Foundations and Trends® in Signal Processing

Signal Decomposition Using Masked Proximal Operators

Suggested Citation: Bennet E. Meyers and Stephen P. Boyd (2023), "Signal Decomposition Using Masked Proximal Operators", Foundations and Trends® in Signal Processing: Vol. 17, No. 1, pp 1–78. DOI: 10.1561/200000122.

Bennet E. Meyers
Stanford University
bennetm@stanford.edu

Stephen P. Boyd
Stanford University
boyd@stanford.edu

the essence of knowledge
Boston — Delft

Contents

Signal Decomposition Using Masked Proximal Operators

Bennet E. Meyers[1] and Stephen P. Boyd[2]

[1]*Stanford University, USA; bennetm@stanford.edu*
[2]*Stanford University, USA; boyd@stanford.edu*

ABSTRACT

We consider the well-studied problem of decomposing a vector time series signal into components with different characteristics, such as smooth, periodic, nonnegative, or sparse. We describe a simple and general framework in which the components are defined by loss functions (which include constraints), and the signal decomposition is carried out by minimizing the sum of losses of the components (subject to the constraints). When each loss function is the negative log-likelihood of a density for the signal component, this framework coincides with maximum a posteriori probability (MAP) estimation; but it also includes many other interesting cases. Summarizing and clarifying prior results, we give two distributed optimization methods for computing the decomposition, which find the optimal decomposition when the component class loss functions are convex, and are good heuristics when they are not. Both methods require only the masked proximal operator of each of the component loss functions, a generalization of the well-known proximal operator that handles missing entries in its argument. Both methods are distributed, *i.e.*, handle each component separately. We derive tractable methods for evaluating the

Bennet E. Meyers and Stephen P. Boyd (2023), "Signal Decomposition Using Masked Proximal Operators", Foundations and Trends® in Signal Processing: Vol. 17, No. 1, pp 1–78. DOI: 10.1561/200000122.

masked proximal operators of some loss functions that, to our knowledge, have not appeared in the literature.

1

Introduction

The decomposition of a time series signal into components is an age old problem, with many different approaches proposed, including traditional filtering and smoothing, seasonal-trend decomposition, Fourier and other decompositions, PCA and newer variants such as nonnegative matrix factorization, various statistical methods, and many heuristic methods. It is believed that ancient Babylonian mathematicians used harmonic analysis to understand astronomical observations as collections of 'periodic phenomena' [58].

As we will discuss in detail in §3, formulating the problem of decomposing a time series signal into components as an optimization problem has a long history. We introduce a simple framework that unifies many existing approaches, where components are described by their loss functions. Once the component class loss functions are chosen, we minimize the total loss subject to replicating the given signal with the components. We give a simple unified algorithm, based on variations of well-known algorithms, for carrying out this decomposition, which is guaranteed to find the globally optimal decomposition when the loss functions are all convex, and is a good heuristic when they are not. The method accesses the component loss functions only through a modified proximal operator interface, which takes into account that some data in the original signal may be missing. The method is distributed, in that each component class is handled separately, with the algorithm coordinating them.

Handling of missing data. The methods discussed in this monograph are designed to handle missing data in the original signal to be decomposed, a common situation in many practical settings. The signal components in the decomposition, however, do not have any missing data; by summing the components in the decomposition, we obtain a guess or estimate of the missing values in the original signal. This means that signal decomposition can be used as a sophisticated method for guessing or imputing or interpolating missing or unknown entries in a signal. This allows us to carry out a kind of validation or self-consistency check on a decomposition, by pretending that some known entries are missing, and comparing the imputed values to the known ones.

Expressivity and interpretability. The general framework described here includes many well-known problems as specific instances, and it enables the design of newer, more complex components classes than traditional simple ones such as a periodic signal, a trend, a smooth signal, and so on. For example we can define a signal component class that consists of periodic, smooth, and nonnegative signals, or piecewise constant signals that have no more than some specified number of jumps. The resulting decomposition is always interpretable, since we specify the component classes.

Outline. We describe the signal decomposition framework in §2, where we pose signal decomposition as an optimization problem, concluding with an illustrative simple example in §2.9. In §3 we cover related and previous work and methods. Two distributed methods for solving the signal decomposition problem, based on variations of well established algorithms, are described in §4. The next two sections concern loss functions for signal component classes: general attributes are described in §5 and some example classes in §6. The topic of how to fit component class losses given archetypal examples is discussed in §6.3. We conclude the monograph with examples using real data: Weekly CO2 measurements at Mauna Loa in §7.1, hourly traffic over a New York bridge in §7.2, and 1-minute power output for a group (fleet) of seven photo-voltaic (PV) installations in §7.3.

Software. Our monograph is accompanied by an open-source software implementation called `OSD`, short for 'Optimization(-based) Signal Decomposition', available at:

https://github.com/cvxgrp/signal-decomposition.

2

Signal Decomposition

2.1 Signal decomposition into components

Vector time series signal. Consider a vector time series or signal, possibly with missing entries, $y_1, \ldots, y_T \in (\mathbf{R} \cup \{?\})^p$. We denote the ith entry of y_t as $(y_t)_i = y_{t,i}$. The value ? denotes a missing entry in the signal; we 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 $y_1^T, \ldots, y_T^T$.

The mask operator. Let $q = |\mathcal{K}|$ be the total number of known entries in y, with $q \leq Tp$. We introduce the mask operator $\mathcal{M} : (\mathbf{R} \cup \{?\})^{T \times p} \to \mathbf{R}^q$, which simply lists the entries of its argument that are in $\mathcal{K}$ in a vector, in some known order. We will also use its adjoint $\mathcal{M}^*$, which takes a vector in $\mathbf{R}^q$ and puts them into a $T \times p$ matrix, in the correct order, with other entries zero. Note that while the original signal y can have missing entries, the vector $\mathcal{M}y$ does not. We also observe that for any $z \in \mathbf{R}^{T \times p}$, $\mathcal{M}^*\mathcal{M}z$ is z, with the entries in $\mathcal{U}$ replaced with zeros.

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

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

We refer to this constraint, that the sum of the components matches the given signal at its known values, as the consistency constraint, which can be expressed as

$$\mathcal{M}y = \mathcal{M}x^1 + \cdots + \mathcal{M}x^K. \tag{2.1}$$

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

$$\widehat{y}_{t,i} = x^1_{t,i} + \cdots + x^K_{t,i}, \quad (t,i) \in \mathcal{U}, \tag{2.2}$$

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

2.2 Component classes

The K components are characterized by functions $\phi_k : \mathbf{R}^{T \times p} \to \mathbf{R} \cup \{\infty\}$, $k = 1, \dots, K$. We interpret $\phi_k(x)$ as the loss of or implausibility that $x^k = x$. We will see later that in some cases we can interpret the classes statistically, with $\phi_k(x)$ the negative log-likelihood of x for signal class k. 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.

We will see many examples of component class losses later, but for now we mention a few simple examples.

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

$$\phi(x) = \frac{1}{Tp} \sum_{t,i} (x_{t,i})^2 = \frac{1}{Tp} \|x\|_F^2, \tag{2.3}$$

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 (2.3). 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

$$\phi(x) = \frac{1}{(T-1)p} \sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_F^2, \tag{2.4}$$

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

$$\phi(x) = \begin{cases} 0 & x_{t,i} \in \{0,1\} \text{ for all } t,i \\ \infty & \text{otherwise.} \end{cases} \tag{2.5}$$

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.

2.3 Signal decomposition problem

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

$$\begin{array}{ll} \text{minimize} & \phi_1(x^1) + \cdots + \phi_K(x^K) \\ \text{subject to} & \mathcal{M}y = \mathcal{M}x^1 + \cdots + \mathcal{M}x^K, \end{array} \tag{2.6}$$

with variables $x^1, \dots, x^K$. 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.

We observe that the entries of the mean-square small component x^1 with indices in $\mathcal{U}$ do not appear in the contraints, so their optimal value is zero, *i.e.*, $x^1 = \mathcal{M}^*\mathcal{M}x^1$. It follows that $\phi_1(x^1) = \frac{1}{Tp}\|\mathcal{M}x^1\|_2^2$. We can now eliminate x^1, and express the SD problem as the unconstrained problem

$$\text{minimize} \quad \tfrac{1}{Tp}\left\|\mathcal{M}y - \mathcal{M}x^2 - \cdots - \mathcal{M}x^K\right\|_2^2 + \phi_2(x^2) + \cdots + \phi_K(x^K), \tag{2.7}$$

with variables $x^2, \dots, x^K$. From a solution of this problem we can recover an optimal x^1 for (2.6) from the residual in the first term, as $x^1 = \mathcal{M}^*(\mathcal{M}y - \mathcal{M}x^2 - \cdots - \mathcal{M}x^K)$.

Solving the signal decomposition problem. If the class losses ϕ_k are all convex functions, the SD problem (2.6) is convex, and can be efficiently solved globally [21]. In other cases it can be very hard to find a globally optimal solution, and we settle for an approximate solution. In §4 we will describe two methods that solve the SD problem when it is convex (and has a solution), and approximately solve it when it is not. The first method is based on block coordinate descent (BCD) [8], [85], and the second is based on ADMM [20], an operator splitting method. Both methods handle each of the component classes separately, using the masked proximal operators of the loss functions (described in §4.1). This gives a very convenient software architecture, and makes it easy to modify or extend it to many component classes.

Existence and uniqueness of decomposition. With the assumption that the first component class is mean-square small, and all other component classes contain at least one signal with finite loss, the SD problem is always feasible. But it need not have a solution, or when it does, a unique solution. For example, consider $K = 2$ with a mean-square small component and a Boolean component. If the (t, i) entry in y is unknown, then $x^2_{t,i}$ can be either 0 or 1, without affecting feasibility or the objective. The uniqueness of specific instances of the SD problem (particularly when $K = 2$) has been studied extensively [31], [55]. (See §3 for a longer discussion.)

2.4 Statistical interpretation

We can give the losses a simple statistical interpretation in some cases, which conversely can be used to suggest class losses. Suppose that ϕ is continuous on its domain, with

$$Z = \int \exp -\phi(x)\, dx < \infty.$$

(The integration is with respect to Lebesgue measure.) We associate with this component class the density

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

Thus, $\phi(x)$ is a constant plus the negative log-likelihood of x under this density, a standard statistical measure of implausibility. Convex loss functions correspond to log-concave densities.

As an example, with the mean-square loss $\phi(x) = \frac{1}{2Tp}\|x\|_F^2$ (note the additional factor of two in the denominator), the associated density is Gaussian, with the entries of x IID $\mathcal{N}(0, 1)$. As another example, the mean-square smooth component class with loss (2.4) has $Z = \infty$, so we cannot associate it with a density.

When all component classes have $Z < \infty$, we can interpret the SD problem statistically. Suppose $x^1, \ldots, x^K$ are independent random variables with densities $p_1, \ldots, p_K$. Then the SD objective is a constant plus the negative log-likelihood of the decomposition with $x^1, \ldots, x^K$, and the SD decomposition is the maximum a posteriori probability (MAP) decomposition of the observed signal y.

2.5 Optimality and stationarity conditions

Here we give optimality or stationarity conditions for the SD problem for some special but common cases. In all cases, the conditions include primal feasibility (2.1), *i.e.*, consistency, and a second condition, dual feasibility, which has a form that depends on the properties of the losses.

Differentiable losses. We first suppose that the losses are differentiable. The dual feasibility condition is that there exists a Lagrange multiplier $\nu \in \mathbf{R}^q$ for which

$$\nabla \phi_k(x^k) = \mathcal{M}^* \nu, \quad k = 1, \ldots, K,$$

where $\nu \in \mathbf{R}^q$ is a dual variable or Lagrange multiplier associated with the consistency constraint (2.1). In words: the gradients of the losses all agree, and are zero in the unknown entries. If all losses are convex, this condition together with primal feasibility are the necessary and sufficient optimality conditions for the SD problem. If the losses are not all convex, then this condition together with primal feasibility are stationarity conditions; they hold for any optimal decomposition, but there can be non-optimal points that also satisfy them.

Since $\phi_1(x) = \frac{1}{Tp}\|x\|_F^2$, we have $\nabla \phi^1(x) = (2/Tp)x$. The dual conditions can then be written as

$$x^1 = \mathcal{M}^*\mathcal{M}x^1, \qquad \nabla \phi_k(x^k) = \frac{2}{Tp}x^1, \quad k = 2, \ldots, K, \tag{2.8}$$

i.e., the gradients of the component class losses all equal the mean-square residual, scaled by $2/(Tp)$ in the known entries, and are zero in the unknown entries. These are also the conditions under which the gradients of the objective in the unconstrained SD problem formulation (2.7) with respect to $x^2, \ldots, x^K$ are all zero.

Convex losses. If the losses are convex but not differentiable, we replace the gradients in (2.8) with subgradients, to obtain

$$x^1 = \mathcal{M}^*\mathcal{M}x^1, \quad g^k = \frac{2}{Tp}x^1, \quad g^k \in \partial \phi_k(x^k), \quad k = 2, \ldots, K, \tag{2.9}$$

where $\partial \phi_k(x^k)$ is the subdifferential of ϕ_k at x^k. This condition, together with primal feasibility, are optimality conditions for the SD problem.

Other cases. When the losses are neither convex nor differentiable, the stationarity conditions can be very complex, with the gradients in (2.8) or subgradients in (2.9) substituted with some appropriate generalized gradients.

2.6 Signal class parameters

The component class losses ϕ_k can also have parameters associated with them. When we need to refer to the parameters, we write $\phi_k(x^k)$ as $\phi_k(x^k; \theta_k)$, where $\theta_k \in \Theta_k$, the set of allowable parameters. These parameters are fixed whenever we solve the SD problem, but it is common to solve the SD problem for several values of the parameters, and choose one that works well (*e.g.*, using a validation method described later). The role of the parameters θ_k will be made clear when we look at examples. For now, though, we mention a few common examples.

Weight or scaling parameters. It is very common for a parameter to scale a fixed function, *i.e.*, $\phi(x; \theta) = \theta \ell(x)$, $\theta \in \Theta = \mathbf{R}_{++}$, the set of positive numbers. (Of course we can have additional parameters as well.) In this case we interpret the parameters as weights that scale the relative implausibility of the component classes. We will use the more traditional symbol λ to denote scale factors in loss functions, with the understanding that they are part of the parameter θ.

Value and constraint parameters. Parameters are often used to specify constant values that appear in the loss function. For example we can generalize the Boolean loss function, which constrains the entries of x to take on values in $\{0, 1\}$, to one where the entries of x take on values in a finite set $\{\theta_1, \ldots, \theta_M\}$, where $\theta_i \in \mathbf{R}$, *i.e.*,

$$\phi(x) = \begin{cases} 0 & x_{t,i} \in \{\theta_1, \ldots, \theta_M\} \text{ for all } t, i \\ \infty & \text{otherwise.} \end{cases} \tag{2.10}$$

In this case, the parameters give the values that the entries of x are allowed to take on. As another example, consider a loss function that constrains the entries of x to lie in the interval $[\theta_1, \theta_2]$ (with $\theta_1 \leq \theta_2$). Here the parameters set the lower and upper limits on the entries of x.

Basis. Another common use of parameters is to specify a basis for the component, as in

$$\phi(x) = \begin{cases} 0 & x = \theta z \text{ for some } z \in \mathbf{R}^{d\times p} \\ \infty & \text{otherwise,} \end{cases} \tag{2.11}$$

where $\theta \in \mathbf{R}^{T\times d}$, and $z \in \mathbf{R}^{d\times p}$. This component class requires each column of x, *i.e.*, the scalar time series associated with an entry of x, to be a linear combination of the basis (scalar) signals given by the columns of θ (sometimes referred to as a dictionary). The entries of z give the coefficients of the linear combinations; for example, the first column of x is $z_{11}\theta_1 + \cdots + z_{1d}\theta_d$, where θ_i is the ith column of θ, *i.e.*, the ith basis signal.

2.7 Model selection

We refer to a particular choice of component classes and their parameter values as an SD model. A natural question is: How should we choose the SD model? In some contexts such as prediction in machine learning the analogous question of what prediction model we should use, and what parameters we should select, has a straightforward answer: We should use the model that has the best out-of-sample prediction performance. (In some cases there are secondary objectives such as model simplicity or interpretability.) At the other extreme we have unsupervised machine learning methods, such as clustering methods, where it is harder to identify a measure of model performance, and therefore harder to find a method for choosing one model over another. In such cases the model and parameter values are chosen so that the results correspond to what the user expects or wants to see. If the model can handle missing data, it can also be checked for internal consistency by checking how it imputes values that we actually know, but pretend while building the model are unknown. Signal decomposition lies closer to the unsupervised learning setting.

The methods described in this monograph are typically applied in situations where the analyst has a strong prior belief about what they want from a decomposition, often drawn from domain expertise. The analyst has a rough sense of the number of components they are looking for and the general characteristics of those components, which inform the selection of K, ϕ_k, and θ_k.

The classic example of this style of analysis is seasonal-trend decomposition (see §3 and §7.1), in which a scalar signal ($p = 1$) is decomposed into $K = 3$ components: seasonal, trend, and residual. (We will see later that this can be approached as an SD problem.) Here we use the strong prior that the seasonal component should vary smoothly over the year, and the trend component must change slowly. So $K = 3$, and the specific forms of the component losses, are not arbitrary; each has a specific meaning. In this case the weights or parameters in the loss functions are chosen to give a plausible or useful decomposition.

