A Framework for Signal Decomposition with Applications to Solar Energy Generation

Bennet Meyers

University Ph.D. Dissertation Defense

Department of Electrical Engineering

Advisor: Stephen Boyd

December 8, 2022

Outline

What I've worked on the last 4+ years

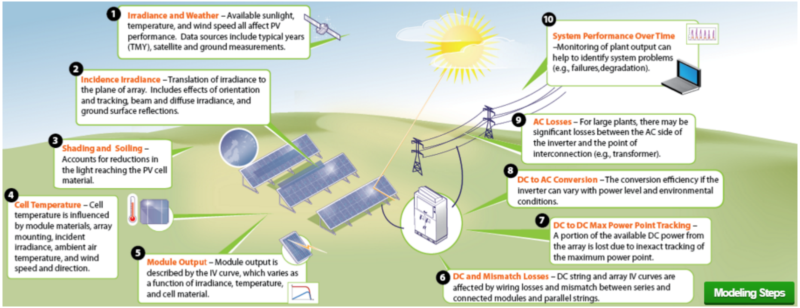
- research focused on two areas
  - optimization as a methodology for signal decomposition
  - developing useful, practical methods for solar power data science
- published 12 papers and 2 open-source software packages
- at SLAC, principle investigator on two U.S. DOE projects (SETO #34911 and #38529)

A selection of relevant papers

- B. Meyers, M. Tabone, and E. C. Kara. Statistical clear sky fitting algorithm. In *45th IEEE Photovoltaic Specialists Conference*, 2018
- 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*, 2019
- B. Meyers, E. Apostolaki-Iosifidou, and L. T. Schelhas. Solar data tools: Automatic solar data processing pipeline. In *47th IEEE Photovoltaic Specialists Conference*, 2020
- B. Meyers and D. J. F. Rodriguez. Estimation of shade losses in unlabeled PV data. In *49th IEEE Photovoltaic Specialists Conference*, 2022
- B. Meyers. Estimation of soiling losses in unlabeled PV data. In *49th IEEE Photovoltaics Specialists Conference*, 2022
- L. Volpatti, B. Meyers, and S. Boyd. Signal decomposition via quadratic-separable optimization. *[in progress]*, 2023
- B. Meyers and S. Boyd. Signal decomposition using masked proximal operators. *Foundations and Trends in Signal Processing*, 2023 (accepted)

- will (mostly) discuss last paper today

Motivation: Analysis of PV data



via Sandia PVPMC

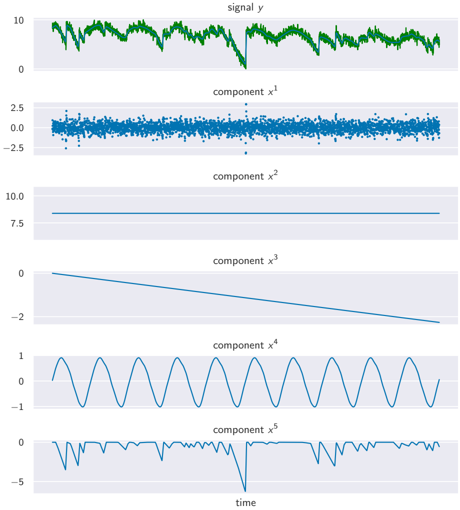
- standard approach uses photovoltaic (PV) output time series *plus*
  - location, mounting, orientation, ...
  - time series for irradiance, temperature, wind speed, ...
- can be very difficult (sometime impossible) to gather all this data
- we propose using only PV output time series, using signal decomposition

What is signal decomposition?

- given time series y
  (possibly with missing data)
- decompose as

$$y = x^1 + x^2 + \cdots + x^K$$

- components x^i have known characteristics
  - *e.g.*, smooth, periodic, nonnegative, nonincreasing, sparse, ...
- find $x^1, \ldots, x^K$ given y and characteristics
- often an underspecified problem
- raises question: how to choose one?



