



2024 PVPMC

Solar Data Tools

Version 1 release, features, applications, and future work

Bennet Meyers / Staff Scientist / Applied Energy Division

May 2024

bennetm@stanford.edu

Agenda

1. Introduction to Solar Data Tools
2. Brief overview of signal decomposition
3. New SDT features available as of version 1.3.2
4. New methods for executing SDT: Dask and Marimo

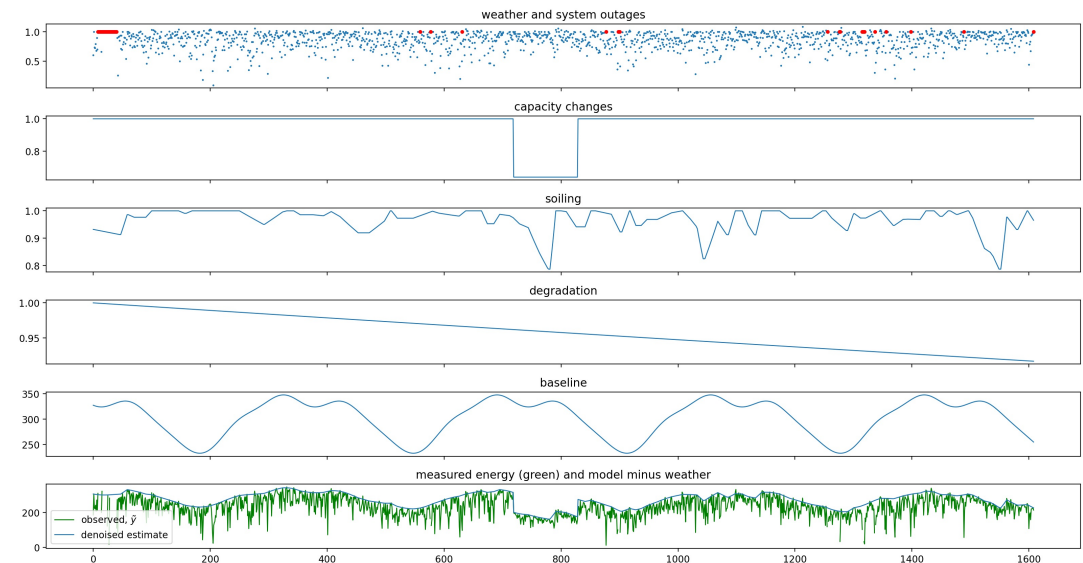
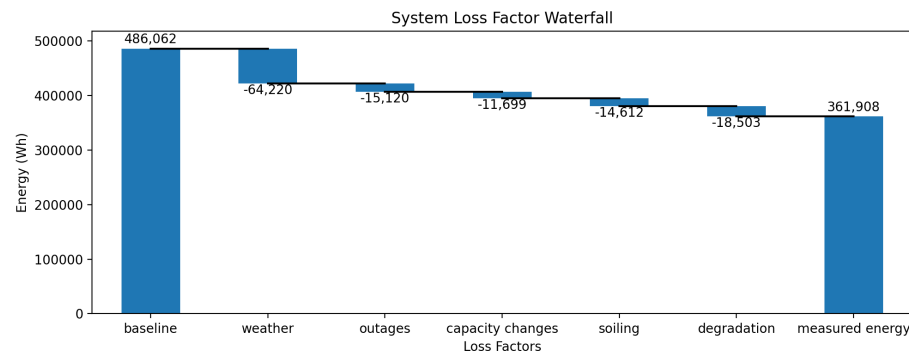
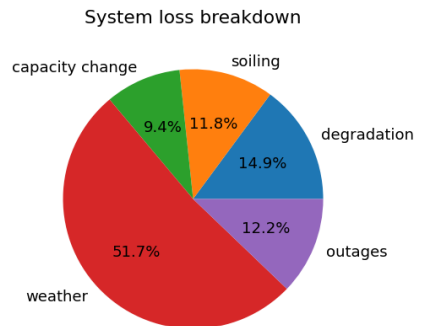
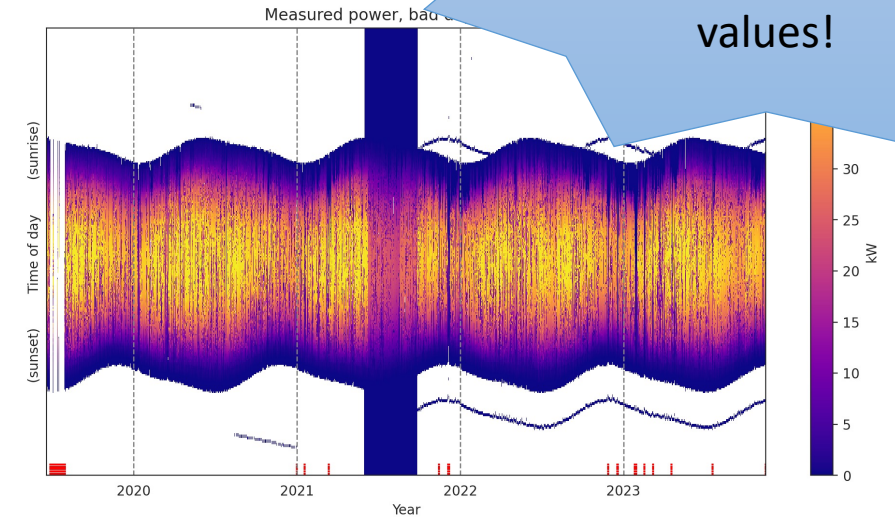
1

Introduction to Solar Data Tools

Solar Data Tools (SDT)

- Open-source python library for analyzing PV power (and irradiance) time-series data
- Methods for data cleaning, filtering, plotting, and analysis
- Goal to enable analysis of **unlabeled** PV data—no model, no meteorological data, no performance index required
- Takes a **statistical signal processing** approach
- Data processing steps are largely **pre-defined** and **automatic** regardless of system type—from utility tracking systems to multi-pitch rooftop systems

Easily make a heat map of any data, even with missing values!