This can be contrasted and compared with PCA, where we need to determine the number of principal components K to use. Aside from the general idea that smaller K is to be preferred over larger K, there is no particular meaning to prefer $K = 3$. In PCA we let the data determine K, typically by finding the smallest K for which the model is at least reasonably self-consistent.

For SD, the specific components, and the form of the loss functions, are specified by the analyst. The quality of the decomposition is judged using the analyst's domain expertise and intuition. It is also possible to validate an SD model, or at least, check its consistency. We describe this now.

Model validation. We can validate, or at least check consistency of, a choice of the component classes and their parameter values. To do this, we select (typically randomly) some entries of y that are known, denoted $\mathcal{T} \subset \mathcal{K}$ (for 'test'), and replace them with the value ?. A typical choice of the number of test entries is a fraction of the known entries, such as 20%. We then carry out the decomposition by solving the SD problem, using the entries $\mathcal{K} \setminus \mathcal{T}$ of y. This decomposition gives us estimates or guesses of the entries of y in $\mathcal{T}$, given by (2.2). Finally, we check these estimates against the true values of y, for example by evaluating the mean-square test error

$$\frac{1}{|\mathcal{T}|p} \sum_{(t,i)\in\mathcal{T}} (y_{t,i} - \widehat{y}_{t,i})^2.$$

A more stable estimate of test error can be found by evaluating the mean-square test error for multiple test sets $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(M)}$, each with the same number of entries, and averaging these to obtain a final mean-square error.

It is reasonable to prefer a model (*i.e.*, choice of component classes and their parameters) that results in small test error, compared to another model with higher test error. The out-of-sample validation method described above can be used to guide the choice of the component classes and parameters that define an SD model.

Validating with non-unique decompositions. We note that the basic validation method fails when the SD problem has multiple solutions, or more precisely, when multiple optimal signal decompositions correspond to different values of $\widehat{y}_{t,i}$ for $(t,i) \in \mathcal{T}$. One simple work-around is to regard the multiple solutions as each providing an estimate of the missing entry, and to evaluate the test loss using the best of these estimates. For example, suppose the second component class is Boolean, so $(x^2)_{t,i}$ can have the value 0 or 1 for $(t,i) \in \mathcal{T}$. We judge the error using

$$\frac{1}{|\mathcal{T}|p} \sum_{(t,i)\in\mathcal{T}} \min_{x^2_{t,i}\in\{0,1\}} (y_{t,i} - \widehat{y}_{t,i})^2.$$

Parameter search. As is standard in machine learning and data fitting, it is common to carry out multiple decompositions with the same loss functions but different parameters, and validate each of these choices on one or more test sets, as described above. We then choose as the final parameter values ones corresponding to the lowest achieved test error. As in machine learning and data fitting, the final decomposition is then fit with all known data, using the parameter values found in the parameter search.

2.8 Data pre-processing

As in other data processing problems, pre-processing the raw data is often useful, leading to better results or interpretability.

Standarization. The most basic pre-processing is to standardize the entries of y, with a scale and offset for each component that results in the average value being around zero and the standard deviation around one. In some cases, for example when the entries of y are all measured in the same physical units, it can be more appropriate to use the same scaling for all components of y.

Log transform. If the data are all positive and vary over a large range of values, a log transform of the raw data can be appropriate. Roughly speaking, this means that we care about relative or fractional deviations, as opposed to absolute errors in the raw data, *e.g.*, we consider the values 10 and 11 to be as close as the values 1000 and 1100. With a log transform, the signal decomposition has an interpretation as a multiplicative decomposition (in the raw data), as opposed to an additive decomposition. If we denote the raw data as $\tilde{y}$ and the transformed data as $y = \log \tilde{y}$ (entrywise), and the decomposition is $y \overset{\mathcal{K}}{=} x^1 + \cdots + x^K$, in terms of the raw data we have

$$\tilde{y}_{t,i} = \tilde{x}^1_{t,i} \cdots \tilde{x}^K_{t,i}, \quad (t,i) \in \mathcal{K},$$

where $\tilde{x}^i = \exp x^i$ (entrywise), $i = 1, \ldots, K$. The signals $\tilde{x}^i$ can be though of as multiplicative factors.

2.9 Simple example

In this section we give a simple synthetic example to illustrate the idea.

Signal decomposition model. We construct an SD problem with $p = 1$ (*i.e.*, a scalar signal), $T = 500$, and $K = 3$ component classes: mean-square small, mean-square second-order smooth, and a scaled Boolean. For mean-square small we use loss function (2.3), and for mean-square second order smooth we use the loss

$$\phi_2(x) = \frac{\theta_1}{(T-2)p} \sum_{t=2}^{T-1} (x_{t+1} - 2x_t + x_{t-1})^2, \tag{2.12}$$

where θ_1 is a positive weight parameter. For the Boolean component class, we require that all entries of x are in $\{0, \theta_2\}$, where θ_2 is another positive parameter. Our SD problem contains two signal class parameters, θ_1 and θ_2. Since ϕ_3 is not convex, the SD problem is not convex. (Nevertheless the methods we describe below do a good job at approximately solving it.)

Data generation. We generate a signal y of length $T = 500$ as a sum of three 'true' signal components, one that is Gaussian noise, one that is smooth, and one that is Boolean, *i.e.*, takes on only two values. The first signal, denoted $\tilde{x}^1 \in \mathbf{R}^T$, has IID entries $\mathcal{N}(0, 0.1^2)$. The second true component is the quasiperiodic signal with three frequencies

$$\tilde{x}^2_t = \sum_{j=1}^{3} a_j \cos(\omega_j t + \delta_j), \quad t = 1, \ldots, T,$$

where $a_j > 0$ are the amplitudes, $\omega_j > 0$ are the frequencies, and $\delta_j \in [0, 2\pi]$ are the phases, all chosen randomly. The last true component signal $\tilde{x}^3$ has the form

$$\tilde{x}^3_t = \begin{cases} \tilde{\theta}_2 & \sum_{j=1}^{3} a'_j \cos(\omega'_j t + \delta'_j) \geq 0 \\ 0 & \sum_{j=1}^{3} a'_j \cos(\omega'_j t + \delta'_j) < 0, \end{cases}$$

for $t = 1, \ldots, T$, where a'_j, ω'_j, and δ' are a different set of amplitudes, frequencies, and phases, also chosen randomly. We construct the signal as

$$y = \tilde{x}^1 + \tilde{x}^2 + \tilde{x}^3,$$

with $\tilde{\theta}_2$ (the 'true' value of θ_2) chosen randomly. The signal y and the three true components $\tilde{x}^i$ are shown in Figure 2.1. The data in this example have no missing entries.

Parameter search. We use the method described below to approximately solve the SD problem for a grid of 21 values of θ_1, logarithmically spaced between 10^{-1} and 10^6, and 21 values of θ_2, linearly spaced between 0.1 and 2.0, for a total of 441 different values of the parameters. For each of these, we evaluate the test error using 10 random selections of the test set as described above. Thus all together we solved 4410 instances of the SD problem, which took about 13 minutes on a 2016 MacBook Pro. Each SD problem took about 0.17 seconds to solve. (We solved the problems sequentially, but the computation is embarrassingly parallel and could have been carried out faster using more processors.)

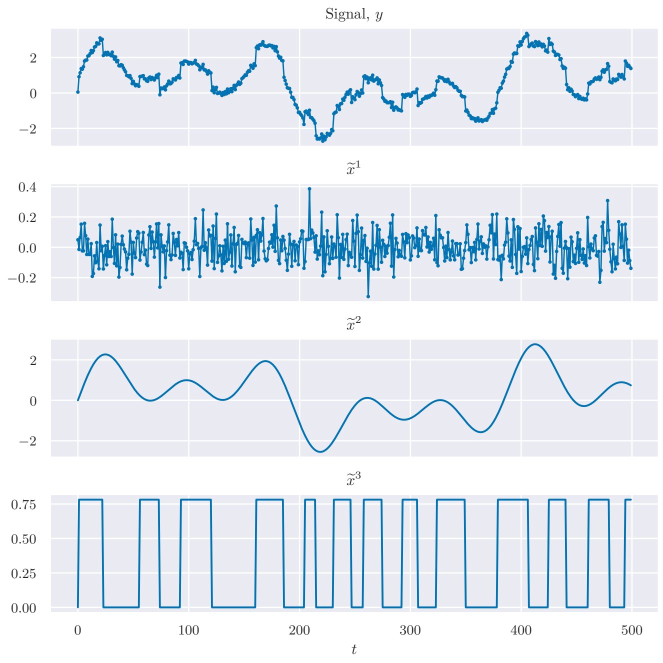

Figure 2.1: Synthetic data for simple signal decomposition example.

The mean-square test error for the parameter grid search is shown as a heat map in Figure 2.2. We use the final values $\theta_1 = 320$, $\theta_2 = 0.765$, which achieved the smallest mean-square test error. Having chosen θ_1 and θ_2, we approximately solve the SD problem one final time, using all the data.

There is some discrepancy between the value we find $\theta_2 = 0.765$ and the true value used to generate the data, $\widetilde{\theta}_2 = 0.7816$, due to the discreteness of the grid search. (In a real application, we might do a secondary, refined grid search of values near the best ones found in this crude grid search.)

Final decomposition. The final decomposition is shown in Figure 2.3. Evidently the decomposition is quite good. The Boolean component is exactly reconstructed, aside from the slight discrepancy in its amplitude. The smooth component is also well reconstructed, with an RMS (root mean-square) error about 0.04.

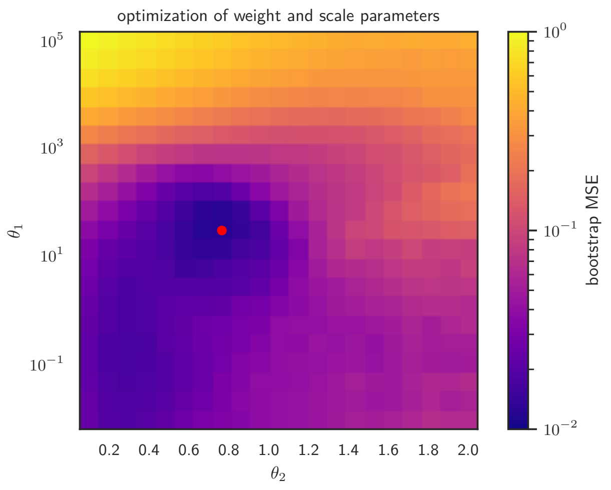

Figure 2.2: Validation mean-square test error as a function of the parameters θ_1 and θ_2.

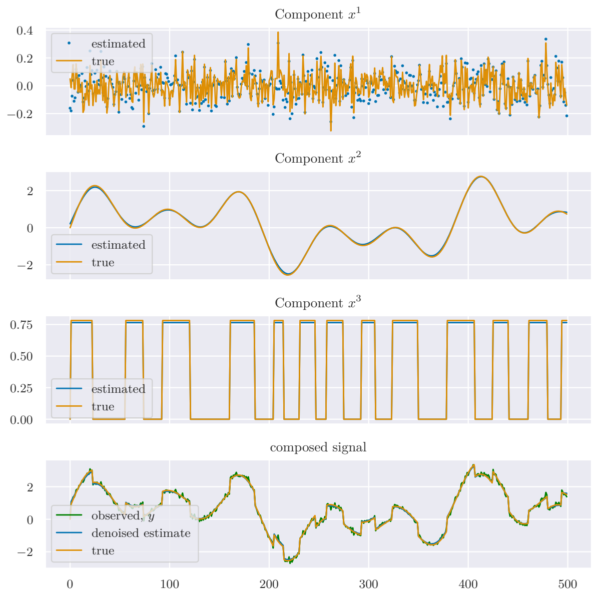

Figure 2.3: Signal decomposition for simple example. The top three plots show the true component and the esimated component. The bottom plot shows the original signal y and $x^2 + x^3$, *i.e.*, the decomposition without the residual component x^1.

3

Background and Related Work

Here we discuss a wide variety of methods that relate to the topic of signal decomposition, some of which are quite old. Many methods described below do not explicitly form an optimization problem, and when they do, it need not conform to the signal decomposition framework described in this monograph. Other methods involve minimizing a sum of loss functions for signal component classes subject to their sum matching an observed or given signal at known entries, exactly as in the proposed framework. In these cases, the discussed methods are often specific instances of the SD problem (2.6), and these connections will be noted where appropriate. The SD formulation can be thought of as a generalization of the specific approaches to signal decomposition discussed in this section.

Regression. Least-squares linear regression is a particular instance of the SD problem, with two component classes, a mean-square small component, and a component defined by a basis (2.11), with the basis components the regressors or features. This SD problem instance admits a well-known, closed form solution [22, Ch. 12]. The idea of solving an over-determined system of linear equations by minimizing the sum of the squares of the errors was proposed independently by the mathematicians Carl Friedrich Gauss and Adrien-Marie Legendre around the beginning of the 19th century. Statistical justifications for this fitting procedure were subsequently provided by Gauss, Laplace, Cauchy, and Thiele, among others [32].

Robust regression. Robust regression covers a variety of techniques to reduce model variance in the presence of data 'outliers,' which is a term without a precise definition but can be thought of as data points that are not well explained by a linear regression model. Common methods include Huber regression [42], [43], Theil-Sen estimation [68], [76], and RANSAC [33], which are included in the popular Python package, `scikit-learn` [62]. In the SD framework, the residual component class used in linear regression is substituted with an alternative penalty function that is less sensitive to outliers. The penalty function formulation is discussed in detail in [21, §6.1 and §6.4]. Interestingly, the idea of minimizing the sum of absolute errors in an over-determined system of equations actually predates the development of least-squares minimization, having been proposed in the mid-18th century by Roger Joseph Boscovich [33]. In the SD framework, robust regression is modeled using two residual classes, one the standard mean-square small, and the other a loss function that grows slowly for large values, like the average absolute loss

$$\phi(x) = \frac{1}{Tp}\|x\|_1 = \frac{1}{Tp}\sum_{t=1}^{T}\sum_{i=1}^{p}|x_{t,i}|. \tag{3.1}$$

(This same loss is used as a convex heuristic for a sparse signal, *i.e.*, one with many entries zero.)

Regularized regression. Regularized regression, also known as penalized regression or shrinkage methods, is a family of estimators that introduce an additional penalty term on coefficients of a linear regression problem. Well known examples include ridge regression [41], [63], [79], lasso regression [78], and elastic-net regression [90]. An overview of different regularizer functions for regression is given in [21, §6.3], and a review of other regressor selection methods is given in [39, Ch. 3–4]. In the SD framework, regularized regression is modeled by extending the basis class (2.11) with an additional loss term on the internal variable, as in

$$\phi(x) = \begin{cases} \ell(z) & x = \theta z \text{ for some } z \in \mathbf{R}^{d \times p} \\ \infty & \text{otherwise.} \end{cases}$$

Isotonic regression. In isotonic (or monotonic) regression we fit a given signal with a non-decreasing (or non-increasing) signal [10], [86]. This is a particular instance of the SD problem, with $p = 1$ and $K = 2$ component classes: a sum-of-squares small residual and a monotone component, which has a loss function that is zero if its argument is non-decreasing and infinite otherwise. As efficient algorithm, with complexity linear in T, is included in `scikit-learn` [62]. A detailed discussion of a linear time algorithm and the connection to projection operators is given in [38].

Trend filtering. Trend filtering, also called signal smoothing, is the process of estimating a slowly varying trend from a scalar time series that includes rapidly varying noise. In many cases this is also a special case of SD, with a mean-square residual component and a component that is slowly varying, for example, with a mean-square second difference loss function. Trend filtering has been employed in a wide variety of applications and settings, including astrophysics [80], geophysics [6], [15], [16], social sciences [51], biology [52], medicine [37], image processing [77], macroeconomics [40], [69], and financial time series analysis [82, §11]. Many specific trend filtering methods have been proposed, including moving-average filtering [60] and Hodrick-Prescott (HP) filtering [40], [50] being two of the most well known. More recently, Kim et al. have proposed ℓ_1 trend filtering [47], which uses as component loss function the ℓ_1 norm of the second difference, which tends to result in piecewise affine signals (see §6.2 for an example). A Bayesian interpretation of trend filtering and signal denoising is presented in [17], [28], [77], [80].

Seasonal-trend decomposition. Seasonal-trend decomposition was originally motivated by the analysis of economic data which tend to have strong seasonality; this method is arguably what most people think of when they hear the term "time series decomposition," having first been proposed in the 1920s as a natural extension of moving average smoothing [3]. Seasonal-trend decomposition is the only one presented in the chapter on time series decomposition in Hyndman and Athanasopoulos [44, §6]. A popular algorithm that implements a specific method for seasonal-trend decomposition is STL [27], with packages available for Python, R, and Matlab [71]–[73].

STL can be considered a specific case of the SD problem, with a scalar signal y and $K = 3$ component classes, *i.e.*, seasonal, trend, and residual. However, STL does not formulate the method as an optimization problem and uses an iterative heuristic to form the estimates of the components.