An age-old problem

- Babylonians use of harmonic analysis in astronomy ($\sim$1000 BCE)
- Fourier analysis in physics, astronomy (early 1800s)
- trend filtering in econometrics, earth science (1920s, formalized in 1990s)
- seasonal-trend decomposition (1920s, formalized as STL in 1990)
- sparse signal recovery in radio astronomy, JPEG standard, LASSO (1970s–1990s)
- convex demixing in geophysics (1970s)

- more recent techniques (2000–):
  - basis pursuit, dictionary learning, contextually supervised source separation, low-rank factorizations, ...

Our focus

- practical solution methods for carrying out decompositions
  - not theory and formal proofs
- specifying component characteristics via an extensible modeling language
  - not discovering components and their characteristics
- usable, open-source artifacts

```
# residual component
c1 = SumSquare(weight=1/len(y))
# constant component
c2 = NoSlope()
# linear degradation component
c3 = Aggregate([NoCurvature(), FirstValEqual(0)])
# seasonal baseline component
c4 = Aggregate([SumSquare(weight=5e0, diff=2), Periodic(365),
                AverageEqual(0, period=365)])
# soiling component
c5 = Aggregate([Inequality(vmax=0), SumAbs(weight=1e-5),
                SumQuantile(weight=1e-5, diff=1, tau=0.9),
                SumAbs(weight=5e-3, diff=2)])

problem = Problem(data=y, components=[c1, c2, c3, c4, c5])
```

Outline

Signal decomposition problem

$$\begin{array}{ll} \text{minimize} & \phi_1(x^1) + \cdots + \phi_K(x^K) \\ \text{subject to} & y_t = x_t^1 + \cdots + x_t^K, \quad t \in \mathcal{K} \end{array}$$

- data $y \in (\mathbf{R} \cup \{?\})^T$ with known entries $\mathcal{K} = \{t \mid y_t \in \mathbf{R}\}$
- variables are components $x^1, \ldots, x^K \in \mathbf{R}^T$
- components sum to signal for known entries
- component class costs $\phi_1, \ldots, \phi_K : \mathbf{R}^T \to \mathbf{R} \cup \{\infty\}$
  - $\phi_i(x)$ is the implausibility that $x^i = x$
  - infinite values encode constraints
- we choose decomposition to minimize total implausibility
- we refer to a solution as *an optimal signal decomposition*
- when class costs are convex, problem is convex

Imputing missing entries

$$y_t = x_t^1 + \cdots + x_t^K, \quad t \in \mathcal{K}$$

- decomposition gives a method for estimating missing data values, *i.e.*, *imputation*

$$\widehat{y}_t = x_t^1 + \cdots + x_t^K, \quad t \notin \mathcal{K}$$

  (recall $x^i \in \mathbf{R}^T$ have no missing values)
- provides a basis for a validation method (more on that later)

Simple class examples

mean-square small

$$\phi(x) = \frac{1}{T}\sum_{t=1}^{T}(x_t)^2$$

- encourages small x

mean-square smooth

$$\phi(x) = \frac{1}{T-1}\sum_{t=1}^{T-1}(x_{t+1}-x_t)^2$$

- encourages smooth x

Boolean

$$\phi(x) = \begin{cases} 0 & x_t \in \{0,1\} \\ \infty & \text{otherwise} \end{cases}$$

- requires values 0 or 1

(we'll see more complex examples later)

Statistical interpretation

- assuming $Z = \int \exp -\phi(x)\, dx < \infty$, define density

$$p(x) = \frac{1}{Z} \exp -\phi(x)$$

- $\phi(x)$ is negative log-likelihood of x (plus constant)
- suppose
  - all classes have a density
  - $x^1, \ldots, x^K$ are independent random variables with densities $p_1, \ldots, p_K$
- then SD is the *maximum a posteriori* (MAP) estimate of components