The Data Handler Object

webinar link →



<https://youtu.be/XKbqILAEwOQ?si=HGdhrZEwLp9EGnhN>

- Most user interaction with SDT will be through the **data handler**

```
from solardatatools import DataHandler
df = ... # load data as pandas data frame
dh = DataHandler(df)
dh.run_pipeline()
dh.run_loss_factor_analysis()
```

- Main pipeline runs preprocessing, cleaning, and quality checks
 - Includes heatmap/matrix embedding, sunrise/sunset estimation, clear day labeling, clipping detection, and more
- After pipeline is run, user can invoke other analyses (loss factors, location estimation, etc.) or plotting functions
- See recent SETO webinar (QR code) for full demo of SDT features and code

get started →

```
pip install solar-data-tools
```

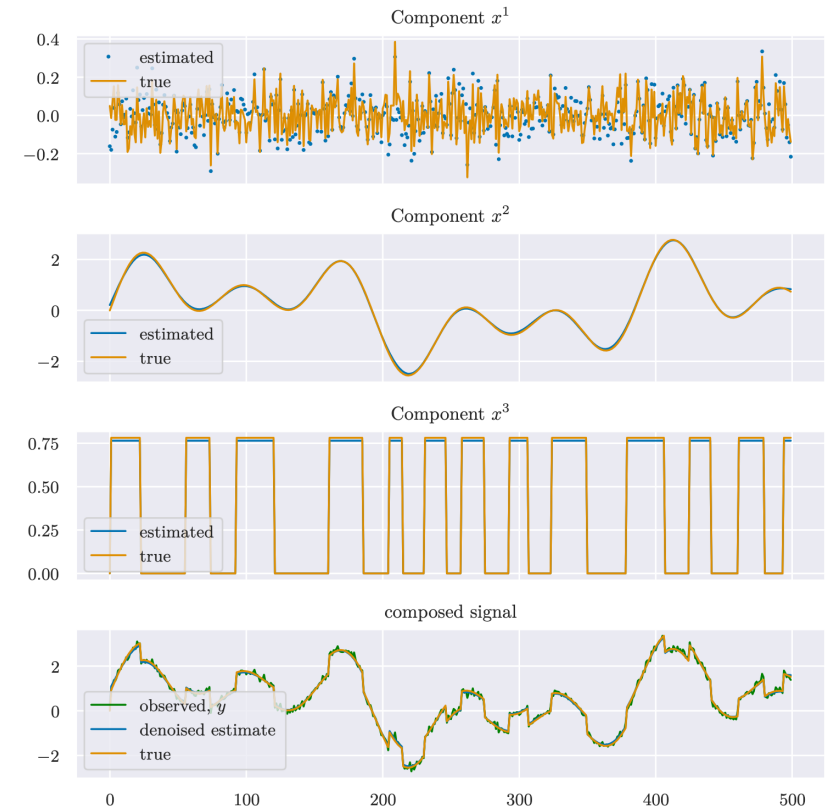
2

Brief overview of signal decomposition

Signal Decomposition

A modern take on an age-old problem

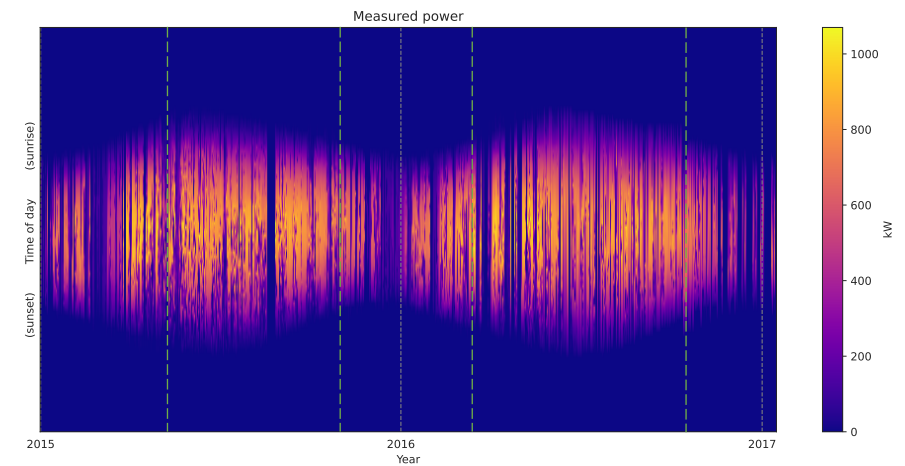
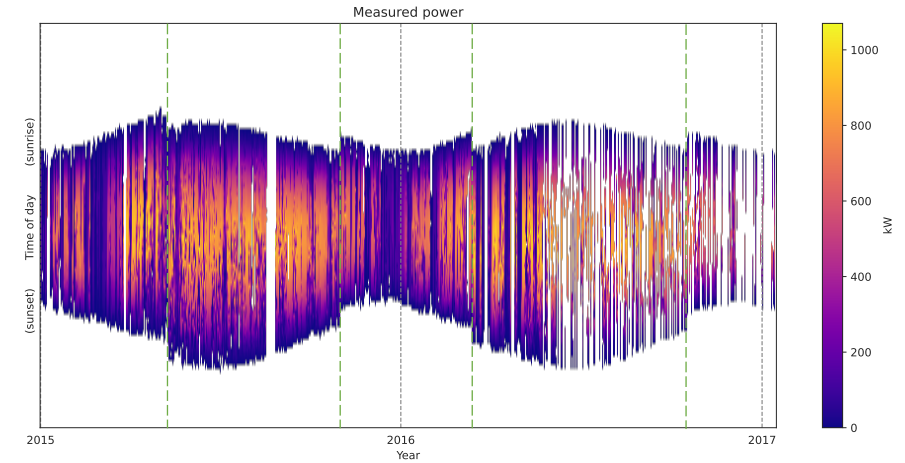
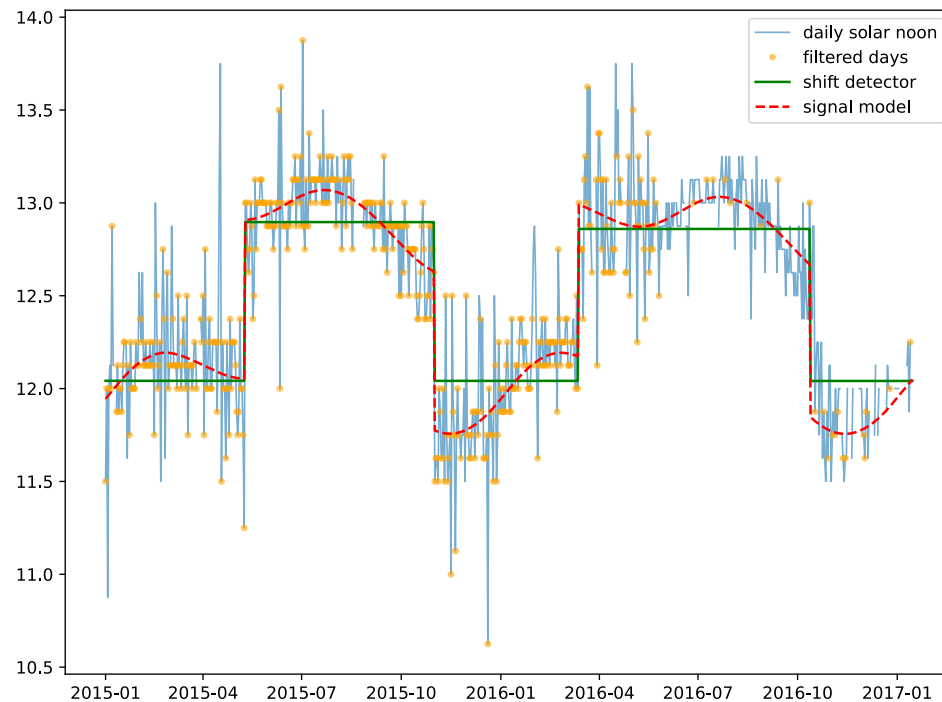