Modern extensions of the seasonal-trend decomposition problem have been introduced. In 2019, researchers from the remote sensing community proposed an extension that introduces a new 'abrupt change' component, which is modeled as a piecewise linear component with a small number of breakpoints [89]. Somewhat unique to this work is a focus on calculating uncertainty in the components and particularly the breakpoint locations.

Traditional frequency domain filtering. Traditional EE-style filtering (*e.g.*, [59]) can be interpreted as a form of SD. For example, low pass filtering decomposes a signal into a smooth component (the filter output) and a small, rapidly varying component (the residual, or difference of the original signal and the low pass signal). This can often be represented as SD with two components, a residual and a smooth or low-pass component with appropriate time-invariant quadratic loss function. A traditional filter bank can be interpreted as giving a decomposition of a signal into multiple components, each one corresponding to a different region (or sub-band) in the spectrum of the signal.

Sparse signal recovery. Sparse signal recovery is concerned with finding sparse representations of signals with respect to some known (typically over-complete) basis. The use of (convex) optimization to solve sparse signal recovery problems has a long history with many proposed approaches, and there are some very nice overviews available in [54], [81], [84]. These methods have historically been applied to the problem of data compression, such as the JPEG and JPEG2000 standards [7], [53]. These methods are all related to the regularized linear inverse problem [25],

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = y, \end{array} \tag{3.2}$$

where the matrix A and the vector y are problem data, and f is some 'complexity measure' that encourages sparseness. A common variant is to relax the equality constraint,

$$\text{minimize} \quad \|Ax - y\|_2^2 + \lambda f(x). \tag{3.3}$$

When $f(x) = \|x\|_1$, (3.2) is known as basis pursuit or compressed sensing, and (3.3) is the lasso, which we encountered in the previous paragraph. The geometry of these and related problems, specifically in the case where $f(x) = \|x\|_1$, has been extensively analyzed to determine when sparse signals are recoverable in [2]. The matrix A generally represents the data generation process, either derived from known measurements or, in the case of dictionary methods, derived from pre-defined, parameterized waveforms, like sinusoids or wavelets. With dictionary learning methods the matrix A is fit to the data as well [81]. When A is introduced as a decision variable, problems (3.2) and (3.3) are no longer convex, but there exist well established methods exists for approximately solving problems of this form [83].

Matrix completion. In the basic formulation of this problem, we seek a low rank matrix X which matches a known matrix M at a set of known indices [24]. A closely related problem is (robust) principle component pursuit, in which an observed matrix is decomposed into a low-rank component and a sparse component [23], [84].

Convex demixing. Convex demixing has a long history [55, §7.1], beginning in the geophysics community in the 1970s [26], [75]. It refers to the task of identifying two (or sometime more) 'structured signals,' given only the sum of the two signals and information about their structures [54], [55]. The standard formulation for convex demixing is

$$\begin{array}{ll} \text{minimize} & f(x) + \lambda g(z) \\ \text{subject to} & x + z = y, \end{array} \tag{3.4}$$

where x and z are the decision variables, y is the observed signal, and λ is a regularization parameter. This is evidently a two-class, convex SD problem. In this formulation, the focus tends to be on demixing signals that are sparse in various senses. A classic example is the 'spikes and sines problem', which shows up in a variety of applications including astronomy, image inpainting, and speech enhancement in signal processing [31], [70]. More generally, these types of problems include demixing two signals that are sparse in mutually incoherent bases, decoding spread-spectrum transmissions in the presence of impulsive (*i.e.*, sparse) errors, and removing sparse corruptions from a low-rank matrix. Problem (3.4) has been deeply studied in many contexts, and much of the existing work has focused on finding solution methods and analyzing recovery bounds (*i.e.*, uniqueness) when f and g are various sparsity-inducing matrix norms [5], [25]. A three-operator extension of (3.4)—where one operator is a smooth, nonconvex function and the other two operators are convex functions—is studied in [88]. These are instances of the signal decomposition problem.

Contextually supervised source separation (CSSS). This is an optimization-based framework for solving signal decomposition problems, in which the signal components are assumed to be roughly correlated with known basis vectors [87], and is very similar in many ways to the method presented in this monograph. CSSS is extensible, allowing for different loss terms on the linear representations, component estimates, and linear fit coefficients. The SD formulation proposed in this monograph is a further generalization of contextually supervised source separation, and the proposed solution method in §4 solves all instances of contextually supervised source separation as a subset of all SD problems.

Infimal convolution. The infimal convolution of functions $f_i : \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, K$ denoted $f_1 \Box \cdots \Box f_K$, is defined as

$$(f_1 \Box \cdots \Box f_K)(v) = \inf \left\{ f_1(x^1) + \cdots + f_K(x^k) \mid v = x^1 + \cdots + x^K \right\}$$

as described (for convex functions) in [66, §16] and [61, §3.1]. The case of nonconvex functions was considered in [64]. We see that the SD problem, with no missing data, is the problem of evaluating the infimal convolution of the component loss functions, on the given signal y.

Proximal operator. The proximal operator of a function f arises often in optimization, and is the basis of the solution methods described below. The details are given below, but we note there that evaluating a proximal operator of the function f is an SD problem (again, with no missing data) with a mean-square loss and the loss f.

Our contribution. We present a common formulation for describing generalized signal decomposition problems as optimization problems. This treatment fully embraces the handling of missing data and is extensible to many new problem formulations. When no data is missing, this framework exactly represents many methods described in this section as specific cases. Aside from the use of a masked proximal operator (described below), the proposed solution method is based on well known algorithms, block coordinate descent (BCD) and the alternating direction method of multipliers (ADMM). We note that ADMM is a common choice for convex demixing problems [54], and that we are able to apply BCD to these problems because of the structure that we enforce on the signal decomposition models that the first term be a mean-square-small residual term.

4

Solution Methods

In this section we describe two related methods for solving the SD problem (when it is convex), and approximately solving it (when it is not convex). Both rely on the masked proximal operators of the component class losses, but aside from that, they are small variations of block coordinate descent and the alternating direction method of multipliers. Finally, we describe a hybrid algorithm, combining the BCD and ADMM approaches.

4.1 Masked proximal operator

Recall that the *proximal operator* [56], [61] of ϕ_k is defined as

$$\begin{aligned}\mathbf{prox}_{\phi_k}(v) &= \operatorname*{argmin}_x \left(\phi_k(x) + \frac{\rho}{2}\|x - v\|_F^2\right) \\ &= \operatorname*{argmin}_x \left(\phi_k(x) + \frac{\rho}{2}\sum_{t,i}(x_{t,i} - v_{t,i})^2\right),\end{aligned}$$

where ρ is a positive parameter, and $v \in \mathbf{R}^{T\times p}$. When ϕ_k is convex, the function minimized is strictly convex, so there is a unique argmin. When ϕ_k is not convex, there can be multiple argmins; we simply choose one.

The *masked proximal operator* is defined as

$$\begin{aligned}\mathbf{mprox}_{\phi_k}(v) &= \operatorname*{argmin}_x \left(\phi_k(x) + \frac{\rho}{2}\|\mathcal{M}(x-v)\|_2^2\right) \\ &= \operatorname*{argmin}_x \left(\phi_k(x) + \frac{\rho}{2}\sum_{(t,i)\in\mathcal{K}} (x_{t,i}-v_{t,i})^2\right).\end{aligned}$$

Roughly speaking, it is the proximal operator, with the norm term only taken over known entries. (The masked proximal operator depends on $\mathcal{K}$, but we suppress this dependency to keep the notation lighter.) The function minimized in the masked proximal operator need not have a unique minimizer, even when ϕ_k is convex. In this case, we simply pick one.

When the function ϕ_k takes on the value ∞ (*i.e.*, encodes constraints), the point $x = \mathbf{mprox}_{\phi_k}(v)$ is feasible, *i.e.*, satisfies $\phi_k(x) < \infty$. We also note that $\mathbf{mprox}_{\phi_k}(v)$ does not depend on $v_{t,i}$ for $(t,i) \in \mathcal{U}$, so we have

$$\mathbf{mprox}_{\phi_k}(v) = \mathbf{mprox}_{\phi_k}(\mathcal{M}^*\mathcal{M}v). \tag{4.1}$$

When there are no unknown entries, *i.e.*, $\mathcal{U} = \emptyset$, the masked proximal operator reduces to the standard proximal operator. There is another simple connection between the proximal operator and the masked proximal operator. Starting with a loss function ϕ, we define the function

$$\widetilde{\phi}(z) = \inf\{\phi(\mathcal{M}^*\mathcal{M}z + u) \mid \mathcal{M}u = 0\},$$

which is, roughly speaking, the original loss function where we minimize over the unknown entries in y. If ϕ is convex, so is $\widetilde{\phi}$, since it is its partial minimization [21, §3.2.5]. The masked proximal operator is then

$$\mathbf{mprox}_{\phi}(v) = \mathbf{prox}_{\widetilde{\phi}}(v),$$

the proximal operator of the partially minimized loss function.

For many component loss functions we can work out the masked proximal operator analytically. In many other cases we can compute it with reasonable cost, often linear in T, the length of the signals. The monographs [61, §6] and [20] discuss the calculation of proximal operators in depth and list many well known results. Many closed form proximal operators are listed in the appendix of [28]. Many of these have straightforward extensions to the masked proximal operator.

As a final generalization, we introduce the weighted proximal operator, which we define as

$$\mathbf{wprox}_{\phi_k}(v) = \operatorname*{argmin}_x \left(\phi_k(x) + \frac{\rho}{2} \sum_{(t,i)\in\mathcal{K}} w_{t,i}(x_{t,i} - v_{t,i})^2 \right),$$

with nonnegative weights $w_{t,i} \in \mathbf{R}_+$ for all $(t,i) \in \mathcal{K}$. The weighted proximal operator arises in the evaluation of certain masked proximal operators, as discussed in §5.3 and §5.4. When all the weights are one, the weighted proximal operator coincides with the masked proximal operator.

Proximal operator as SD problem. We note that the proximal operator itself can be seen as a simple instance of an SD problem, with v playing the role of y, and components x and $v - x$, with associated loss functions ϕ_k and $(\rho/2)\|\cdot\|_F^2$, respectively. The masked proximal operator is the version of this signal decomposition problem with missing entries in v.

Thus, evaluating the masked proximal operator is the same as solving a simple SD problem with two components, one of which is scaled mean-square small. Our algorithms, described below, solve (or approximately solve) the general SD problem by iteratively solving these simple two component SD problems for each component.

Surrogate gradient. When ϕ is convex, the optimality condition for evaluating the masked proximal operator $x = \mathbf{mprox}_\phi(v)$ tells us that

$$g = \rho\mathcal{M}^*\mathcal{M}(v - x) \in \partial\phi(x), \tag{4.2}$$

where $\partial\phi(x)$ is the subdifferential (set of all subgradients) of ϕ at x. So evaluating the masked proximal operator at a point v automatically gives us a subgradient of the loss at the image point $x = \mathbf{mprox}_\phi(v)$. When ϕ is not convex, we can interpret g in (4.2) as a surrogate gradient.

Stopping criterion. In both algorithms, $x^2, \dots, x^K$ are found by evaluating the loss function masked proximal operators, *i.e.*,

$$x^k = \mathbf{mprox}_{\phi_k}(v^k), \quad k = 2, \dots, K,$$

for some v^k. (The particular v^k used to find x^k depend on which algorithm is used, but each of them satisfies $v^k = \mathcal{M}^*\mathcal{M}v^k$, *i.e.*, they are zero in the unknown entries of y.) We define $x^1 = \mathcal{M}^*\mathcal{M}(y - x^2 - \cdots - x^K)$, so $x^1, \dots, x^K$ are feasible and $x^1 = \mathcal{M}^*\mathcal{M}x^1$.

We combine (2.9) with (4.2) and define the optimality residual r as

$$r = \left(\frac{1}{K-1} \sum_{k=2}^{K} \left\| \rho \mathcal{M}^*\mathcal{M}(v^k - x^k) - \frac{2}{Tp} x^1 \right\|_F^2 \right)^{1/2}, \qquad (4.3)$$

which can be written as

$$r = \left(\frac{1}{K-1} \sum_{k=2}^{K} \left\| \mathcal{M} \left(\rho(v^k - x^k) - \frac{2}{Tp} x^1 \right) \right\|_2^2 \right)^{1/2}.$$

When $r = 0$ and the losses are convex, $x^1, \dots, x^K$ are optimal.

Both algorithms use the standard stopping criterion

$$r \le \epsilon^{\text{abs}} + \epsilon^{\text{rel}} \left\| \frac{2}{Tp} x^1 \right\|_F = \epsilon^{\text{abs}} + \epsilon^{\text{rel}} \left\| \frac{2}{Tp} \mathcal{M} x^1 \right\|_2, \qquad (4.4)$$

where ϵ^{abs} and ϵ^{rel} are specified positive absolute and relative tolerances.

4.2 Block coordinate descent algorithm

The BCD algorithm repeatedly minimizes the objective in (2.7),

$$\frac{1}{Tp} \left\| \mathcal{M}y - \mathcal{M}x^2 - \cdots - \mathcal{M}x^K \right\|_2^2 + \phi_2(x^2) + \cdots + \phi_K(x^K),$$

over a single (matrix) variable x^k, holding the other variables fixed. Minimizing the objective over x^k, with x^i fixed for $i \ne k$, is the same as evaluating the masked proximal operator of ϕ_k:

$$x^k = \mathbf{mprox}_{\phi_k} \left(y - \sum_{i \ne k} x^i \right)$$

with parameter $\rho = 2/(Tp)$. (Note that the masked proximal operator does not depend on the entries of its argument that are unknown in y.) There are many choices for the sequence in which we minimize over the variables, but we will use the simplest round-robin method, updating x^2, then x^3, and on to x^K, and then back to x^2 again. This gives the SD-BCD algorithm described below, with superscript j on the variables denoting iteration number, where an iteration consists of one cycle of (successively) minimizing over $x^2, \ldots, x^K$.

Algorithm 4.2.1 Block coordinate descent algorithm for SD problem (SD-BCD)

```
*Initialize.* Set $(x^k)^0$, $k = 2, \ldots, K$, as some initial estimates.
**for** iteration $j = 0, 1, \ldots$
**for** component class $k = 2, \ldots, K$
*Update a component using masked proximal operator.*
```

$$(x^k)^{j+1} = \mathbf{mprox}_{\phi_k}\left(y - \sum_{i<k}(x^i)^{j+1} - \sum_{j>k}(x^i)^j\right).$$

In SD-BCD we use the most recently updated value for the other components, in Gauss-Seidel fashion. Since we fix $\rho = 2/(Tp)$, this algorithm contains no parameters to tune. Note that SD-BCD accesses the component class loss functions only through their masked proximal operators; in particular we never evaluate ϕ^k or its derivatives.

Stopping criterion. We evaluate the stopping criterion (4.4) at the end of each iteration, using $x^1 = \mathcal{M}^*\mathcal{M}(y - x^2 - \cdots - x^K)$ and v^k the argument of the proximal operator in SD-BCD.

Convergence. SD-BCD is evidently a descent algorithm, *i.e.*, the objective is nonincreasing in each iteration. (In fact, it is nonincreasing after each update of one of the components.) Well known simple examples show that block coordinate descent need not converge to an optimal point even when the objective is convex. There is a large body of literature on the convergence of block coordinate descent type methods. Some recent review papers inlcude [8], [65], [85] and a classic textbook that addresses the topic is [9, §3.7]. These convergence proofs often rely on randomly permutating the block update order, but we have found this has no practical effect on the convergence of SD-BCD. None of cited literature exactly proves the convergence of the algorithm presented here, so we give a simple proof that any fixed point of SD-BCD must be optimal, when the losses are all convex. When one or more loss functions are not convex, the algorithm may (and often does) converge to a non-optimal stationary point.

Fixed point of SD-BCD. Here we show that if $x^2, \ldots, x^K$ are a fixed point of SD-BCD, and the losses are all convex, then the decomposition is optimal. If these variables are a fixed point, then for $k = 2, \ldots, K$,

$$x^k = \mathbf{mprox}_{\phi_k}\left(y - \sum_{i\geq 2,\ i\neq k} x^i\right).$$

From these and (4.2) we find that for $k = 2, \ldots, K$,

$$\begin{aligned} g^k &= \frac{2}{Tp}\mathcal{M}^*\mathcal{M}\left(y - \sum_{i\geq 2,\ i\neq k} x^i - x^k\right) \\ &= \frac{2}{Tp}\mathcal{M}^*\mathcal{M}\left(y - \sum_{i=2}^{K} x^i\right) \\ &= \frac{2}{Tp}x^1 \\ &\in \partial\phi_k(x^k), \end{aligned}$$

where in the third line we use $x^1 = \mathcal{M}^*\mathcal{M}(y - \sum_{i=2}^K x^i)$. This is the optimality condition (2.9).