- moving on, we won't focus on statistical framing

Class parameters

- class costs have *parameters* $\theta \in \Theta$, expressed as $\phi(x;\theta)$
- examples
  - scale a fixed function, *i.e.*, $\phi(x;\theta) = \theta\ell(x)$, $\theta > 0$
    (typically use symbol λ here)
  - specify constant values or constraints, *e.g.*, $\phi(x;\theta) = \mathcal{I}(\theta_1 \leq x \leq \theta_2)$, $\theta_1 \leq \theta_2$
  - specify a basis, *i.e.*, $\phi(x,z;\theta) = \mathcal{I}(x = \theta z)$, $\theta \in \mathbf{R}^{T \times d}$
    (sometimes called a *dictionary*, note helper variable $z \in \mathbf{R}^d$)

- parameters are used to change or shape the decomposition
- common practice: find decomposition for several values of parameters, then select best (more on that later)

Example: Sum-absolute small with transform

$$\phi(x; \lambda) = \lambda \|Dx\|_1$$

- $D \in \mathbf{R}^{r \times T}$; λ is a weight parameter
- encourages sparsity of Dx

- $D = I$: encourages sparse x (Laplacian prior)
- D is 1st order difference matrix: encourages sparse differences in x, *i.e.*, piecewise constant x
- D is 2nd order difference matrix: encourages sparse second differences, *i.e.*, piecewise affine x

Example: Monotonic with regularization

$$\phi(x;\lambda) = \begin{cases} \lambda\ell(x) & x_t \leq x_{t+1} \text{ for } t = 1, \ldots, T-1 \\ \infty & \text{otherwise} \end{cases}$$

- $\ell(x) : \mathbf{R}^T \to \mathbf{R} \cup \{\infty\}$ is some loss; λ is a weight parameter
- x cannot decrease

- $\ell(x) = \sum_t (x_{t+1} - x_t)^2$: encourages smooth (and monotone) x
- $\ell(x) = \sum_t |x_{t+2} - 2x_{t+1} + x_t|$: encourages piecewise affine (and monotone) x

Hold out validation

- select some entries of y randomly as 'test'' or 'hold out' set $\mathcal{T} \subset \mathcal{K}$
- find decomposition using entries $\mathcal{K} \setminus \mathcal{T}$ and impute held out entries $\widehat{y}_t$, $t \in \mathcal{T}$
- evaluate mean-square test error $\frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} (y_t - \widehat{y}_t)^2$
- for more stable validation, can use cross validation

- validation guides choice of component losses and parameters, together with expert prior knowledge

Specifying signal decomposition in OSD

OSD: open source Python package for optimization-based signal decomposition

```
# residual component
c1 = SumSquare(weight=1/len(y))
# constant component
c2 = NoSlope()
# linear degradation component
c3 = Aggregate([NoCurvature(), FirstValEqual(0)])
# seasonal baseline component
c4 = Aggregate([SumSquare(weight=5e0, diff=2), Periodic(365),
               AverageEqual(0, period=365)])
# soiling component
c5 = Aggregate([Inequality(vmax=0), SumAbs(weight=1e-5),
               SumQuantile(weight=1e-5, diff=1, tau=0.9),
               SumAbs(weight=5e-3, diff=2)])

problem = Problem(data=y, components=[c1, c2, c3, c4, c5])
```

- classes are predefined Python objects
- construct models by combining objects
- can carry out (cross) validation

Outline

Introduction

Optimization based signal decomposition

Solution methods

Example: Estimated soiling losses in solar generators

Solving SD problems

- we'll give the flavor of three solution methods
- all provably give a (globally) optimal decomposition when problem is convex
- or reasonable approximate decomposition when problem is not convex
- all use the approach of breaking a problem into smaller problems
- we'll assume that first component class is mean-square small

$$\phi_1(x) = \frac{1}{T}\|x\|_2^2$$

so we interpret x^1 as a residual

Block coordinate descent algorithm

$$\begin{array}{ll} \text{minimize} & \frac{1}{T}\|x^1\|_2^2 + \phi_2(x^2) + \cdots + \phi_K(x^K) \\ \text{subject to} & y_t = x_t^1 + \cdots + x_t^K, \quad t \in \mathcal{K} \end{array}$$