- Tutorial: <http://signal-decomp-tutorial.org/>
- Monograph: https://web.stanford.edu/~boyd/papers/sig_decomp_mprox.html
- Under the hood of SDT, we solve a many different *signal decomposition* problems over the data
- Enables everything from the time-shift detection analysis to the loss factor analysis
- All these problems use a principled approach and **modeling language** to statistically **decomposing** discrete time signals
- Theory and modeling language developed under this project
- Modeling language and approach widely applicable outside of solar



An illustrative example of signal decomposition a toy data set, reproduced from the monograph

SD Example: Time Shift Detection

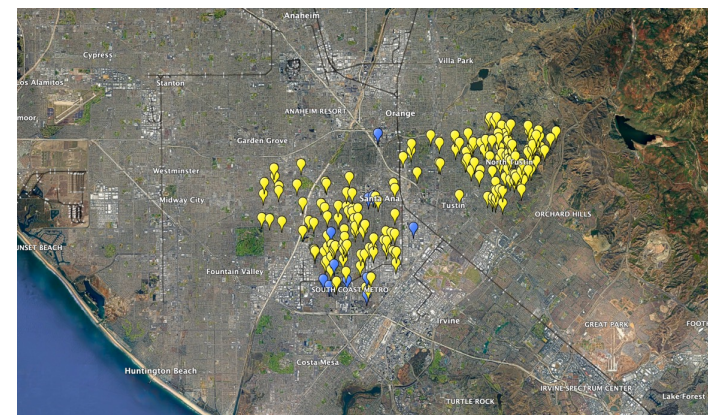
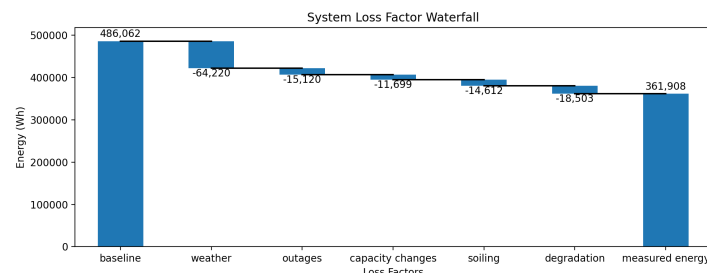
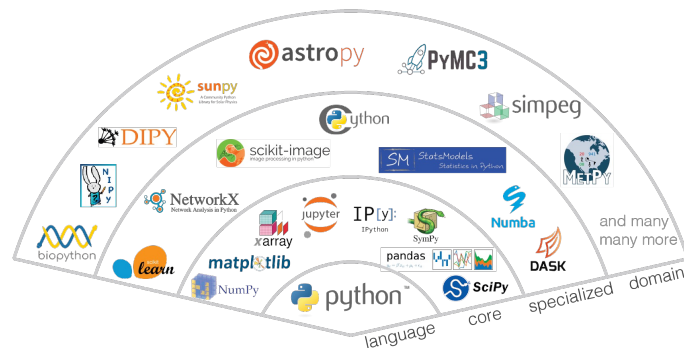
- Estimate solar noon from data
- Fit **three component SD model**—piecewise constant, smooth & periodic, and a noise term
- Holdout validation used to set problem weights