4.3 ADMM algorithm

Here we introduce an operator splitting method for the SD problem. The particular operator splitting method we use is the alternating directions method of multipliers (ADMM) [20], [35], [36]. The ADMM algorithm we develop for the SD problem is closely related to the sharing problem [20, §7.3] and the optimal exchange problem [20, §7.3.2], but not the same. The algorithm uses a scaled dual variable $u \in \mathbf{R}^q$, and we denote iteration number with the superscript j.

```
Algorithm 4.3.1 ADMM FOR SD PROBLEM (SD-ADMM)
Initialize. Set $u^0 = 0 \in \mathbf{R}^q$, and $(x^k)^0 \in \mathbf{R}^{T \times p}$, $k = 1, \dots, K$, as some initial estimates
for iteration $j = 0, 1, \dots$
    1. Evaluate masked proximal operators of component classes in parallel.
        $(x^k)^{j+1} = \mathbf{mprox}_{\phi_k}((x^k)^j - 2\mathcal{M}^* u^j), \quad k = 1, \dots, K.$
    2. Dual update.
        $u^{j+1} = u^j + \frac{1}{K}\left(\sum_{k=1}^{K} \mathcal{M}(x^k)^{j+1} - \mathcal{M}y\right).$
```

A detailed derivation of this algorithm is given in the Appendix. Unlike BCD, SD-ADMM is not a descent method. It is also not a feasible method: the iterates satisfy the consistency constraint $\mathcal{M}y = \mathcal{M}x^1 + \cdots + \mathcal{M}x^K$ only in the limit.

Interpretations. From the dual update, we see that u^j is the running sum of the residual in the consistency constraint, scaled by $1/K$; this term is used in the argument of the masked proximal operator to drive x^k to optimality.

Convergence with convex losses. When ϕ_k are all convex, SD-ADMM converges to a solution, and ρu^j converges to an optimal dual variable ν [20, §3.2]. In particular, the consistency constraint (2.1) holds asymptotically.

Convergence with nonconvex losses. When any of the loss functions is nonconvex, there are no convergence guarantees at all. The ADMM algorithm need not converge, and if it converges it need not converge to a solution of the SD problem. But it has been observed in practice that ADMM, when applied to nonconvex problems, often converges to a useful value, which in this case is a useful signal decomposition; see, *e.g.*, [20, §9].

Stopping criterion and final decomposition. The consistency constraint generally does not hold for the iterates. To obtain a decomposition that satisfies the consistency constraint, we can simply absorb the residual in the consistency constraint into x^1 to obtain a feasible signal decomposition. We can then evaluate the residual in (4.4), with v^k the arguments of the proximal operators in step 1 of SD-ADMM.

Choice of ρ. When the problem is convex, SD-ADMM converges to a solution for any positive value of the algorithm parameter ρ, although the practical convergence speed can be affected the choice of ρ. The natural value $\rho = 2/(Tp)$ seems to give good performance in practice. When the problem is not convex, the choice of ρ is more critical, and can affect whether or not the algorithm converges, and when it converges, the decomposition found. For such problems too, the natural choice $\rho = 2/(Tp)$ seems to often give good results, although we have found that scaling this value can improve the practical convergence for some nonconvex problems. We take $\rho = 2\eta/(Tp)$, with η in the range between 0.5 and 2.

4.4 Hybrid algorithms

Comparison of SD-BCD and SD-ADMM. For convex SD problems, SD-BCD often outperforms SD-ADMM, but not by much. For nonconvex SD problems, we have found that SD-ADMM often outperforms SD-BCD in the quality of the decomposition found. Specifically, CD-BCD often ends up converging to a poor local minimum, whereas SD-ADMM is able to find a much better (lower objective) decomposition. On the other hand, for nonconvex SD problems, one or two iterations of SD-BCD, starting from the decomposition found by SD-ADMM, can lead to a modest improvement in the objective value found. (These iterations cannot increase the objective, since SD-BCD is a descent method.)

Hybrid methods. A reasonable strategy, and the default in our implementation, is to use SD-BCD if the SD problem is convex. If the SD problem is nonconvex, the default uses SD-ADMM (with scale factor $\eta = 0.7$) until convergence, and then follows this with SD-BCD, again run until convergence (quite often, but not always, only a few iterations). This hybrid method seems to work well on a wide variety of SD problems.

Numerical examples. In this monograph we consider four numerical examples, summarized in Table 4.1. They include convex and nonconvex problems, and range from small to large, with the SD problem in `PV` having over 700,000 variables. We use these examples to illustrate the convergence of the hybrid algorithm. In Figure 4.1 we plot the residual (4.3) versus iteration number for these four problems.

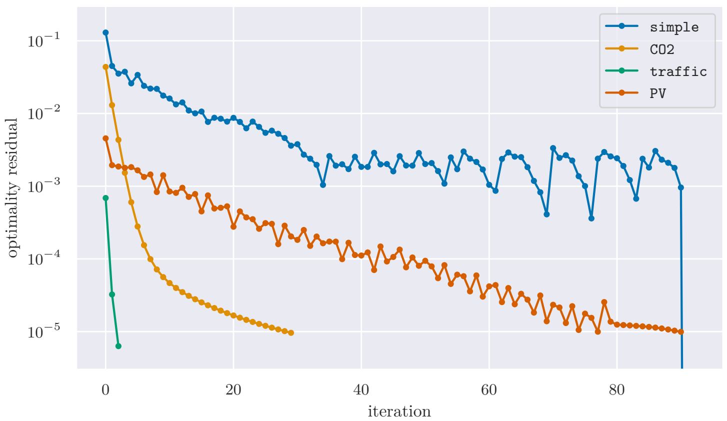

Figure 4.1: Residual versus iteration number for the 4 numerical examples given in §2.9, §7.1, §7.2, and §7.3 respectively.

We see rapid and monotonic convergence for problems `CO2` and `traffic`, which are convex. For `simple` and `PV`, which are nonconvex, we can see the switch to SD-BCD at the end, with a sharp reduction in residual in `simple` in just a few iterations, and a smoother reduction of residual over 12 iterations in `PV`. None of the examples requires more than 100 iterations to converge.

Table 4.1: Summary of numerical examples

Name	Section	K	T	p	Size (KTp)	q	Convex
`simple`	§2.9	3	500	1	1,500	500	no
`CO2`	§7.1	3	2,459	1	7,377	2,441	yes
`traffic`	§7.2	5	105,552	1	527,760	101,761	yes
`PV`	§7.3	5	20,212	7	707,420	135,899	no

5

Component Class Attributes

In this section we describe some very basic attributes that component class losses can have.

5.1 Separability

A component class loss function can be separable across time, or entries, or both.

Time-separable losses. A function $\phi : \mathbf{R}^{T \times p} \to \mathbf{R} \cup \{\infty\}$ is separable across time if it has the form

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

for some functions $\ell_t : \mathbf{R}^p \to \mathbf{R} \cup \{\infty\}$, $t = 1, \ldots, T$. It is common for the loss functions to not depend on t, in which case we say it is time-invariant. A simple example is the mean-square loss (2.3), with $\ell_t(x_t) = \frac{1}{Tp}\|x_t\|_2^2$ for all t.

Entry-separable losses. A component class function ϕ is separable across entries if it has the form

$$\phi(x) = \sum_{i=1}^{p} \ell_i(\breve{x}_i)$$

for some functions $\ell_i : \mathbf{R}^T \to \mathbf{R} \cup \{\infty\}$, $i = 1, \ldots, p$, where $\breve{x}_i$ is the ith column of x (which can be interpreted as a scalar time series), the ith entry of the vector time series $\{x_t\}$. Here too it is common for the loss function to not depend on i, in which case we say it is symmetric (in the entries of x_t). The mean-square loss (2.3) is symmetric (in addition to being time-separable).

Separability and proximal operators. Separability reduces the complexity of evaluating the masked proximal operator. For example if ϕ is separable across time, say, $\phi(x) = \sum_t \ell_t(x_t)$, its masked proximal operator is

$$\mathbf{mprox}_\phi(v) = \begin{bmatrix} \mathbf{mprox}_{\ell_1}(v_1)^T \\ \vdots \\ \mathbf{mprox}_{\ell_T}(v_T)^T \end{bmatrix},$$

i.e., we can evaluate the masked proximal operator in parallel for each time $t = 1, \ldots, T$. (Note the masked proximal operator for t depends on the missing data for that time period.)

5.2 Time-invariance

Time-invariance or shift-invariance is another important attribute. We let $M < T$ denote the memory of the loss function ϕ. We say ϕ is time-invariant if it has the form

$$\phi(x) = \sum_{t=1}^{T-M+1} \ell(x_{t:t+M-1}),$$

where $x_{t:t+M-1}$ is the $M \times p$ slice of x, that includes rows $t, \ldots, t+M-1$, and $\ell : \mathbf{R}^{M \times p} \to \mathbf{R} \cup \{\infty\}$ is the slice loss. Thus, a time-invariant component class loss is sum of the slice loss, applied to all M-long slices of its argument. With this definition, a time-separable time-invariant loss is a special case of time-invariance, with memory $M = 1$.

The second-order mean-square smooth loss (2.12) is a simple example of a time-invariant component class loss, with $M = 3$. As another example, consider the class of P-periodic signals, with loss

$$\phi(x) = \begin{cases} 0 & x_{t+P} = x_t, \quad t = 1, \ldots, T-P, \\ \infty & \text{otherwise}, \end{cases} \tag{5.1}$$

which has memory $M = P + 1$.

5.3 Convex quadratic

A loss is convex quadratic if it has the form

$$\phi(x) = \begin{cases} (1/2) x_:^T P x_: + q^T x_: + r & A x_: = b \\ \infty & \text{otherwise}, \end{cases} \tag{5.2}$$

where $x_: \in \mathbf{R}^{Tp}$ is a vector representation of x (and $x_:^T$ is its transpose), $P \in \mathbf{R}^{Tp \times Tp}$ is symmetric positive semidefinite, $q \in \mathbf{R}^{Tp}$, $r \in \mathbf{R}$, $A \in \mathbf{R}^{L \times Tp}$, and $b \in \mathbf{R}^L$. Thus ϕ is convex quadratic, with some equality constraints. We have already encountered a few examples of convex quadratic loss functions, such as mean-square small and mean-square smooth.

As a more interesting example, consider the P-periodic smooth loss, defined as

$$\phi(x) = \frac{1}{Pp} \left(\|x_2 - x_1\|_2^2 + \cdots + \|x_P - x_{P-1}\|_2^2 + \|x_1 - x_P\|_2^2 \right), \tag{5.3}$$

provided x is P-periodic, *i.e.*, $x_{t+P} = x_t$ for $t = 1, \ldots, T-P$, and $\phi(x) = \infty$, otherwise. This is the same as the P-periodic loss (5.1), with mean-square smoothness, taken circularly.

Masked proximal operator of convex quadratic loss. The masked proximal operator of a convex quadratic loss function can be efficiently evaluated; more precisely, after the first evaluation, subsequent evaluations can be carried out more efficiently. Evaluating the masked proximal operator involves minimizing a convex quadratic function subject to equality constraints, which in turn can be done by solving a set of linear equations, the KKT (Karush-Kuhn-Tucker) equations [22, §16]. If we cache the factorization used to solve this set of linear equations (*e.g.*, the LDL^T factorization of the coefficient matrix), subsequent evaluations require only the so-called back-solve step, and not the factorization. This idea is often exploited in ADMM; see [20, §4.2].

Weighted proximal operator. In evaluating the masked proximal operators of certain convex quadratic loss functions, it can more computationally efficient to evaluate a related weighted proximal operator. This is seen commonly with loss functions that are P-periodic. In this case, the solution to the masked proximal operator may be found by evaluating a smaller weighted proximal operator. Specifically, the weighted proximal operator is evaluated over a vector $z \in \mathbf{R}^{P\times p}$, representing a single period of component. The input to this smaller proximal operator is the original input, averaged across periods, using only the available data, *e.g.*, the entries in $\mathcal{K}$. The weights are defined as the number of real entries used in each averaging operation, divided by the total possible number of entries. (Some additional care must be taken here when evaluating signals that are not an even multiple of the period length.)

5.4 Common term

Another common attribute of a component class is when it represents a common term across the entries of the signal. The loss has the form

$$\phi(x) = \begin{cases} \tilde{\phi}(z) & x_t = z_t\mathbf{1}, \quad t = 1,\ldots,T \\ \infty & \text{otherwise,} \end{cases} \tag{5.4}$$

where $\tilde{\phi} : \mathbf{R}^T \to \mathbf{R} \cup \{\infty\}$ is a loss function for a scalar signal. Roughly speaking, this component class requires that all entries of x (*i.e.*, its columns) are the same, and uses a scalar-valued signal loss function on the common column. If $\tilde{\phi}$ is separable, then ϕ is separable across time.

The proximal operator of such a ϕ is readily found in terms of the proximal operator of $\tilde{\phi}$. It is

$$\mathbf{prox}_{\phi}(v) = \mathbf{prox}_{\tilde{\phi}}((1/n)v\mathbf{1})\mathbf{1}^T.$$

In words: to evaluate the proximal operator for a common term loss function, we first average the columns of v, then apply the proximal operator of $\widetilde{\phi}$, and finally broadcast the result to all columns.

The masked proximal operator is a bit more complex. Each row can have a different number of entries in the known set, so the average across columns must be taken with respect to the number of real entries in the row instead of the number of columns. However, to make use of the scalar formulation $\widetilde{\phi}(z)$, we must invoke the weighted proximal operator,

$$\begin{aligned}\mathbf{mprox}_{\phi}(v) &= \underset{x}{\operatorname{argmin}}\left(\widetilde{\phi}(z) + \frac{\rho}{2}\sum_{t,i\in\mathcal{K}}(x_{t,i} - v_{t,i})^2\right), \quad \text{s.t. } x_t = z_t\mathbf{1} \\ &= \mathbf{wprox}_{\widetilde{\phi}}(\mathbf{ravg}(v))\mathbf{1}^T,\end{aligned}$$

where $\mathbf{ravg} : (\mathbf{R} \cup \{?\})^{T\times p} \to (\mathbf{R} \cup \{?\})^T$ is the row-wise average of the matrix v, over only the known entries. (If a row has no known entries, the function returns ? for that time index.) The weights are the number of known entries used in each averaging operation, divided by the total possible number of entries.

6

Component Class Examples

There is a wide variety of useful component classes; in this section we describe some typical examples. In most cases the proximal operator of the loss is well known, and we do not give it; we refer the reader to other resources, such as [20], [28], [61]. When the loss function is convex, but an analytical method to evaluate the proximal operator is not known, we can always fall back on a numerical method, *e.g.*, using CVXPY [1], [30]. In a few cases where we believe our method of evaluating the proximal operator is new, we give a short description of the method.

6.1 Time-separable classes

Time-separable classes are given by the loss functions ℓ_t on $\mathbf{R}^p$. We have already seen the mean-square small class, with loss $\ell_t(u) = \frac{1}{Tp}\|u\|_2^2$, and the finite set class, which requires that x_t be one of a given set of values. We mention a few other examples in this section.

Value constraint component classes. As an extension of the finite value class, we require that $x_t \in \mathcal{S}_t$, where $\mathcal{S}_t \subset \mathbf{R}^p$ is some given set. If $\mathcal{S}_t$ are all convex, we have a convex loss function. Simple convex examples include the nonnegative component class, with $\mathcal{S} = \mathbf{R}_+^p$, and the vector interval signal class, with $\mathcal{S} = \{u \mid x_t^{\min} \leq u \leq x_t^{\max}\}$, where the inequality is elementwise and $x_t^{\min}$ and $x_t^{\max}$ are given lower and upper limits on the entries of the signal (which can be parameters). In addition to the constraint $x_t \in \mathcal{S}_t$, we can add a nonzero penalty function of x_t to the objective.

Mean-square close entries. The loss

$$\ell(u) = \frac{1}{p}\sum_{i=1}^{p}(u_i - \mu)^2, \qquad \mu = \frac{1}{p}\sum_{i=1}^{p} u_i, \tag{6.1}$$

which is the variance of the entries of the vector u, defines the mean-square close entries class. If we scale this class by a very large weight, this gives an approximation of the common term class (5.4) (with $\widetilde{\phi} = 0$), in which the entries of the signal must be the same for each t.

Robust losses. We can modify the sum of squares loss so the component class can include signals with occasional outliers, using so-called robust losses, which grow linearly for large arguments, when they are convex, or sub-linearly when they are not. One well-known examples is the Huber loss, defined as

$$\ell(u) = \sum_{i=1}^{p} H(u_i), \qquad H(a) = \begin{cases} a^2 & |a| \leq M \\ M(2|a| - M) & |a| > M, \end{cases}$$

where $M > 0$ is a parameter [21, §6.1.2]. An example of a nonconvex robust loss is the log Huber loss,

$$\ell(u) = \sum_{i=1}^{p} \widetilde{H}(u_i), \qquad \widetilde{H}(a) = \begin{cases} a^2 & |a| \leq M \\ M^2(1 + 2\log(|a|/M)) & |a| > M. \end{cases}$$

Convex sparsity inducing losses. The loss function $\ell(u) = \|u\|_2$ (note that this norm is not squared) leads to vector-sparse (also called block sparse) component signals, *i.e.*, ones for which for many values of t, we have $x_t = 0$. In machine learning this is referred to as group lasso [39, §3.8.4]. With this loss, we typically find that when $x_t \neq 0$, all its entries are nonzero. The loss function $\ell(u) = \|u\|_1$, sum-absolute small component class, tends to yield signals that are component-wise sparse, *i.e.*, for many values of (t, i), we have $x_{t,i} = 0$.