SD-BCD algorithm:

- minimize over x^1 and x^2, holding others constant
- then over x^1 and x^3, then over x^1 and x^4, ...
- continue cyclically in Gauss-Seidel fashion

SD-BCD is a feasible descent method, *i.e.*, iterates are feasible and objective decreases

Masked proximal operator

- minimizing over x^1 and x^i is evaluating the *masked proximal operator* of ϕ_i

$$\mathbf{mprox}_{\phi_i}(v) = \operatorname*{argmin}_x \left(\phi_i(x) + \frac{\rho}{2} \sum_{t \in \mathcal{K}} (x_t - v_t)^2 \right)$$

  for SD-BCD, $\rho = 2/T$
- *cf.* proximal operator of ϕ^i

$$\mathbf{prox}_{\phi_i}(v) = \operatorname*{argmin}_x \left(\phi_i(x) + \frac{\rho}{2} \sum_{t} (x_t - v_t)^2 \right)$$

- with no unknown entries, the two are identical

Alternating direction method of multipliers algorithm

SD-ADMM:

Initialize. Set $u^0 = 0 \in \mathbf{R}^q$, and $(x^i)^0 \in \mathbf{R}^T$, $i = 1, \ldots, K$, as some initial estimates
for iteration $j = 0, 1, \ldots$

1. *Evaluate masked proximal operators of component classes in parallel.*

$$(x^i)^{j+1} = \mathbf{mprox}_{\phi_k}((x^i)^j - 2\mathcal{M}^* u^j), \quad i = 1, \ldots, K.$$

2. *Dual update.*

$$u^{j+1} = u^j + \frac{1}{K}\left(\sum_{i=1}^{K} \mathcal{M}(x^i)^{j+1} - \mathcal{M}y\right).$$

- $\mathcal{M}z$ gives entries z_i, $i \in \mathcal{K}$, in some known order
- $\mathcal{M}^* u$ puts entries of u into known entries, and zeros in unknown entries

SD-BCD versus SD-ADMM

- both access component losses only via masked proximal operators
- both converge to an optimal decomposition when SD problem is convex

- for convex problems
  - $\rho = 2/T$ works well for both SD-BCD and SD-ADMM
  - SD-BCD is a bit faster than SD-ADMM

- for nonconvex problems
  - $\rho = 1.4/T$ gives good results for SD-ADMM
  - SD-ADMM tends to give better decompositions than SD-BCD

But . . . masked proximal operators can be difficult to evaluate

- consider class of smooth x bounded by 1

$$\phi(x) = \begin{cases} \sum_{t=1}^{T-2}(x_t - 2x_{t+1} + x_{t+2})^2 & \|x\|_\infty \leq 1 \\ \infty & \text{otherwise} \end{cases}$$

- simple and useful class with no closed-form **mprox** (or **prox**)
- but, can be expressed as sum of two functions with simple proximal operators

$$\phi(x) = \sum_{t=1}^{T-2}(x_t - 2x_{t+1} + x_{t+2})^2 + \mathcal{I}(\|x\|_\infty \leq 1) = f(x) + g(x)$$

- this is an example of a *quadratic-separable* (QS) function
- inspiration for third solution method

Quadratic-separable optimization

quadratic-separable (QS) optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) + g(x) \\ \text{subject to} & Ax = b \end{array}$$

- f is convex quadratic
- g is separable, $g(x) = g_1(x_1) + \cdots + g_n(x_n)$
- linear equality constraints

