Implements estimation algorithms for determining system latitude, longitude, tilt, and azimuth from measured power," `https://github.com/slacgismo/pv-system-profiler`, DOI: 10.5281/zenodo.5484417

References cited

[1] M. Davis, C. Smith, B. White, R. Goldstein, X. Sun, M. Cox, G. Curtin, R. Manghani, S. Rumery, C. Silver, and J. Baca, *U.S. Solar market insight executive summary, 2020 year in review*. Wood Mackenzie and SEIA, 2021.

[2] M. G. Deceglie, D. Jordan, A. Nag, C. A. Deline, and A. Shinn, "RdTools: An open source python library for PV degradation analysis," National Renewable Energy Lab. (NREL), Golden, CO (United States), Tech. Rep., 2018.

[3] K. A. Klise, "Performance monitoring using Pecos," Sandia National Lab. (SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2016.

[4] T. Townsend, C. Whitaker, B. Farmer, and H. Wenger, "New performance index for PV system analysis," *Conference Record of the IEEE Photovoltaic Specialists Conference*, vol. 1, pp. 1036–1039, 1994.

[5] D. C. Jordan, M. G. Deceglie, and S. R. Kurtz, "PV degradation methodology comparison—a basis for a standard," in *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*, 2016, pp. 0273–0278.

[6] M. G. Deceglie, L. Micheli, and M. Muller, "Quantifying soiling loss directly from PV yield," *IEEE Journal of Photovoltaics*, vol. 8, no. 2, pp. 547–551, 2018.

[7] B. Meyers, M. Deceglie, C. Deline, and D. Jordan, "Signal processing on pv time-series data: Robust degradation analysis without physical models," *IEEE Journal of Photovoltaics*, vol. 10, no. 2, pp. 546–553, 2020.

[8] S. Lindig, D. Moser, A. J. Curran, K. Rath, A. Khalilnejad, R. H. French, M. Herz, B. Müller, G. Makrides, G. Georghiou, A. Livera, M. Richter, J. Ascencio-Vásquez, M. van Iseghem, M. Meftah, D. Jordan, C. Deline, W. van Sark, J. S. Stein, M. Theristis, B. Meyers, F. Baumgartner, and W. Luo, "International collaboration framework for the calculation of performance loss rates: Data quality, benchmarks, and trends (towards a uniform methodology)," *Progress in Photovoltaics: Research and Applications*, vol. 29, no. 6, pp. 573–602, 2021.

[9] D. C. Jordan, C. Deline, S. R. Kurtz, G. M. Kimball, and M. Anderson, "Robust PV Degradation Methodology and Application," *IEEE Journal of Photovoltaics*, vol. 8, no. 2, pp. 525–531, 2018.

[10] B. Marion, "Clear Sky Irradiance Year-to-Year Variations and Trends," in *2020 47th IEEE Photovoltaic Specialists Conference (PVSC)*, vol. 2020-June, no. November. IEEE, jun 2020, pp. 0481–0484. [Online]. Available: https://ieeexplore.ieee.org/document/9300858/

[11] F. Antonanzas-Torres, R. Urraca, J. Polo, O. Perpiñán-Lamigueiro, and R. Escobar, "Clear sky solar irradiance models: A review of seventy models," *Renewable and Sustainable Energy Reviews*, vol. 107, pp. 374–387, 2019.

[12] L. Micheli, M. Theristis, A. Livera, J. S. Stein, G. E. Georghiou, M. Muller, F. Almonacid, and E. F. Fernandez, "Improved PV Soiling Extraction through the Detection of Cleanings and Change Points," *IEEE Journal of Photovoltaics*, vol. 11, no. 2, pp. 519–526, 2021.

[13] E. Kam-Lum, B. Meyers, D. Cosme, B. Aissa, and G. Scabbia, "Soiling rate determination from referenced systems in desert climate using pvinsight soiling algorithm," in *2021 IEEE 48th Photovoltaic Specialists Conference (PVSC)*, 2021, pp. 2552–2554.

[14] M. K. Williams, S. L. Kerrigan, A. Thornton, and L. Energy, "Automatic detection of pv system configuration," in *Proceedings of World Renewable Energy Forum*, 2012.