Non-convex sparsity inducing losses. The most obvious one is the cardinality or number of nonzeros loss, with $\ell(u)$ being the number of nonzero entries in u (or, in the vector version, 0 if $u = 0$ and 1 otherwise). In this case the overall loss $\phi(x)$ is the number of nonzero values of $x_{t,i}$. A variation is to limit the number of nonzeros to some given number, say, r, which gives the r-sparse signal component class.

These losses are nonconvex, but have well-known analytic expressions for their proximal operators. For example when the loss is the number of nonzero entries in x, the proximal operator is so-called hard thresholding [20, §9.1.1],

$$\mathbf{prox}_{\phi_k}(v)_{t,i} = \begin{cases} 0 & |v_{t,i}| \leq \sqrt{2/\rho} \\ v_{t,i} & |v_{t,i}| > \sqrt{2/\rho}, \end{cases} \qquad t = 1, \ldots, T, \quad i = 1, \ldots, p.$$

Quantile small. The quantile loss [48], [49] is a variation on the ℓ_1 loss $\|u\|_1$, that allows positive and negative values to be treated differently:

$$\ell(u) = \sum_{i=1}^{p} \left(|u_i| + (2\tau - 1)u_i\right), \tag{6.2}$$

where $\tau \in (0, 1)$ is a parameter. For $\tau = 0.5$, this class simplifies to sum-absolute small. (Its proximal operator is given in [61, §2.2, §6.5.2].)

6.2 Time-invariant classes

Any time separable loss for which ℓ_t do not depend on t is time-invariant. We give a few other examples here.

Index-dependent offset. In the common term class (5.4), the entries of signals are the same. The index-dependent offset class is analogous: Its signals are different for different indexes, but the same over time. It is given by $\phi(x) = 0$ if for some z, $x_t = z$ for all t, where $z \in \mathbf{R}^p$, and ∞ otherwise. Of course we can add a penalty on z. This loss is time-invariant, with a memory of one.

Higher order mean-square smooth component classes. We have already mentioned the mean-square smooth class which uses the first-order difference (2.4), and its extension to the second-order difference (2.12). Higher order mean-square smooth classes use higher order differences.

Mean-absolute smooth. Replacing the mean-square penalty in mean-square first-order smooth classes with a average-absolute penalty yields a components whose signal entries are typically piecewise constant. With the second-order difference,

$$\phi(x) = \frac{1}{(T-2)p} \sum_{t=1}^{T-2} \|x_t - 2x_{t+1} + x_{t+2}\|_1, \tag{6.3}$$

we obtain a class who entries are typically piecewise linear. (This is discussed under the name ℓ_1-trend filtering in §3.)

Periodic. The component class of signals with period P has loss function

$$\phi(x) = \begin{cases} 0 & x_{t+P} = x_t, \quad t = 1, \ldots, T-P, \\ \infty & \text{otherwise.} \end{cases} \tag{6.4}$$

We can also express this using a basis.

To this constraint we can add a loss function such as mean-square signal or mean-square smooth, to obtain, for example, the component class of P-periodic mean-square smooth signals. (In this case the differences are computed in a circular fashion.)

Quasi-periodic. A variation on the periodic signal class does not require strict periodicity, but allows some variation period to period, with a penalty for variation. The simplest version uses the quadratic loss function

$$\phi(x) = \sum_{t=1}^{T-P} \|x_{t+P} - x_t\|_2^2, \tag{6.5}$$

the sum of squares of the differences in signal values that are P period apart. Variations include adding a smoothness term, or replacing the sum of squares with a sum of norms, which tends to give intervals of time where the signal is exactly periodic.

Composite classes. Components may be combined to generate more complex loss functions. An example that we will use later has time entries that are smooth (2.12) and periodic (6.4) and entries that are mean-square close (6.1),

$$\phi(x;\lambda_1,\lambda_2) = \begin{cases} \lambda_1\ell_1(x)+\lambda_2\ell_2(x) & x_{t+P}=x_t, \quad t=1,\ldots,T-P, \\ \infty & \text{otherwise.} \end{cases} \tag{6.6}$$

where

$$\begin{aligned} \ell_1(x) &= \frac{1}{(T-2)p}\sum_{t=1}^{T-2}\|x_t-2x_{t+1}+x_{t+2}\|_2^2, \\ \ell_2(x) &= \frac{1}{p}\sum_{t=1}^{T}\sum_{i=1}^{p}(x_{t,i}-\mu_t)^2, \quad \mu_t=\frac{1}{p}\sum_{i=1}^{p}x_{t,i}. \end{aligned}$$

This composite example is convex quadratic (§5.3).

Monotone non-decreasing. The monotone nondecreasing loss is

$$\phi(x) = \begin{cases} 1 & x_{t+1,i}\ge x_{t,i} \quad t=1,\ldots,T-1, \quad i=1,\ldots,p \\ 0 & \text{otherwise.} \end{cases}$$

It is used in monotone or isotonic regression, typically to represent something like cumulative wear, that does not decrease over time. This loss is a constraint, but we can add an additional term such as mean-square smoothness.

Markov. The Markov class is, roughly speaking, an extension of the finite set class (2.10) that includes costs for the different values, as well as transitions between them. It is specified by some distinct values $\theta_1,\ldots,\theta_M\in\mathbf{R}^p$, a transition cost matrix $C\in\mathbf{R}_+^{M\times M}$, and state cost vector $c\in\mathbf{R}_+^M$. Like the finite set component class, the loss is ∞ unless for each t, we have $x_t\in\{\theta_1,\ldots,\theta_M\}$. We write this as $x_t=\theta_{s_t}$, where we interpret $s_t\in\{1,\ldots,M\}$ as the state at time t. When this holds, we define

$$\phi(x)=\sum_{t=1}^{T}c_{s_t}+\sum_{t=2}^{T}C_{s_t,s_{t-1}}.$$

The first term is the state cost, and the second is the cost of the state transitions.

This component class gets it name from a statistical interpretation in terms of a Markov chain. If the state s_t is a Markov chain with states $\{1, \ldots, M\}$, with transition probabilities $\pi_{ij} = \mathbf{Prob}(s_t = i \mid s_{t-1} = j)$. Then with $c = 0$ and $C_{i,j} = \log \pi_{ij}$, the loss is the negative log-likelihood, up to a constant.

The proximal operator of this component loss function can be efficiently evaluated using standard dynamic programming. We create a graph with MT nodes, with each node corresponding to one state at one time. All nodes at time t are connected to all nodes at time $t-1$ and $t+1$, so there are $(T-1)M^2$ edges. Let v be the signal for which we wish to evaluate the proximal operator. At each node we attach the cost $(\rho/2)\|v_t - c_s\|_2^2$, and on each edge from state s at time $t-1$ to state s' at t we attach the cost $C_{s,s'}$. Then $\phi(x) + (\rho/2)\|v - x\|_F^2$ is exactly the path cost through this graph. We can minimize this over $s_1, \ldots, s_T$ using dynamic programming to find the shortest path. The cost is $O(TM^3)$ flops, which is linear in the signal length T.

Single jump. As a variation on the Markov component class we describe the single jump component class. We describe it for a scalar signal *i.e.*, $p = 1$; it is extended to vector signals with a loss that is separable across entries. The loss function is

$$\phi(x) = \begin{cases} 1 & x = (0_\tau, a\mathbf{1}_{T-\tau}) \\ 0 & x = 0 \\ \infty & \text{otherwise}, \end{cases} \tag{6.7}$$

for some (jump magnitude) $a \neq 0$ and some (jump time) $\tau \in \{1, \ldots, T\}$. Roughly speaking, feasible signals start at zero and either stay zero, or jump once, at a time τ, to the value a. The cost is zero if x is zero, and one if it does jump. This loss function is evidently nonconvex.

Its proximal operator is readily evaluated directly, by evaluating

$$(\rho/2)\|x - v\|_2^2 + \phi(x)$$

for all feasible x. For $x = 0$ we have the value $(\rho/2)\|v\|_2^2$. For a jump at time τ, the value of a that minimizes the cost above is simply the average of x_t over $t = \tau, \ldots, T$. This value and the cost is readily computed recursively, so the proximal operator can be evaluated in time linear in T. This method extends reaadily to the masked proximal operator.

6.3 Fitting component class losses

In the discussion above we specify component classes directly in terms of the loss function. We mention here that it is also possible to fit a component class loss from examples of signals in that class, assuming they are available.

One simple method is based on the statistical interpretation given in §2.4. Given a collection of example signals, we fit a statistical model, for example a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ with an appropriate mean $\mu \in \mathbf{R}^{Tp}$ and covariance $\Sigma \in \mathbf{R}^{Tp \times Tp}$. We use as loss for this component class the convex quadratic $\phi(x) = (x_: - \mu)^T \Sigma^{-1} (x_: - \mu)$, which is the negative log-likelihood, up to a scale factor and constant. If we fit a statistical model for each component of the signals we obtain an entry-separable loss; if we fit a common model for the entries of the signal, we obtain an entry-separable symmetric loss. We can fit a time-invariant loss by creating a common statistical model of all M-long slices of the signal examples, and using the negative log-likelihood as the slice loss.

Another elementary method for fitting a loss to example signals uses the singular value decomposition (SVD) or generalized low-rank model [83] to find a set of archetype signals $a^1, \ldots, a^r \in \mathbf{R}^{T \times p}$, for which each of the examples is close to a linear combination of them. We then use the basis loss function

$$\phi(x) = \begin{cases} 0 & x = z_1 a^1 + \cdots + z_r a^r \text{ for some } z \in \mathbf{R}^r \\ \infty & \text{otherwise.} \end{cases} \tag{6.8}$$

(As a variation on this, we can find a set of (scalar) archetypes in $\mathbf{R}^T$ for which each component of the examples in close to a linear combination of them, as in (2.11).) A soft version of the basis loss is the loss function

$$\phi(x) = \min_z \| x - z_1 a^1 - \cdots - z_r a^r \|_F^2, \tag{6.9}$$

which has full domain. (It can also be thought of as a combination of two classes: the basis class, and the a mean-square small residual class.)

The soft basis model can be used to fit a time-invariant loss. We use SVD to find a set of archetypes or basis for which each M-long slice of each exmaple is close to a linear combination, and then use the soft basis loss (6.9) as the slice loss.

7

Examples

7.1 Mauna Loa CO_2 measurements

An example often used to demonstrate seasonal-trend decomposition is atmospheric carbon dioxide (CO_2), which has both a strong seasonal component and a underlying trend. These data were utilized in the original STL paper [27] as well as the documentation for various implementations of STL [71]. In this section we compare the Python implementation of STL in the `statsmodels` package to an SD formulation of the problem of decomposing measurements of atmospheric CO_2 into seasonal, trend, and residual components.

Data set. The weekly average CO_2 measured at Mauna Loa, HI from May 1974 through June 2021, available online from the National Oceanic and Atmospheric Administration Global Monitoring Laboratory [74], is shown in Figure 7.1. The data set is a scalar signal of length 2459 with 18 missing entries. In our notation, $y_1, \ldots, y_T \in \mathbf{R} \cup \{?\}$, with $T = 2459$, and $|\mathcal{U}| = 18$.

Decomposition using STL. We use the implementation in `statsmodels` (v0.12.2) with default settings and `period=52`. We note that while

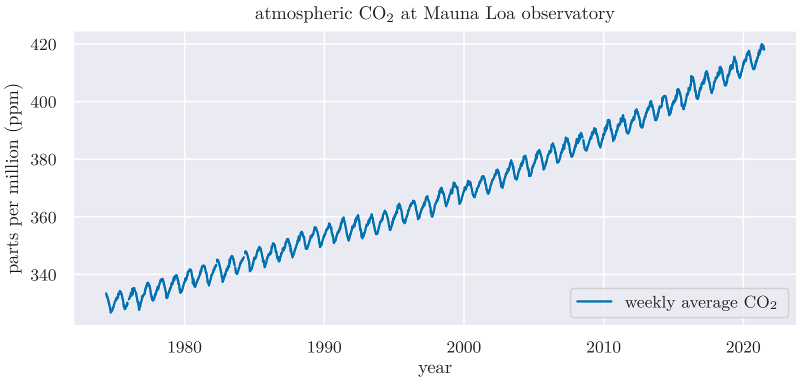

Figure 7.1: Atmospheric CO_2 data obtained from NOAA, which shows clear seasonal and trend components.

the original STL paper describes how to handle missing data, this particular software implementation cannot handle missing values, so we used simple linear interpolation to fill the missing values before running the algorithm. The resulting decomposition is shown in Figure 7.2, using the conventional names for the components. Interestingly, the "seasonal" component in this estimation is not periodic; it almost repeats each year but with some variation.

Decomposition using SD. We form an SD problem with $p = 1$, $T = 2459$, and $K = 3$, with component classes mean-square small (2.3), second-order-difference small (2.12), and a quasi-periodic signal with period 52 (6.5). All the component classes are convex, so this SD problem is convex. This problem has two parameters λ_2 and λ_3, associated with the weights on the second and third loss functions respectively. We found that $\lambda_2 = 10^4$ and $\lambda_3 = 1$ give good results, although better parameter values could be found using a validation procedure. The resulting decomposition is shown in Figure 7.3.

Comparison. The decompositions found using STL and SD, shown in figures 7.2 and 7.3, and nearly identical. The RMS deviation between trend estimates is 7.52×10^{-2}, about 0.02% of the average measured

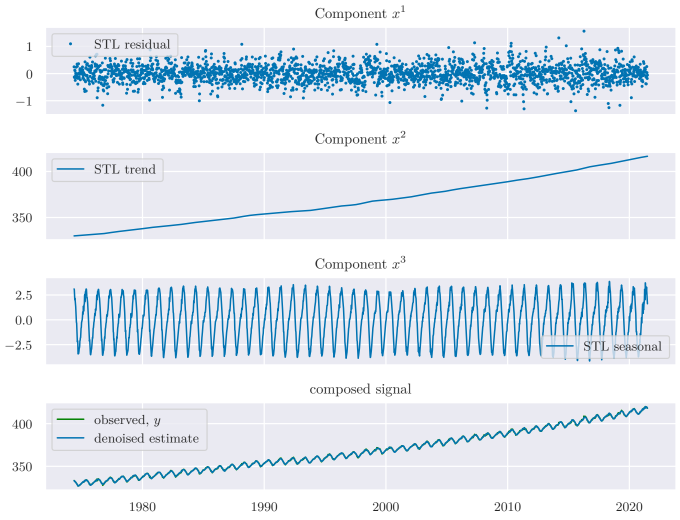

Figure 7.2: Decomposition of the CO_2 data into residual, trend, and seasonal components, using STL.

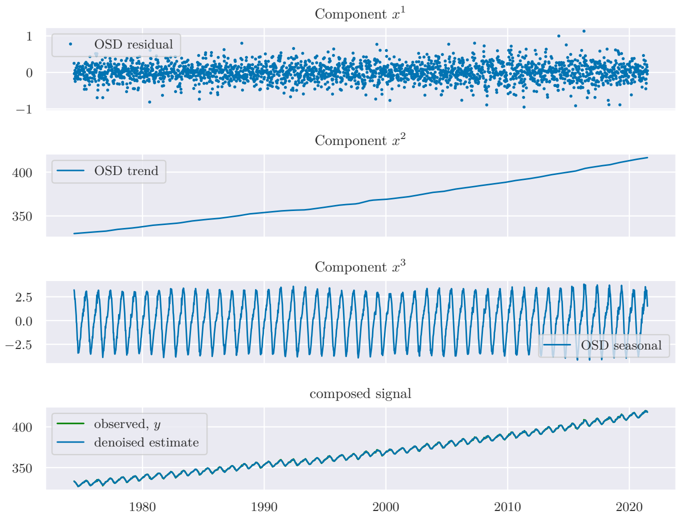

Figure 7.3: Decomposition of the CO_2 data into residual, trend, and seasonal components, using SD. It is nearly identical to the decomposition found by STL, shown in Figure 7.2.

value. The RMS deviation between seasonal estimates is 8.79×10^{-2}. While STL is based on a heuristic algorithm, SD is based on solving a convex optimization problem (for our particular choice of loss functions).

7.2 RFK bridge traffic

This example illustrates how the concept of seasonal-trend decomposition can be extended in the SD framework to handle more complex analyses with additional components. Traffic volume is measured with sensors embedded in the roadways that count the number of cars that pass in each hour; from these data, summary statistics such as "Annual Average Daily Traffic" and "Peak Hour Volume" are derived [67].

Data set. The hourly outbound vehicle count for the Manhattan toll plaza on the Robert F. Kennedy Bridge in New York City from January 1, 2010 through August 28, 2021 is shown in Figure 7.4 as a heat map, with the hour of day shown vertically and the day shown horizontally, and missing entries shown in white. Daily and seasonal variations can be seen, along with the effects of COVID-19. A single week of data is shown in Figure 7.5, where daily variation, and the weekend effect, are evident. The data set is made available online by the New York Metropolitan Transportation Authority (MTA) [57].