3

New SDT features as of v1.3.2

Overview of new features



Fully open-source stack

- Solvers (and all other dependencies) now free and open-source
- MOSEK supported for back compatibility but not required
- Alignment with Scipy “ecosystem”
- Simple pip install → easy to template and scale

New loss factor analysis

- See upcoming PVSC paper/talk
- Form SD problem on daily energy
- 5 losses modeled, e.g. degradation
- Replaces SCSF module (deprecated)
- Monte Carlo used for degradation uncertainty
- Shapely values used for loss allocation (waterfall plot)

Open-source data set

- 1971 inverters, 240 site sites
- Primarily residential (yellow) with some light commercial (blue)
- Sign up for API key to use new function in `soldatataools.dataio`
- PVDB: <https://pvdb.slacgismo.org>
- Email slacgismotutorials@gmail.com with your information and use case

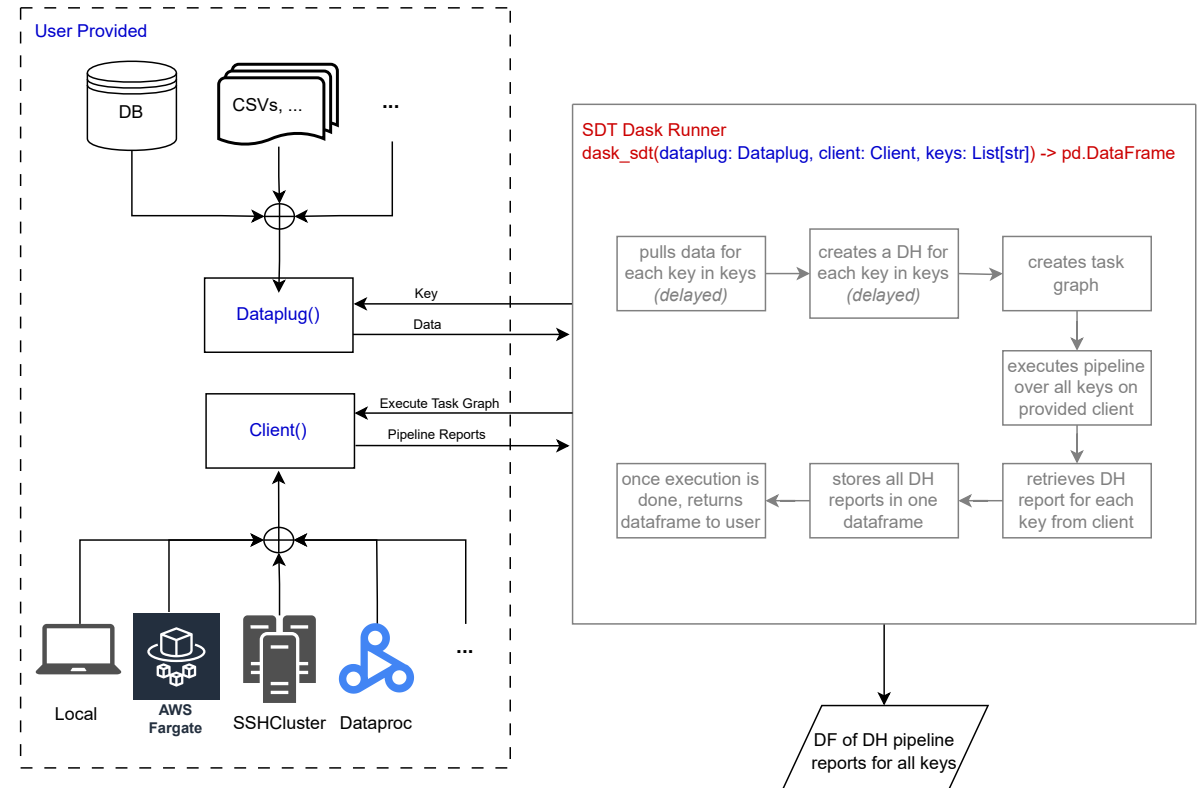
4

New methods for executing SDT: Dask and Marimo

Coordination of many SDT runs

Using Dask to parallelize workflows from single machines to cloud VM clusters

- Parallel execution of Solar Data Tools
- Three main components
 - SDT runner: defines pipeline
 - Data plug(s): defines data source
 - Client(s): defines compute
- Provide 4 example data plugs (local CSVs, S3 bucket, PVDAQ API, PVDB) and template + docs to make others
- Provide 3 example clients (local, AWS Fargate, and Azure) and template + docs to make more