- solution via ADMM only requires $\mathbf{prox}_g$, which is straightforward

SD via QS optimization

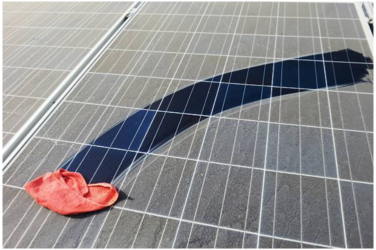
- when all ϕ_i are partial minimizations of QS functions

$$\phi(x) = \inf_z \{f(x,z) + g(x,z) \mid Ax + Bz = c\}$$

  SD problem is a QS problem
- complex class costs are broken into smaller parts
- separable functions with easy proximal operators are *atoms*
- these atoms are the basis of extensible modeling language

Outline

PV soiling



- build-up of material on the surface of PV modules over time
- can cause large yield reduction, as high as $-1\%/\text{day}$
- highly variable and difficult to predict
  - system geometry, local climate/weather, proximity to agriculture/industry,...
- important to industry (entire PVSC subarea, PVRW, *etc.*)

Soiling data

- we use NREL's soiling simulation model (Skomedal and Deceglie 2020)
- simulates daily PV energy including seasonal variation, long-term degradation, and soiling losses
- multiplicative loss model
- soiling component is known exactly, so we can validate SD

Data preprocessing

- take log of signal
- additive decomposition is multiplicative in original data
- min-max scaling to $[0, 10]$
- chosen to provide adequate dynamic range for soiling and degradation components

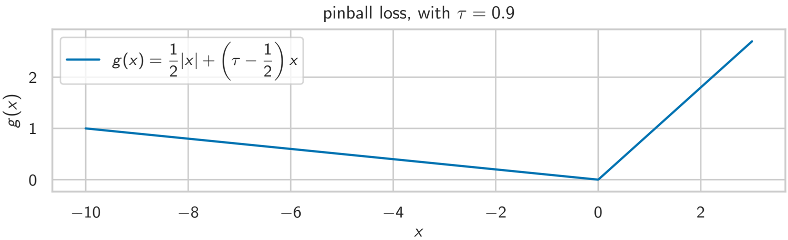
Component class losses

component	$\phi_i(x)$	description
1	$(1/T)\,\|x\|_2^2$	residual
2	$\mathcal{I}(D_1 x = 0)$	constant
3	$\mathcal{I}(D_2 x = 0)$	linear
4	$\lambda_3 \, \|D_2 x\|_2^2 + \mathcal{I}\,(x_t = x_{t+365})$	smooth & periodic
5	$\ell_{\mathrm{soil}}(x)$	soiling

Soiling class loss

$$\ell_{\text{soil}}(x) = \ell_1(x) + \ell_2(x) + \ell_3(x) + \ell_4(x)$$

	soiling loss term	description
ℓ_1	$\mathcal{I}(x \leq 0)$	non-positive
ℓ_2	$\lambda_{5a} \left\Vert x \right\Vert_1$	sparse
ℓ_3	$\lambda_{5b} \sum_{t=1}^{T-1} \left[\frac{1}{2}\left\vert (D_1 x)_t \right\vert + \frac{2}{5}(D_1 x)_t\right]$	asymmetric 1^{st} diff.
ℓ_4	$\lambda_{5c} \left\Vert D_2 x \right\Vert_1$	piecewise affine