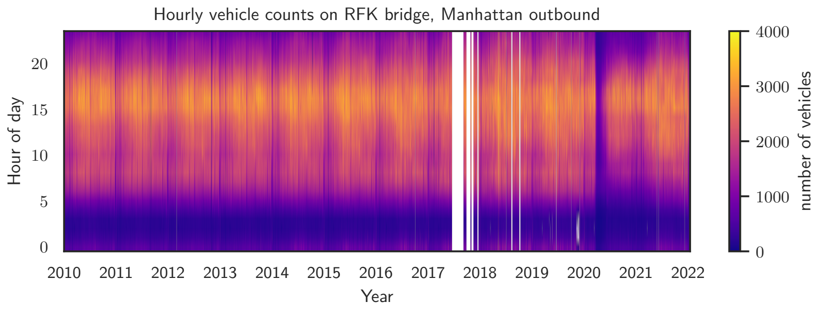

Figure 7.4: Hourly vehicle counts for the outbound Manhattan toll plaza of the Robert F. Kennedy bridge, with hour of day on the y-axis and days on the x-axis. White pixels represent missing values.

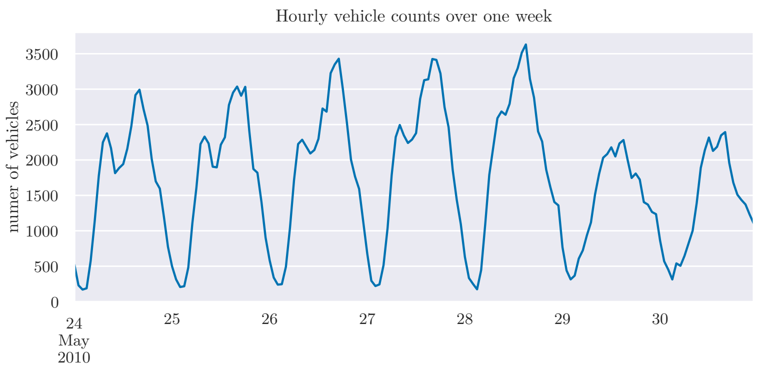

Figure 7.5: One week of hourly vehicle counts in May 2010.

The data y is scalar (*i.e.*, $p = 1$), with $T = 102192$ (24 hours per day, 4258 days). We take the natural logarithm of the data, using the convention $\log 0 =?$. With these unknown entries, plus those that are unknown in the original data set, we have $|\mathcal{U}| = 3767$. Thus the decomposition is multiplicative; the components are multiplied to obtain the decomposition.

SD problem formulation. We form an SD problem with $K = 5$ components. The residual component is mean-square small (2.3), as in previous examples. The second component is the weekly baseline, which is the smooth-periodic cost given in (5.3) with $P = 168$ and a weight parameter λ_2. The third component is the yearly seasonal correction, which is also smooth-periodic (5.3) with $P = 8760$ and weight parameter λ_3, and the additional constraint that the sum over each period must be equal to zero. The fourth component is the long-term trend, modeled as piecewise linear with the ℓ_1 second difference loss (6.3), with weight parameter λ_4 and the additional constraint that the first value of x^4 must be equal to zero. The fifth and final component is a sparse daily outlier, defined as

$$\phi_5(x) = \begin{cases} \lambda_5 \|x\|_1 & x \in \mathcal{D} \\ \infty & \text{otherwise,} \end{cases} \tag{7.1}$$

where $\mathcal{D}$ is the set of signals that are constant over each day. All the component class losses are convex, so this SD problem is convex with parameters λ_2, λ_3, λ_4, and λ_5.

Results. We solve the SD problem using parameter values

$$\lambda_2 = 10^{-1}, \quad \lambda_3 = 5 \times 10^5, \quad \lambda_4 = 2 \times 10^5, \quad \lambda_5 = 1,$$

selected by hand to provide good results. The decomposition yields components that have vastly different timescales.

By exponentiating the component estimates, $\tilde{x}^k = \exp(x^k)$, we recover a multiplicative model of the traffic count data. The residual component $\tilde{x}^1$ is centered around 1, with 90% of the residuals in the interval $[0.74, 1.30]$, shown in Figure 7.6. This means that in any given hour, the decomposition predicts traffic typically within around $\pm 30\%$.

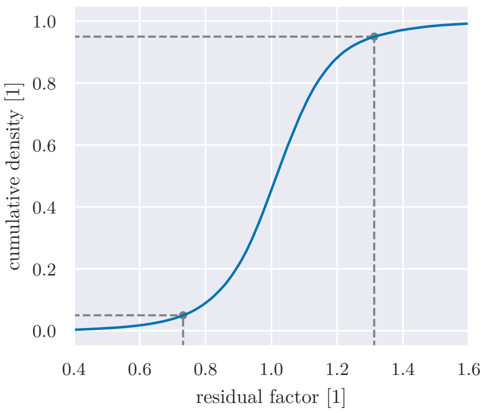

Figure 7.6: Cumulative distribution function of the multiplicative residual $\tilde{x}^1_{t,i}$ for $(t, i) \in \mathcal{K}$. The gray dashed lines indicate the 5th and 95th percentiles. 90% of the residuals are between 0.74 and 1.30.

Figure 7.7 shows one week of the (periodic) weekly baseline. We see many of the phenomena present in Figure 7.5, such as reduced traffic over the weekend, daily variation, and a small increase from Monday to Friday, and a commute rush hour on weekdays.

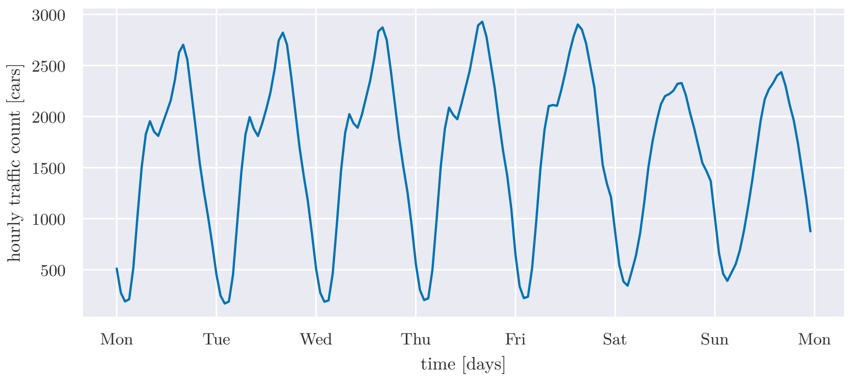

Figure 7.7: Weekly baseline signal $\tilde{x}^2$, shown for a single week (168 values).

Figure 7.8 shows component $\tilde{x}^3$, the seasonal correction factor, which varies from around -9% to $+7\%$, with the peak in summer and the low point in late January and early February.

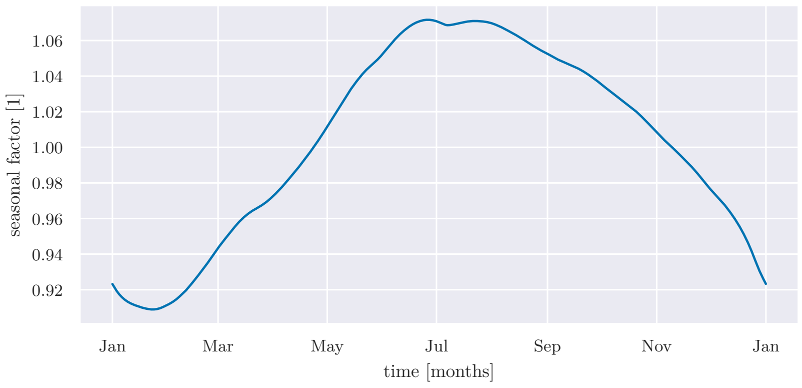

Figure 7.8: Seasonal adjustment $\tilde{x}^3$, shown for a single year (8760 values).

Figure 7.9 shows the long term $\tilde{x}^4$. The component x^4 is piecewise-linear with a small number of breakpoints, so $\tilde{x}^4$ is piecewise exponential, with a small number of breakpoints, shown as red dots in the plot. We can see a slight increase in traffic over the first 10 years followed by the a precipitous drop in traffic due to COVID-19 in early 2020, coinciding with the mandatory lockdown implemented by the New York state government on March 22, 2020 [29].

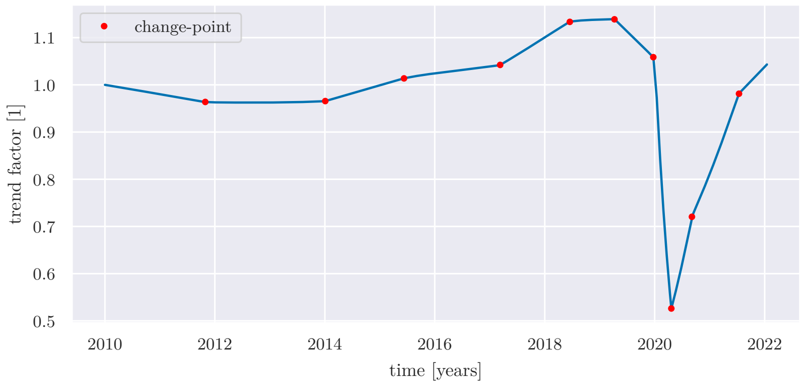

Figure 7.9: Long-term trend multiplicative factor $\widetilde{x}^4$. This trend is piecewise exponential, with breakpoints shown as red dots.

The final component x^5 is sparse, which means that $\widetilde{x}^5$, shown in Figure 7.10, mostly takes on the value one. This component identifies 42 days (out of 4258) as outliers, with multiplicative corrections ranging from around 0.2 (*i.e.*, one fifth the normal traffic on that day) to around twice the normal traffic on that (one) day. All but two of the outliers represent a decrease in traffic on that day. Many of the detected outlier days are weather related, with some notable examples being various blizzards including February 10, 2010 [11], December 27, 2010 [12], January 27, 2015 [13], and February 1, 2021 [14]. About 9 outlier days are associated with reduced traffic during the COVID-19 lockdown event in early 2020. Figure 7.11 highlights the detection of Hurricane Irene in August of 2011 [4], with $\widetilde{x}^5 < 1$ during the hurricane.

The two positive outlier days occur on May 6 and 10, 2018. The authors could find no explanation for the very high measured traffic on those days in the archives of the New York Times and the New York Post. It is possible that sensors were simply malfunctioning on those two days.

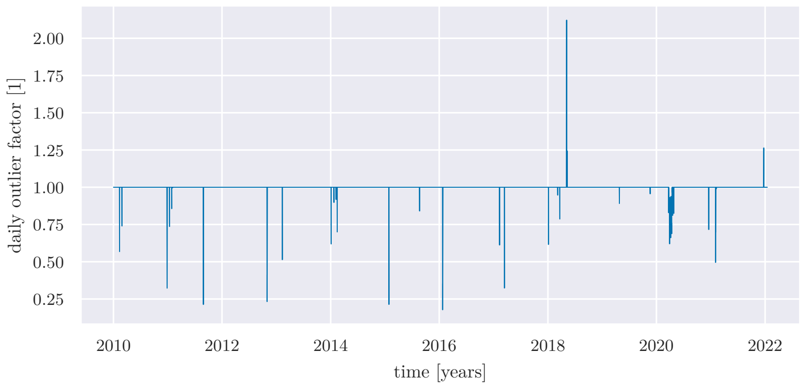

Figure 7.10: Daily outlier component $\widetilde{x}^5$.

7.3 Outage detection in a photovoltaic combiner box

Data set. We consider a set of 7 measurements of real power from inside a photovoltaic (PV) combiner box [34], corresponding to 7 strings of series-connected PV modules that are joined in parallel. These data are from PV strings forming the canopy at the NIST campus in Maryland [19]. Detailed documentation of the PV systems at this site, including system designs, meteorological station information, and site layout, are also available [18]. The canopy has multiple roof orientations, so the constituent strings have similar but different power curves, depending on the specific geometry of each string.

The raw data consist of the power output of each of the 7 PV strings, measured each minute over a month (August 2016), organized into a matrix with each column corresponding to a single string and each row a particular minute. This raw data contains some missing data. The power output of each string depends on available sunlight, weather conditions, soiling accumulation, string geometry, and local shade patterns. Two days of string power output are shown in Figure 7.12.

Data pre-processing. We first eliminate all data points corresponding to night time and early morning and evening, when string powers are zero or very small. We removed data between 5:40pm and 6:49am.

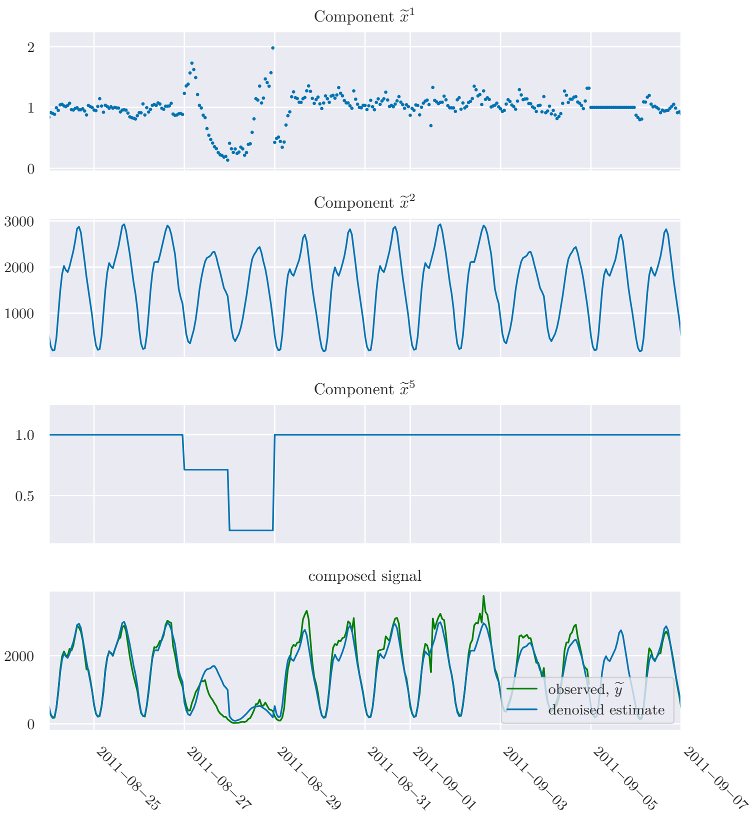

Figure 7.11: Decomposition components for two weeks in August 2011 (336 values). Hurricane Irene hit New York city on August 27 and 28, greating reducing traffic on those days, clearly seen as outliers in $\widetilde{x}^5$.

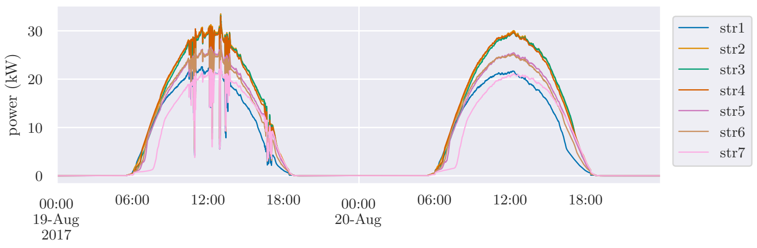

Figure 7.12: Raw PV combiner box data, shown for two days in August 2017.

(These times were found as the times when the whole system was producing less than 10% of system capacity.) Thus each day consists of 652 one minute measurements. Next we scale each of the 7 string powers (columns) so that the 95th percentile is one. This gives each column an approximate range of about 0.1 to 1.3.

Finally we take the log of each power output value, resulting in columns with a range of about -2.3 to 0.25. Carrying out signal decomposition on this log signal gives us a multiplicative decomposition, which makes sense for this application. (For example, a cloud passing between the sun and the string gives a percentage reduction in power.) The final data is a signal y with $T = 20212$, $p = 7$, and $|\mathcal{U}| = 6402$.

Outage simulation. We modify this real data to include some simulated faults or outages, where some part of each PV string no longer generates power. This is modeled as a (multiplicative) reduction in power output, from the time of failure to the end of the data. We simulated these fault for strings 2, 5, and 6, with onset times

$$T_2 = 12132, \quad T_5 = 16573, \quad T_6 = 6063,$$

and power reduction factors

$$f_2 = -7\%, \quad f_5 = -10\%, \quad f_6 = -12.5\%,$$

chosen randomly. (These are realistic values.) The modified data is shown in Figure 7.13, with vertical red lines indicating the onset of the outages in spower trings 2, 5, and 6. These power reductions can be seen in the plot, but would likely be hard to spot by eye.