Example basic usage of SDT Runner Tool

Designed for interoperability with various data plugs and compute clients

```
from sdt_dask.dataplugs.S3Bucket_plug import S3Bucket
from sdt_dask.clients.aws.fargate import Fargate
from sdt_dask.dask_tool.sdt_dask import SDTDask

# Define variables
...

# Defined S3 Bucket dataplug
data_plug = S3Bucket(bucket_name=BUCKET)

# Get all keys from S3 bucket (run all files)
key_list = data_plug._pull_keys()
KEYS = [(key,) for key in key_list]

# Dask Fargate client Setup
client_setup = Fargate(image=IMAGE,
                        tags=TAGS,
                        vpc=VPC,
                        region_name=AWS_DEFAULT_REGION,
                        environment=ENVIRONMENT,
                        n_workers=WORKERS,
                        threads_per_worker=THREADS_PER_WORKER
)

# Dask Local Client Initialization
client = client_setup.init_client()
```

```
# Dask Tool initialization and set up
dask_tool = SDTDask(data_plug=data_plug,
                    client=client,
                    output_path=" ../results/")
dask_tool.set_up(KEYS, fix_shifts=True, verbose=VERBOSE)

# Dask Tool Task Compute
dask_tool.get_result()
```

Point and click UI in Python with Marimo

Write/develop in a notebook

app.py

site key system key execute

```
1 mo.hstack([site_select, sys_select, execute_button])
```

```
1 keys = pd.read_csv("inputs/system_counts_per_site.csv")
2 keys.columns = ["Site ID", "System Count"]
3 system_list = mo.ui.table(keys, page_size=8)
```

```
1 cache_list = mo.ui.table(get_cache_history(), page_size=8, selection="single")
```

System List

Cache List

☐	Site ID	System Count
☐	001C4B008A5	11
☐	001C4B001069	2
☐	0022F2000C2D	4
☐	0022F200152D	3
☐	0022F2002DCE	4
☐	0022F2002E82	1
☐	0022F2005446	6
☐	0022F2005C54	11

240 items << < Page 1 of 30 > >> [Download](#)

```
1 mo.ui.tabs({"System List": system_list, "Cache List": cache_list})
```

DATA SET REPORT

Length5.47 years

Deploy as an application

site key system key execute [data viewer](#) [losses](#) [daily data quality](#) [capacity changes](#) [clipping analysis](#)

System List

Cache List

☐	Site ID	System Index
☐	TADBC1078041	0
☐	ZT162085000441C0944	14
☐	ZT162085000441C0944	20
☐	TAEEC1018327	1
☐	TABA01148077	1

[Download](#)

Measured power

start day number of days

Done, and results cached!

site: TABA01148077, system: 1

DATA SET REPORT

length5.47 years

capacity estimate3.12 kW

data sampling5 minutes

quality score0.97

clearness score0.48

inverter clippingFalse

clipped fraction0.00

capacity changesFalse

data quality warningTrue

time shift errorsFalse

time zone errorsFalse

SDT PVDB app

If time allows, live demo: <http://127.0.0.1:2720/>

- Connects to SLAC PVDB, hosted Redshift database with open-source PV data
- Explore the Southern California fleet data set via point-and-click and learn about SDT features including
 - Data cleaning/labeling
 - Inverter clipping detection
 - Time shift detection
 - Capacity change detection
 - Loss factor analysis, including degradation estimation
- Check it out here! <https://github.com/slacgismo/sdt-pvdb-app>
- Please request an API key and follow the instructions in the readme!



Thank you!

Team Members: Michael Deceglie (NREL), Sara Miskovich (SLAC), Kirsten Perry (NREL), Mehmet Ogut (Stanford), Aramis Dufour (Stanford), Duncan Ragsdale (SLAC), Mitchell Victoriano (SLAC)

CMU students: Nimish Yadav, Victor-Haoyang Lian, Haoxi Zhang, Jose St Louis, Junlin Luo, Tristan Lin, Chengcheng Ding, Xiao Ming

Past SDT Developers: Derin Serbetcioglu, Jonathan Goncalves, Claire Berschauer, Alejandro Londono-Hurtado, Tadatoshi Takahashi, Elpiniki Apostolaki-Iosifidou, David Jose Florez Rodriguez