Parameters

param.	value
λ_3	5
λ_{5a}	1×10^{-5}
λ_{5b}	5×10^{-3}
λ_{5c}	1×10^{-5}

- weights chosen to provide satisfactory results, not through holdout validation

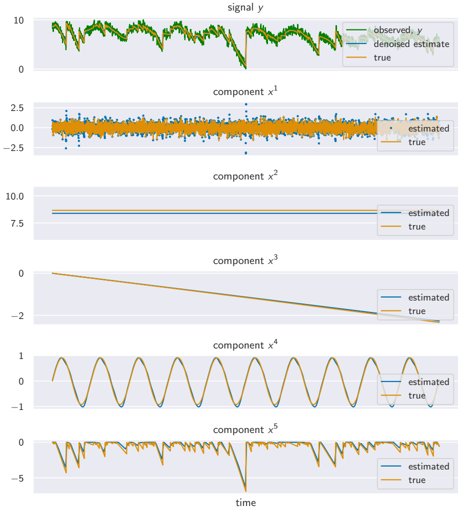
SD code specification

```
# residual component
c1 = SumSquare(weight=1/len(y))
# constant component
c2 = NoSlope()
# linear degradation component
c3 = Aggregate([NoCurvature(), FirstValEqual(0)])
# seasonal baseline component
c4 = Aggregate([SumSquare(weight=5e0, diff=2), Periodic(365),
                AverageEqual(0, period=365)])
# soiling component
c5 = Aggregate([Inequality(vmax=0), SumAbs(weight=1e-5),
                SumQuantile(weight=1e-5, diff=1, tau=0.9),
                SumAbs(weight=5e-3, diff=2)])

problem = Problem(data=y, components=[c1, c2, c3, c4, c5])
```

Results

- solved via QS algorithm
- solve time <10 seconds on 2016 MacBook Pro, no parallelization
  - $T = 3{,}653$
  - 40,543 variables
- components match true values very closely
- median daily soiling rate
  - estimated: $-0.071\%/\text{day}$
  - true: $-0.075\%/\text{day}$
- total annual degradation
  - estimated: $-0.48\%/\text{yr}$
  - true: $-0.50\%/\text{yr}$



Conclusions

- developed an SD framework
- user focus on component characteristics, not solution method
- we developed three custom, distributed solution methods that scale
- implemented a modeling language for SD
- embedded in Solar Data Tools
- already used my me and others on multiple PV data analysis problems

`https://github.com/cvxgrp/signal-decomposition`
`https://github.com/slacgismo/solar-data-tools`

Acknowledgments and thank yous

- Gina, Ezra, and my parents
- Ph.D. advisor Stephen Boyd and M.S. advisor Abbas El Gamal
- my coauthors, many listed on slide 4
- GISMo members past and present: Sila, Emre, Laura, Dave, Mayank, Karen, Pamela, Natalie, and many more
- PVInsight team: Mayank, Duncan, Mike, Kirsten, Kevin, Matt, Mehmet, Jimmy, and previously Elpiniki, Alejandro, Derin, and Jonathan
- CVXGRP and Marimo
- Tassos Golnas, Marie Mapes, and Inna Kozinsky at SETO

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 (SETO) Award Number 34368.

`https://bmeyers.github.io/about`

Appendix

Masking operator

- recall that equality constraint holds over known set
- introduce masking operator $\mathcal{M} : (\mathbf{R} \cup \{?\})^T \to \mathbf{R}^q$, where $q = |\mathcal{K}| \leq T$
- we also use its adjoint $\mathcal{M}^* : R^q \to R^T$, which sets unknown entries to zero
- note that while y can have missing values, $\mathcal{M}y$ does not
- for any $z \in (\mathbf{R} \cup \{?\})^T$, $\mathcal{M}^*\mathcal{M}z$ is z with unknown entries replaced with zeros

Stopping criterion

- last building block is the stopping criterion
- algorithms will be iterative, updating estimates until convergence
- let x^i_- be estimate of x^i before an iterate, and x^i_+ be the estimate after
- can express the optimality residual for unconstrained SD problem as

$$r = \left(\frac{1}{K-1} \sum_{i=2}^{K} \left\| \mathcal{M} \left(\rho(x^i_- - x^i_+) - \frac{2}{T} x^1 \right) \right\|_2^2 \right)^{1/2}$$

- if SD problem is convex and $r = 0$, components are optimal