SD problem formulation. We form an SD problem with $K = 5$ components. Our signal decomposition models string output as the product of a mean-square small residual (2.3), a clear sky signal, a common daily correction term, a common cloud/weather term, and a failure term. The clear sky component is modeled as the composite class that is smooth and periodic in time and close in entries (6.6) (with a small modification to remove the smoothness penalty across day boundaries). This component has two parameters, one for the

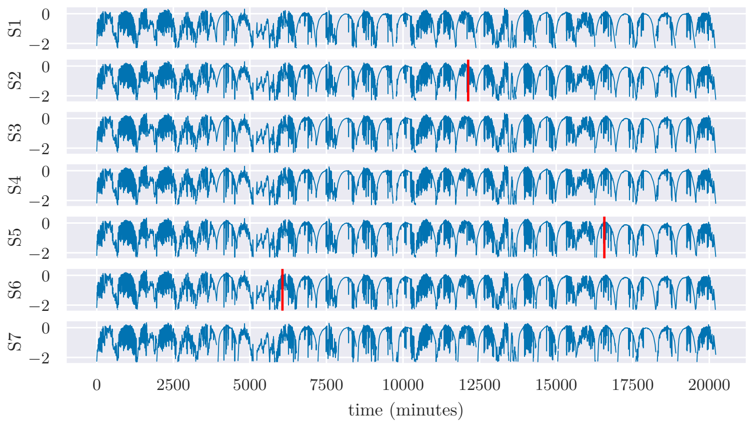

Figure 7.13: PV combiner box data after pre-processing, with simulated outages. The onset times of the simulated outages are shown as vertical red lines.

smoothness term and one for the variance across entries, λ_{2a} and λ_{2b}, respectively. The third component is a daily scale adjustment that is constant across columns, and constant over each day, meant to capture day-to-day macro-scale changes in atmospheric conditions that effect all strings, such as precipitable water and aerosol optical depth [45]. The fourth component is also constant across the columns and has a quantile loss function (6.2). This models a common cloud-loss term between the strings, assumed to be equal because the strings are so close to each other and are experiencing the same local weather. The fourth component has two parameters, the quantile term, τ, which we set to be 0.65, and a weight, λ_4. The third and fourth components make use of the common term formulation (5.4). The fifth component is the failure detector. This component uses the single jump class (6.7), constrained to only have negative jumps, with each column treated independently. The fifth component also has a weight parameter, λ_5. Since the failures are simulated, we know exactly when the onsets are, and what the values are, which we can compare to the estimated failure component.

Results. We solve the SD problem with hand-selected weights,

$$\lambda_{2a} = 5 \times 10^4/(Tp), \quad \lambda_{2b} = 5 \times 10^{-5}/(Tp),$$

$$\lambda_4 = 2/(Tp), \quad \lambda_5 = 10/(Tp),$$

giving us estimates of $x^1, \dots, x^5$. Our estimates of the components are $\widetilde{x^k} = \exp x^k$. We interpret $\widetilde{x}^1, \widetilde{x}^2, \widetilde{x}^4, \widetilde{x}^5$ as multiplicative components, and we interpret $\widetilde{x}^2$ as the baseline clear sky values, normalized. It takes approximately 15 seconds to run the SD-ADMM algorithm to convergence on a 2016 MacBook Pro, with no parallelization of the proximal operator evaluations. A segment of the decomposition is shown in Figure 7.14, highlighting 5 days of data for string 2, including the time of an estimated failure.

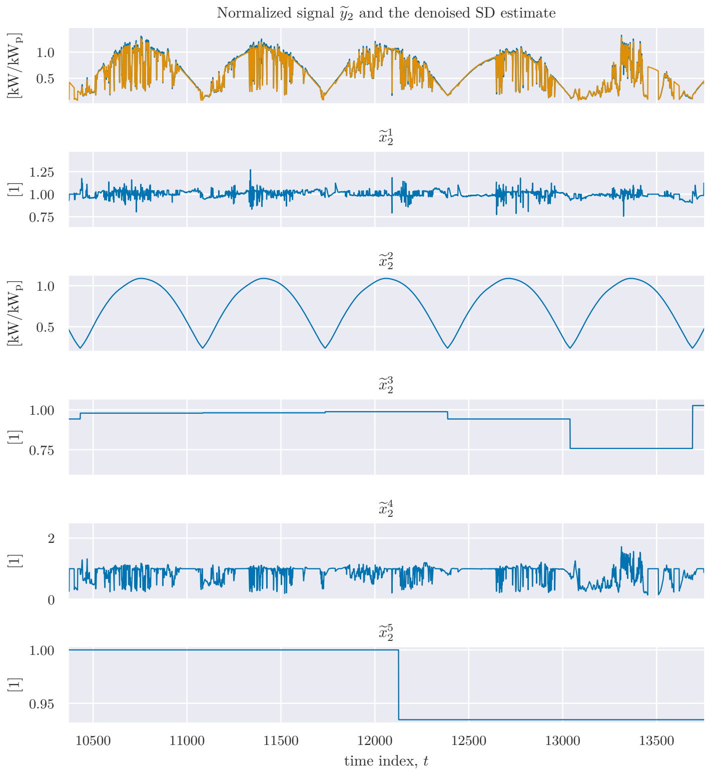

Figure 7.14: Components $\widetilde{x^k}$ for string 2 over 5 days.

The residual term $\widetilde{x}^1$ is shown as a histogram in Figure 7.15. The residual is centered at 1 and has a standard deviation of 0.082. 95% of the entries in the known set have residuals in the range of $[0.85, 1.15]$, *i.e.*, $\pm 15\%$.

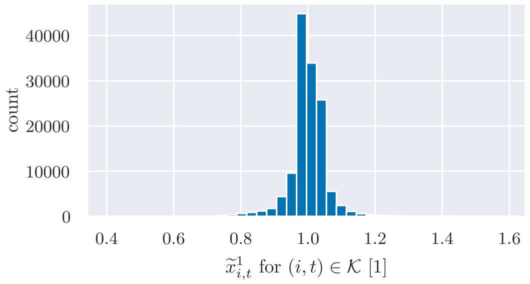

Figure 7.15: Histogram of the residual term $\widetilde{x}^1$ for all entries of the known set $\mathcal{K}$.

The clear sky component $\widetilde{x}^2$ is shown in Figure 7.16. We plot two days of this periodic component to illustrate the discontinuities in values between adjacent days. We see that the clear sky estimates for the strings are smooth in time, and vary a bit between strings.

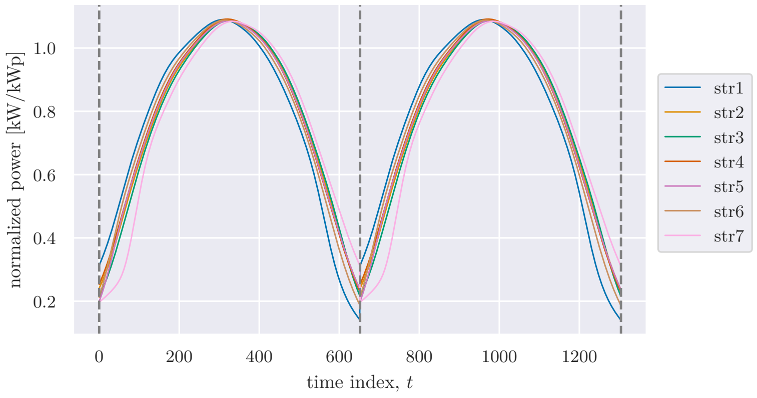

Figure 7.16: The clear sky component $\widetilde{x}^2$, with two days shown.

The common daily scale factor $\widetilde{x}^3$, shown in Figure 7.17(a), is constant across days and across columns. This can be thought of how much the clear sky signals need to be scaled to recreate any given day, and the all strings must agree on the factor. Days with significant cloud cover tend to have much smaller scale factors, while clearer days tend to vary by about 10–15%.

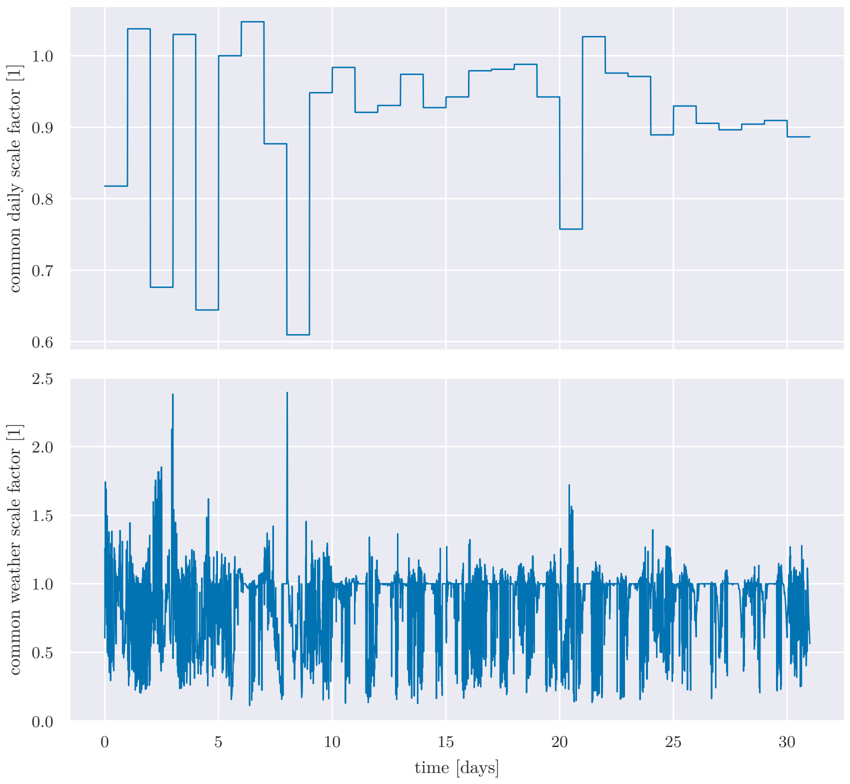

Figure 7.17: (a) Top, the common daily scale factor $\widetilde{x}^3$. (b) Bottom, the common weather component $\widetilde{x}^4$. Only one column of each component is plotted as all columns are equal to each other.

The common weather term $\widetilde{x}^4$, shown in Figure 7.17(b), is also constant across columns, and it captures the effects of local weather, particularly attenuation by clouds. This term is typically a loss, that is $\widetilde{x}^4 < 1$. We chose the value of the quantile parameter $\tau = 0.65$ through hand-tuning and selecting a value that gave good agreement between the clear sky component and the measured data on periods without significant cloud impacts. While having a weather correction term that is larger than about 1.5 does not make much physical sense (see, for example, [46]), we observe that this factor is applied to the combination of components 2 and 3, the clear sky component and the daily scale factor. In fact, we see the larger values in $\tilde{x}^4$ exactly on the days that are highly cloudy and use very small daily scale factors.

The failure component $\tilde{x}^5$ correctly identifies the failures correctly as appearing in only strings 2, 5, and 6, as depicted in Figure 7.18, which shows the predicted and real failure onset times and amounts. The estimated failure time is 5 minutes late for string 2, about 2 hours late for string 5, and exactly correct for string 6; for all three strings, the loss was detected within the same day that the failure occurred. We can also see that the estimated failure amounts are quite good. A comparison of actual to predicted failures is given in Table 7.1.

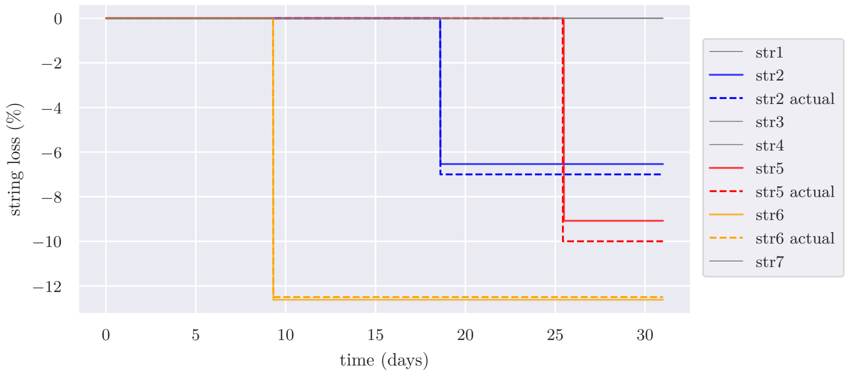

Figure 7.18: Failure component, shown as the percentage $100 \times (1 - \tilde{x}^5)\%$. The dashed lines show the actual simulated failures.

Table 7.1: Outage detection results

string	metric	actual	predicted
2	amount (%)	-7	-6.24
2	time (days)	18.60	18.61
5	amount (%)	-10	-9.10
5	time (days)	25.42	25.60
6	amount (%)	-12.5	-12.44
6	time (days)	9.30	9.30

Acknowledgements

This material is based on 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 38529. Stephen Boyd's work was funded in part by the AI Chip Center for Emerging Smart Systems (ACCESS).

The authors thank Joel Tropp for useful suggestions on an early draft of this monograph.

Appendix

SD-ADMM Algorithm Derivation

To derive an ADMM algorithm for SD, we introduce new variables $z^1, \ldots, z^K \in \mathbf{R}^q$ and reformulate the SD problem (2.7) as

$$\begin{array}{ll} \text{minimize} & \phi_1(x^1) + \cdots + \phi_K(x^K) \\ \text{subject to} & \mathcal{M}x^k - z^k = 0, \quad k = 1, \ldots, K \\ & \mathcal{M}y = z^1 + \cdots + z^K. \end{array}$$

We let g denote the indicator function of the last constraint,

$$g(z^1, \ldots, z^K) = \begin{cases} 0 & \mathcal{M}y = z^1 + \cdots + z^K \\ \infty & \text{otherwise,} \end{cases}$$

so the SD problem can be expressed as

$$\begin{array}{ll} \text{minimize} & \phi_1(x^1) + \cdots + \phi_K(x^K) + g(z^1, \ldots, z^K) \\ \text{subject to} & \mathcal{M}x^k - z^k = 0, \quad k = 1, \ldots, K. \end{array} \tag{2}$$

We write this in compact form as

$$\begin{array}{ll} \text{minimize} & \phi(x) + g(z) \\ \text{subject to} & \mathcal{M}x^k - z^k = 0, \quad k = 1, \ldots, K, \end{array} \tag{3}$$

where $x = (x^1, \ldots, x^K)$, $z = (z^1, \ldots, z^K)$, and $\phi(x) = \phi_1(x^1) + \cdots + \phi_K(x^K)$. We are now ready to derive the ADMM algorithm.

We form the augmented Lagrangian, with parameter $\rho > 0$,

$$\begin{aligned} L_\rho(x,z,\lambda) &= \phi(x) + g(z) + \sum_{k=1}^K \left({\lambda^k}^T(\mathcal{M}x^k - z^k) + (\rho/2)\|\mathcal{M}x^k - z^k\|_2^2 \right) \\ &= \phi(x) + g(z) + (\rho/2)\sum_{k=1}^K \left(\|r^k + u^k\|_2^2 - \|u^k\|_2^2 \right), \end{aligned}$$

where $r^k = \mathcal{M}x^k - z^k$ are the residuals, λ^k are the dual variables, and $u^k = (1/\rho)\lambda^k$ are the so-called scaled dual variables [20, §3.1.1].

Iteration j of ADMM consists of three steps:

$$\begin{aligned} x^{j+1} &= \operatorname*{argmin}_x L_\rho(x^j, z^j, u^j) \\ z^{j+1} &= \operatorname*{argmin}_z L_\rho(x^{j+1}, z^j, u^j) \\ (u^k)^{j+1} &= (u^k)^j + \mathcal{M}(x^k)^{j+1} - (z^k)^{j+1}, \quad k = 1,\dots,K, \end{aligned}$$

which we refer to as the x-update, z-update, and u-update, respectively.

We now work out and simplify these steps. Since L_ρ is separable in x^k, we can minimize over x^k separately in the x-update to obtain

$$(x^k)^{j+1} = \operatorname*{argmin}_{x^k} \left(\phi_k(x^k) + (\rho/2)\|\mathcal{M}x^k - (z^k)^j + (u^k)^j\|_2^2 \right) \quad k = 1,\dots,K. \quad (4)$$

The z-update can be written as

$$z^{j+1} = \Pi(\mathcal{M}(x^1)^{j+1} + (u^1)^j, \dots, \mathcal{M}(x^K)^{j+1} + (u^K)^j),$$

where Π is the projection onto the domain of g, *i.e.*, the constraints $\mathcal{M}y = z^1 + \cdots + z^k$. To simplify notation, let $a^k = \mathcal{M}(x^k)^{j+1} + (u^k)^j$. The z-update can be written as

$$\begin{aligned} (z^k)^{j+1} &= a^k + (1/K)(\mathcal{M}y - a^1 - \cdots - a^K) \\ &= \mathcal{M}(x^k)^{j+1} + (u^k)^j + (1/K)(\mathcal{M}y - a^1 - \cdots - a^K). \end{aligned}$$

Now consider the u-update. Plugging in the new z-update above, we get

$$(u^k)^{j+1} = -(1/K)(\mathcal{M}y - a^1 - \cdots - a^K).$$

The righthand side does not depend on k, which means that all $(u^k)^{j+1}$ are the same and can be denoted as u^{j+1}. (This simplification is not unexpected since the original problem has only one dual variable, which is a vector in $\mathbf{R}^q$.) With this simplification, the u-update (now for just one scaled dual variable $u \in \mathbf{R}^q$) becomes

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

Substituting u^j for $(u^k)^j$ in the z-update, we get

$$(z^k)^{j+1} = \mathcal{M}(x^k)^{j+1} - u^{j+1}.$$

Substituting $(z^k)^j = \mathcal{M}(x^k)^j - u^j$ into the original x-update (4) above, we obtain

$$\begin{array}{rcl} (x^k)^{j+1} & = & \underset{x^k}{\operatorname{argmin}} \left(\phi_k(x^k) + (\rho/2)\|\mathcal{M}x^k - \mathcal{M}(x^k)^j + 2u^j\|_2^2\right) \\ & = & \underset{x^k}{\operatorname{argmin}} \left(\phi_k(x^k) + (\rho/2)\|\mathcal{M}(x^k - (x^k)^j + 2\mathcal{M}^* u^j)\|_2^2\right) \\ & = & \mathbf{mprox}_{\phi_k}((x^k)^j - 2\mathcal{M}^* u^j), \end{array}$$

for $k = 1, \ldots, K$. (We use (4.1) in the second line.)