[15] N. Haghdadi, J. Copper, A. Bruce, and I. MacGill, "A method to estimate the location and orientation of distributed photovoltaic systems from their generation output data," *Renewable Energy*, vol. 108, pp. 390–400, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.renene.2017.02.080

[16] A. Londono-Hurtado, B. Meyers, E. Apostolaki-Iosifidou, and R. Flottemesch, "Estimation of photovoltaic system location and orientation from power signals," in *2021 IEEE 48th Photovoltaic Specialists Conference (PVSC)*, 2021, pp. 1807–1812.

[17] E. Hasselbrink, M. Anderson, Z. Defreitas, M. Mikofski, Y. C. Shen, S. Caldwell, A. Terao, D. Kavulak, Z. Campeau, and D. Degraaff, "Validation of the PVLife model using 3 million module-years of live site data," *Conference Record of the IEEE Photovoltaic Specialists Conference*, pp. 7–12, 2013.

[18] B. Meyers, "SLAC AI seminar series: Signal decomposition via distributed optimization," April 2021. [Online]. Available: https://ml.slac.stanford.edu/ai-seminar

[19] M. Wytock and J. Z. Kolter, "Contextually supervised source separation with application to energy disaggregation," *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1–10, 2013. [Online]. Available: http://arxiv.org/abs/1312.5023

[20] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2009.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011. [Online]. Available: http://arxiv.org/abs/1408.2927

[22] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[23] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[24] E. D. Andersen and K. D. Andersen, *The Mosek Interior Point Optimizer for Linear Programming: An Implementation of the Homogeneous Algorithm*. Boston, MA: Springer US, 2000, pp. 197–232. [Online]. Available: https://doi.org/10.1007/978-1-4757-3216-0_8

[25] B. Meyers, D. Serbetcioglu, J. Goncalves, and A. Londono-Hurtado, "slacgismo/solar-data-tools: Solar data tools," Sep. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5504444

[26] B. E. Meyers, E. Apostolaki-Iosifidou, and L. T. Schelhas, "Solar data tools: Automatic solar data processing pipeline," in *2020 47th IEEE Photovoltaic Specialists Conference (PVSC)*, 2020, pp. 0655–0656.

[27] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.

[28] The pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[29] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[30] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[31] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KKD-96 Proceedings*, vol. 96, no. 34, 1996, pp. 226–231.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] B. Meyers, M. Tabone, and E. C. Kara, "Statistical Clear Sky Fitting Algorithm," *2018 IEEE 45th Photovoltaic Specialist Conference (PVSC)*, pp. 1–6, 2018. [Online]. Available: http://www.wcpec7.org/eWCPEC/manuscripts/MeWCPEC930{_}0521112519.pdf

[34] B. Meyers, J. Goncalves, D. Serbetcioglu, and T. Takahashi, "slacgismo/statisticalclearsky: Statistical clear sky," Sep. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5056944

[35] M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models," *Foundations and Trends in Machine Learning*, vol. 9, no. 1, pp. 1–118, 2016. [Online]. Available: https://web.stanford.edu/~boyd/papers/glrm.html

[36] B. Meyers and J. Hoffimann, "Short time horizon solar power forecasting," *Neural Netw*, vol. 3, p. 2340, 2017.

[37] A. Londono-Hurtado, B. Meyers, E. Apostolaki-Iosifidou, and D. Serbetcioglu, "slacgismo/pv-system-profiler: PV system profiler," Sep. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5484417

[38] J. A. Duffie and W. A. Beckman, *Solar engineering of thermal processes*. John Wiley & Sons, 2013.

[39] P. I. Cooper, "The Absorption of Solar Radiation in Solar Stills," *Solar Energy*, vol. 12, no. 3, 1969.

[40] J. Spencer, "Fourier Series Representation of the Position of the Sun," *Search*, vol. 2, no. 5, pp. 162–172, 1971.

[41] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[42] B. E. Meyers, "Identification of best plane-of-array irradiance sensor for pv system performance analytics," in *2021 IEEE 48th Photovoltaic Specialists Conference (PVSC)*, 2021, pp. 1208–1212.

[43] ——, "Dimensionality reduction and clustering analysis of daily pv power signals," in *2021 IEEE 48th Photovoltaic Specialists Conference (PVSC)*, 2021, pp. 1217–1221.