This work is 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 38529

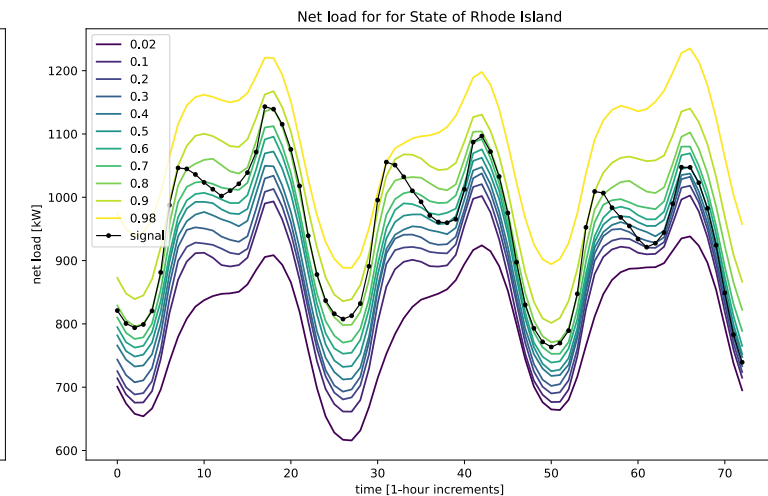
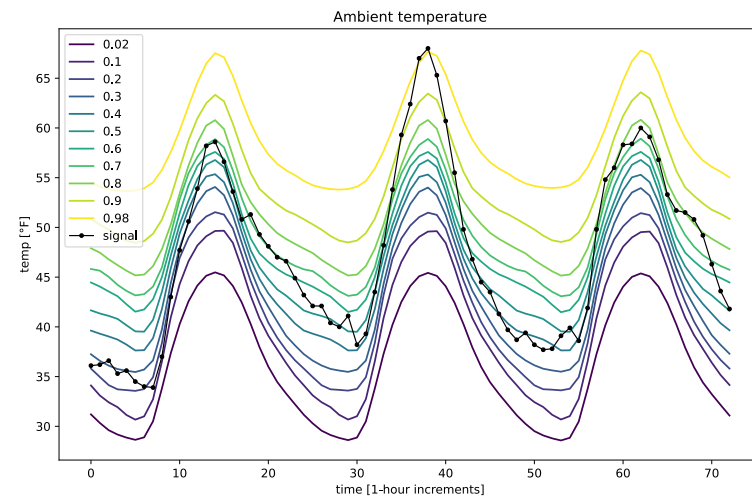
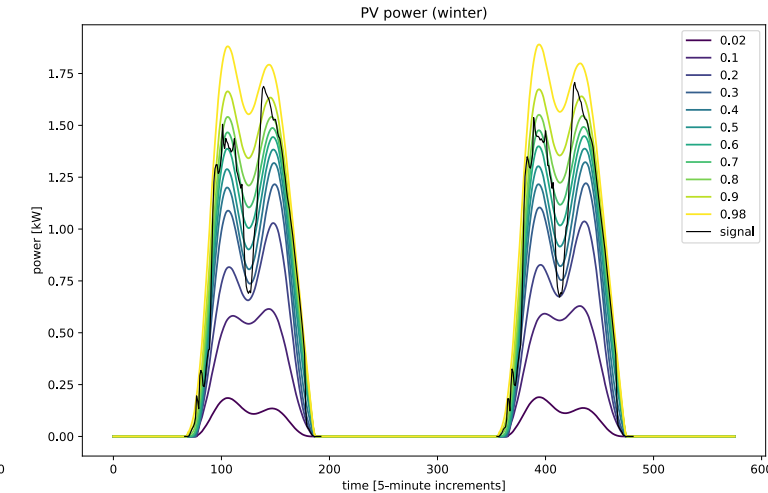
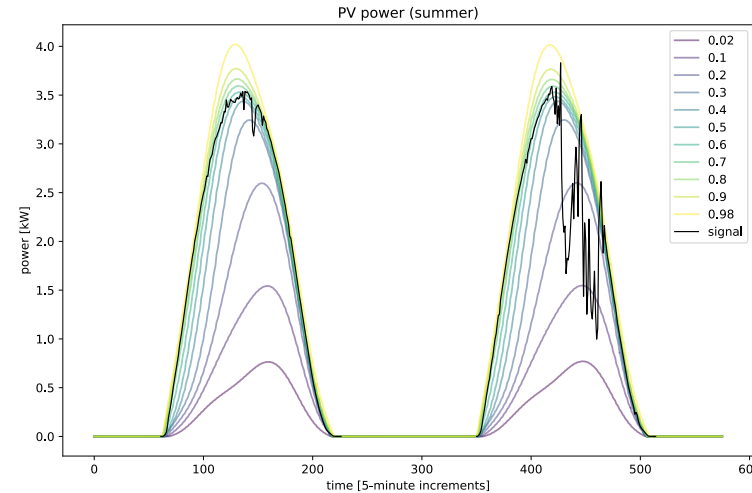
5

Current research and future work

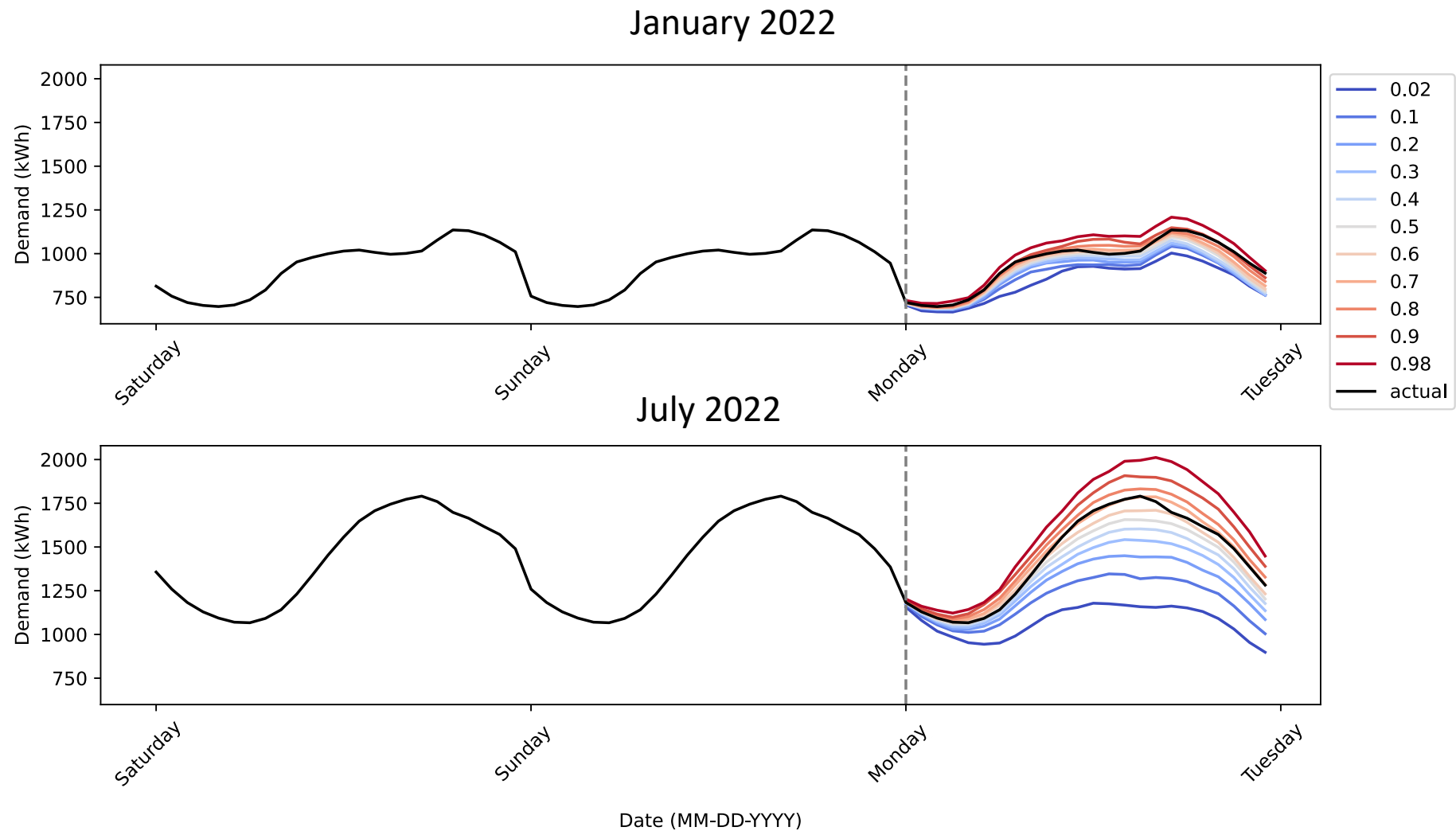
New Work: Multi-/Quasi-Periodic Probability Models

Adaptable, widely applicable approach to modeling (quasi-)periodic probability distributions

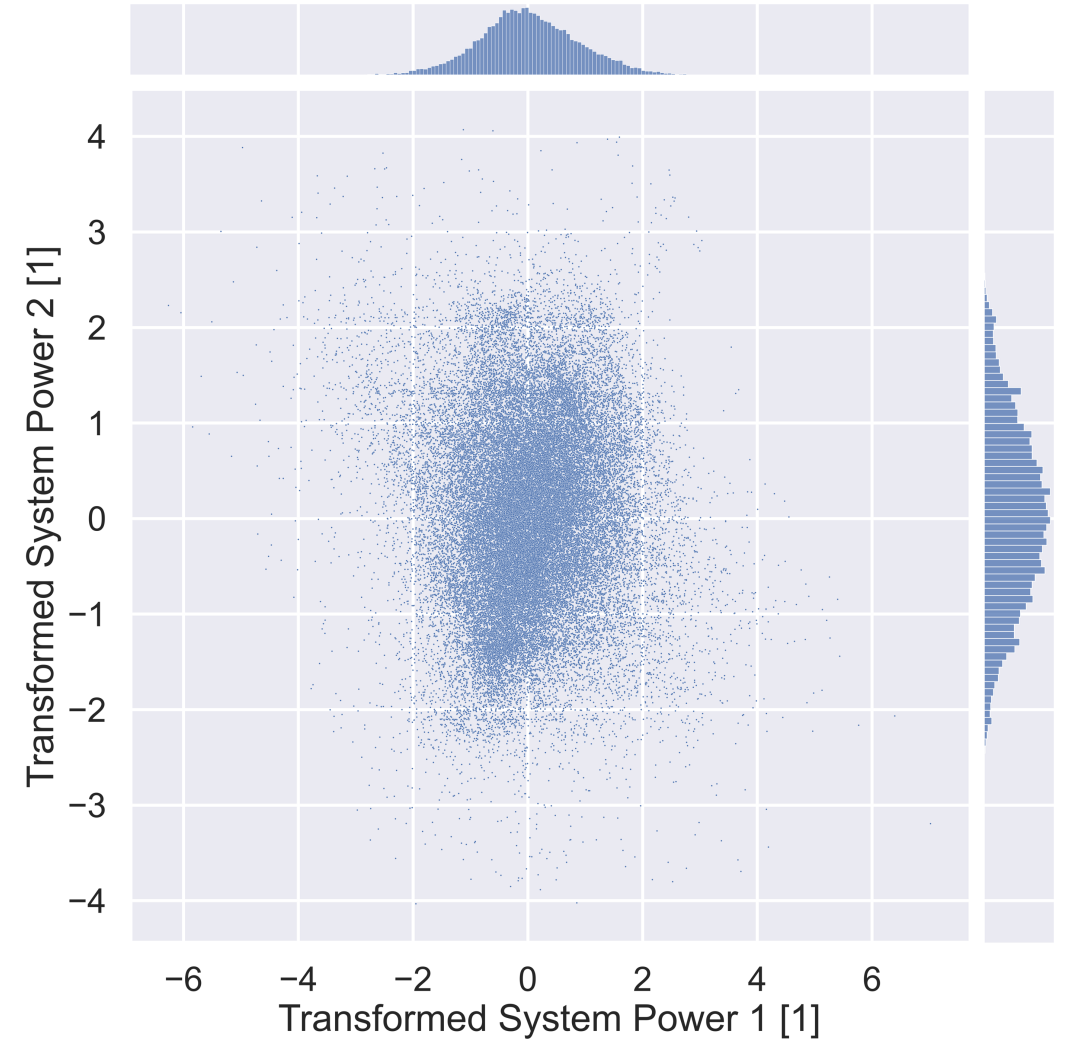
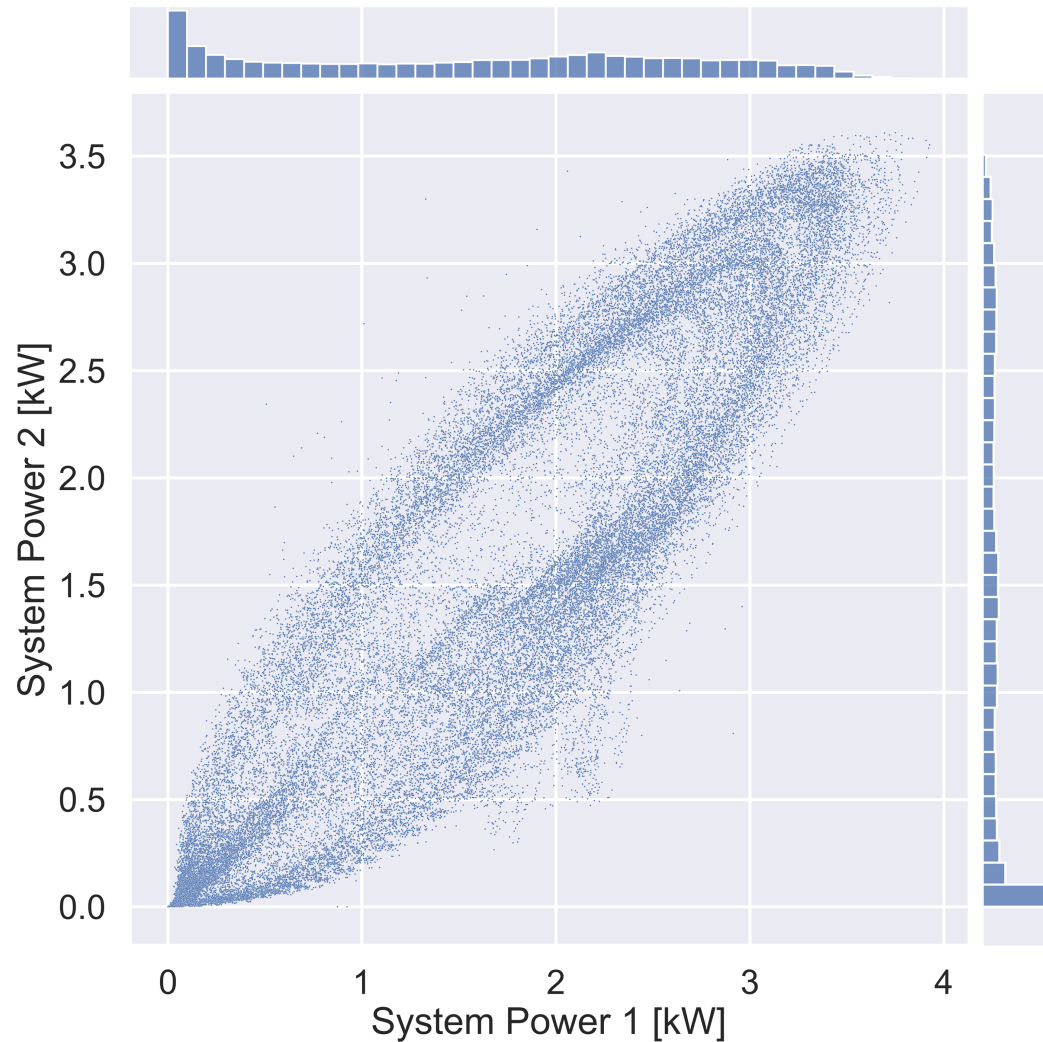
- Time series model based on Fourier expansions (and SD framework)
- Models (exactly) periodic functions as well as a class of *quasi*-periodic functions
 - Solar PV and temp: daily and yearly periods
 - Net load: daily, weekly, and yearly periods (human behavior)
- Truncated Fourier basis at each fundamental period, include cross terms
- Applications: anomaly detection, imputation, forecasting, joint probability modeling, and more



Forecasting: RI Net Load (Holdout Test Set)



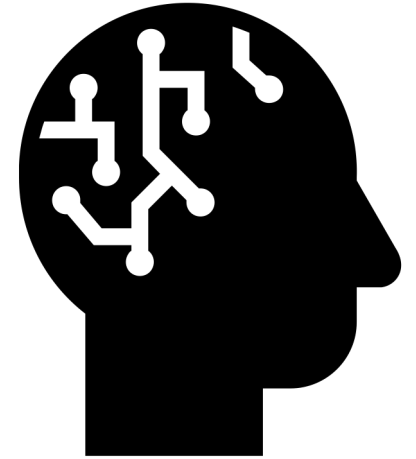
Joint Probability Modeling: Correlated PV Systems



Signal decomposition in relation to other fields

Closely related to various concepts from statistics, machine learning, remote sensing, *et al.*

- In “AI/ML-speak”: SD is a form of *unsupervised machine learning* on time-series data, in which *Bayesian priors* are specified by the analyst, encoding *domain expertise* and *analysis desiderata*.
- Analysts has a strong prior belief about what they want from a decomposition
 - rough sense of the number of components they are looking for
 - the general characteristics of those components
- Canonical example: ‘seasonal-trend decomposition’
 - first developed in 1920s, many off-the-shelf software packages available
 - can be approached as special case of an SD problem
- Contrast with principal component analysis (PCA)—data determines number of components, not analyst
- Also possible to validate an SD model, check it for consistency, and compare multiple SD models via *holdout validation* (long history, robust methods)



All training and inference can be done on standard laptop computers → no need for specialized server farms or expensive GPUs and easy to scale