We now see that the variables z^k have dropped out, and we arrive at the final set of ADMM iterations

$$\begin{array}{rcl} (x^k)^{j+1} & = & \mathbf{mprox}_{\phi_k}((x^k)^j - 2\mathcal{M}^* u^j), \quad k = 1, \ldots, K \\ u^{j+1} & = & u^j + \dfrac{1}{K}\left(\displaystyle\sum_{k=1}^{K} \mathcal{M}(x^k)^{j+1} - \mathcal{M}y\right). \end{array}$$

References

[1] 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, 2018, pp. 42–60.

[2] D. Amelunxen, M. Lotz, M. B. McCoy, and J. A. Tropp, "Living on the edge: Phase transitions in convex programs with random data", *Information and Inference*, vol. 3, no. 3, 2014, pp. 224–294. DOI: 10.1093/imaiai/iau005.

[3] O. Anderson, "On the logic of the decomposition of statistical series into separate components", *Journal of the Royal Statistical Society*, vol. 90, no. 3, 1927, pp. 548–569. DOI: 10.2307/2341204.

[4] L. Avila and J. Cangialosi, "Tropical cyclone report for hurricane Irene (AL092011)", *NOAA National Hurricane Center*, 2013, URL: https://www.nhc.noaa.gov/data/tcr/AL092011_Irene.pdf.

[5] F. Bach, "Structured sparsity-inducing norms through submodular functions", in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23, Curran Associates, Inc., 2010. DOI: 10.48550/arXiv.1008.4220.

[6] R. Baillie and S.-K. Chung, "Modeling and forecasting from trend-stationary long memory models with applications to climatology", *International Journal of Forecasting*, vol. 18, no. 2, 2002, pp. 215–226. DOI: 10.1016/S0169-2070(01)00154-6.

[7] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing", *IEEE Transactions on Information Theory*, vol. 56, no. 4, 2010, pp. 1982–2001. DOI: 10.1109/TIT.2010.2040894.

[8] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods", *SIAM Journal on Optimization*, vol. 23, no. 4, 2013, pp. 2037–2060. DOI: 10.1137/120887679.

[9] D. Bertsekas, *Nonlinear Programming: Third Edition*. Nashua, NH: Athena Scientific, 2016.

[10] M. Best and N. Chakravarti, "Active set algorithms for isotonic regression; a unifying framework", *Mathematical Programming*, vol. 47, 1990, pp. 425–439. DOI: 10.1007/BF01580873.

[11] "February 9–10, 2010 North American blizzard", *Wikipedia*, URL: https://en.wikipedia.org/w/index.php?title=February_9%E2%80%9310,_2010_North_American_blizzard&oldid=1030398339.

[12] "December 26–27th 2010 blizzard", *National Weather Service*, URL: https://www.weather.gov/okx/storm12262010.

[13] "January 26–27 2015 blizzard", *National Weather Service*, URL: https://www.weather.gov/okx/Blizzard_01262715.

[14] "January 31–February 2 2021 winter storm", *National Weather Service*, URL: https://www.weather.gov/okx/WinterStormJan31_Feb22021.

[15] P. Bloomfield, "Trends in global temperature", *Climatic Change*, vol. 21, no. 1, 1992, pp. 1–16. DOI: 10.1007/BF00143250.

[16] P. Bloomfield and D. Nychka, "Climate spectra and detecting climate change", *Climatic Change*, vol. 21, no. 3, 1992, pp. 275–287. DOI: 10.1007/BF00139727.

[17] C. Bouman and K. Sauer, "A generalized Gaussian image model for edge-preserving MAP estimation", *IEEE Transactions on Image Processing*, vol. 2, no. 3, 1993, pp. 296–310. DOI: 10.1109/83.236536.

[18] M. Boyd, "High-speed monitoring of multiple grid-connected photovoltaic array configurations", *NIST Technical Note 1896*, 2015. DOI: 10.6028/NIST.TN.1896.

[19] M. Boyd, T. Chen, and B. Doughert, *NIST Campus Photovoltaic (PV) Arrays and Weather Station Data Sets*. National Institute of Standards and Technology, 2017. DOI: 10.18434/M3S67G.

[20] 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, 2011, pp. 1–122. DOI: 10.1561/2200000016.

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

[22] S. Boyd and L. Vandenberghe, *Introduction to Applied Linear Algebra*. Cambridge university press, 2018.

[23] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?", *Journal of the ACM*, vol. 58, no. 3, 2011, pp. 1–37. DOI: 10.1145/1970392.1970395.

[24] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization", *Foundations of Computational Mathematics*, vol. 9, no. 6, 2009, pp. 717–772. DOI: 10.1007/s10208-009-9045-5.

[25] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky, "The convex geometry of linear inverse problems", *Foundations of Computational Mathematics*, vol. 12, no. 6, 2010, pp. 805–849. DOI: 10.1007/s10208-012-9135-7.

[26] J. F. Claerbout and F. Muir, "Robust modeling with erratic data", *Geophysics*, vol. 38, no. 5, 1973, pp. 826–844. DOI: 10.1190/1.1440378.

[27] R. Cleveland, W. Cleveland, J. McRae, and I. Terpenning, "STL: A seasonal-trend decomposition procedure based on loess", *Journal of Official Statistics*, vol. 6, no. 1, 1990, pp. 3–73, URL: https://www.proquest.com/scholarly-journals/stl-seasonal-trend-decomposition-procedure-based/docview/1266805989/se-2.

[28] P. Combettes and J. Pesquet, "Proximal splitting methods in signal processing", *Springer Optimization and Its Applications*, vol. 49, 2011, pp. 185–212. DOI: 10.1007/978-1-4419-9569-8_10.

[29] A. Cuomo, "Governor Cuomo signs the 'New York State on PAUSE' executive order, March 20, 2020", *New York Governor's Press Office*, URL: https://web.archive.org/web/20200328191630/https://www.governor.ny.gov/news/governor-cuomo-signs-new-york-state-pause-executive-order.

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

[31] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition", *IEEE Transactions on Information Theory*, vol. 47, no. 7, 2001, pp. 2845–2862. DOI: 10.1109/18.959265.

[32] R. Farebrother, "Adrien-Marie Legendre", in *Statisticians of the Centuries*, C. Heyde, E. Seneta, P. Crépel, S. Fienberg, and J. Gani, Eds., New York, NY: Springer, 2001, pp. 101–104. DOI: 10.1007/978-1-4613-0179-0_20.

[33] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, 1981, pp. 381–395. DOI: 10.1145/358669.358692.

[34] E. Franklin, "Solar photovoltaic (PV) system components", *The University of Arizona College of Agriculture & Life Sciences*, 2018, pp. 1–8, URL: https://projects.sare.org/wp-content/uploads/az1742-2017.pdf.

[35] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation", *Computers and Mathematics with Applications*, vol. 2, no. 1, 1976, pp. 17–40. DOI: 10.1016/0898-1221(76)90003-1.

[36] R. Glowinski and A. Marroco, "Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualité, d'une classe de problems de Dirichlet non lineares", *Revue Française d'Automatique, Informatique, et Recherche Opérationelle*, vol. 9, no. R-2, 1975, pp. 41–76. DOI: 10.1051/m2an/197509R200411.

[37] S. Greenland and M. Longnecker, "Methods for trend estimation from summarized dose-response data, with applications to meta-analysis", *American Journal of Epidemiology*, vol. 135, no. 11, 1992, pp. 1301–1309. DOI: 10.1093/oxfordjournals.aje.a116237.

[38] S. Grotzinger and C. Witzgall, "Projections onto order simplexes", *Applied Mathematics and Optimization*, vol. 12, no. 1, 1984, pp. 247–270. DOI: 10.1007/BF01449044.

[39] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Science & Business Media, 2013.

[40] R. Hodrick and E. Prescott, "Postwar U.S. business cycles: An empirical investigation", *Journal of Money, Credit and Banking*, vol. 29, no. 1, 1997, pp. 1–16. DOI: 10.2307/2953682.

[41] A. Hoerl and R. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems", *Technometrics*, vol. 12, no. 1, 1970, pp. 55–67. DOI: 10.1080/00401706.1970.10488634.

[42] P. Huber, "Robust estimation of a location parameter", *The Annals of Mathematical Statistics*, vol. 35, no. 1, 1964, pp. 73–101. DOI: 10.1214/aoms/1177703732.

[43] P. Huber, *Robust statistics*, vol. 523. Hoboken, NJ: John Wiley & Sons, 1981. DOI: 10.1002/0471725250.

[44] R. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice.* OTexts: Melbourne, Australia, 2018.

[45] P. Ineichen, "A broadband simplified version of the Solis clear sky model", *Solar Energy*, vol. 82, no. 8, 2008, pp. 758–762. DOI: 10.1016/j.solener.2008.02.009.

[46] R. Inman, Y. Chu, and C. Coimbra, "Cloud enhancement of global horizontal irradiance in California and Hawaii", *Solar Energy*, vol. 130, 2016, pp. 128–138. DOI: 10.1016/j.solener.2016.02.011.

[47] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "L1 trend filtering", *SIAM Review*, vol. 51, no. 2, 2009, pp. 339–360. DOI: 10.1137/070690274.

[48] R. Koenker and G. Bassett, "Regression quantiles", *Econometrica*, vol. 46, no. 1, 1978, p. 33. DOI: 10.2307/1913643.

[49] R. Koenker and K. F. Hallock, "Quantile regression", *Journal of Economic Perspectives*, vol. 15, no. 4, 2001, pp. 143–156. DOI: 10.1257/jep.15.4.143.

[50] C. Leser, "A simple method of trend construction", *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 23, no. 1, 1961, pp. 91–107. DOI: 10.1111/j.2517-6161.1961.tb00393.x.

[51] S. Levitt, “Understanding why crime fell in the 1990s: Four factors that explain the decline and six that do not”, *Journal of Economic Perspectives*, vol. 18, no. 1, 2004, pp. 163–190. DOI: 10.1257/089533004773563485.

[52] W. Link and F. Sauer, “Estimating equations estimates of trends”, *Bird Populations*, vol. 2, 1994, pp. 23–32, URL: https://bmeyers.github.io/assets/link-1994.pdf.

[53] S. Mallat, *A Wavelet Tour of Signal Processing.* Elsevier, 2009, pp. 20–41. DOI: 10.1016/B978-0-12-374370-1.X0001-8.

[54] M. B. McCoy, V. Cevher, Q. T. Dinh, A. Asaei, and L. Baldassarre, “Convexity in source separation: Models, geometry, and algorithms”, *IEEE Signal Processing Magazine*, vol. 31, no. 3, 2014, pp. 87–95. DOI: 10.1109/MSP.2013.2296605.

[55] M. B. McCoy and J. A. Tropp, “Sharp recovery bounds for convex demixing, with applications”, *Foundations of Computational Mathematics*, vol. 14, no. 3, 2014, pp. 503–567. DOI: 10.1007/s10208-014-9191-2.

[56] J.-J. Moreau, “Fonctions convexes duales et points proximaux dans un espace hilbertien”, *Reports of the Paris Academy of Sciences, Series A*, vol. 255, 1962, pp. 2897–2899.

[57] “Hourly traffic on Metropolitan Transportation Authority (MTA) bridges and tunnels”, *NY Open Data*, URL: https://data.ny.gov/Transportation/Hourly-Traffic-on-Metropolitan-Transportation-Auth/qzve-kjga.

[58] O. Neugebauer, *The Exact Sciences in Antiquity*, ser. Acta historica scientiarum naturalium et medicinalium. Dover Publications, 1969, URL: https://books.google.com/books?id=JVhTtVA2zr8C.

[59] A. Oppenheim and R. Schafer, *Discrete-time Signal Processing*, ser. Prentice-Hall signal processing series. United Kingdom: Pearson, 2010.

[60] D. Osborn, “Moving average detrending and the analysis of business cycles”, *Oxford Bulletin of Economics and Statistics*, vol. 57, no. 4, 1995, pp. 547–558. DOI: 10.1111/j.1468-0084.1995.tb00039.x.

[61] N. Parikh and S. Boyd, “Proximal algorithms”, *Foundations and Trends in Optimization*, vol. 1, no. 3, 2014, pp. 127–239. DOI: 10.1561/2400000003.

[62] 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, 2011, pp. 2825–2830, URL: https://jmlr.org/papers/v12/pedregosa11a.html.

[63] D. Phillips, "A technique for the numerical solution of certain integral equations of the first kind", *Journal of the ACM*, vol. 9, no. 1, 1962, pp. 84–97. DOI: 10.1145/321105.321114.

[64] R. Poliquin and R. Rockafellar, "Prox-regular functions in variational analysis", *Transactions of the American Mathematical Society*, vol. 348, no. 5, 1996, pp. 1805–1838. DOI: 10.1090/S0002-9947-96-01544-9.

[65] P. Richtárik and M. Takáč, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function", *Mathematical Programming*, vol. 144, no. 1-2, 2014, pp. 1–38. DOI: 10.1007/s10107-012-0614-z.

[66] R. Rockafellar, *Convex Analysis*. Princeton University Press, 1970.

[67] R. Sah, "Caltrans traffic census program", *California Department of Transportation*, URL: https://dot.ca.gov/programs/traffic-operations/census.

[68] P. Sen, "Estimates of the regression coefficient based on Kendall's Tau", *Journal of the American Statistical Association*, vol. 63, no. 324, 1968, pp. 1379–1389. DOI: 10.1080/01621459.1968.10480934.

[69] K. Singleton, "Econometric issues in the analysis of equilibrium business cycle models", *Journal of Monetary Economics*, vol. 21, no. 2-3, 1988, pp. 361–386. DOI: 10.1016/0304-3932(88)90036-0.

[70] J.-L. Starck, F. Murtagh, and J. M. Fadili, *Sparse image and signal processing: wavelets, curvelets, morphological diversity*. Cambridge, UK: Cambridge university press, 2010. DOI: 10.1017/CBO9780511730344.

[71] J. Perktold, S. Seabold, and J. Taylor, "Seasonal-trend decomposition using LOESS (STL)", *Statsmodels documentation*, URL: https://www.statsmodels.org/dev/examples/notebooks/generated/stl_decomposition.html.

[72] "STL: Seasonal decomposition of time series by loess", *R documentation*, URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/stl.

[73] "Time series decomposition", *MathWorks documentation*, URL: https://www.mathworks.com/help/econ/detrending.html.

[74] P. Tans and R. Keeling, "Mauna Loa CO2 weekly mean and historical comparisons", *NOAA Global Monitoring Laboratory, Earth System Research Laboratories*, URL: https://gml.noaa.gov/ccgg/trends/data.html.

[75] H. L. Taylor, S. C. Banks, and J. F. McCoy, "Deconvolution with the l1 norm", *Geophysics*, vol. 44, no. 1, 1979, pp. 39–52. DOI: 10.1190/1.1440921.

[76] H. Theil, "A rank-invariant method of linear and polynomial regression analysis", *Proceedings of the Royal Netherlands Academy of Sciences*, vol. 53, 1950, Part I: 386–392, Part II: 521–525, Part III: 1397–1412. DOI: 10.1007/978-94-011-2546-8_20.

[77] A. Thompson and J. Kay, "On some Bayesian choices of regularization parameter in image restoration", *Inverse Problems*, vol. 9, no. 6, 1993, pp. 749–761. DOI: 10.1088/0266-5611/9/6/011.

[78] R. Tibshirani, "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, 1996, pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[79] A. Tikonov, "Solution of incorrectly formulated problems and the regularization method", *Soviet Math.*, vol. 4, 1963, pp. 1035–1038.

[80] D. Titterington, "General structure of regularization procedures in image reconstruction", *Astronomy and Astrophysics*, vol. 144, no. 2, 1985, pp. 381–387.

[81] I. Tošić and P. Frossard, "Dictionary learning", *IEEE Signal Processing Magazine*, vol. 28, no. 2, 2011, pp. 27–38. DOI: 10.1109/MSP.2010.939537.

[82] R. Tsay, *Analysis of Financial Time Series*, ser. Wiley Series in Probability and Statistics. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005, pp. 206–215. DOI: 10.1002/0471746193.

[83] M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models", *Foundations and Trends in Machine Learning*, vol. 9, no. 1, 2016, pp. 1–118. DOI: 10.1561/2200000055.

[84] J. Wright and Y. Ma, *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications*. New York, NY: Cambridge University Press, 2022.

[85] S. Wright, "Coordinate descent algorithms", *Mathematical Programming*, vol. 151, no. 1, 2015, pp. 3–34. DOI: 10.1007/s10107-015-0892-3.

[86] W. B. Wu, M. Woodroofe, and G. Mentz, "Isotonic regression: Another look at the changepoint problem", *Biometrika*, vol. 88, no. 3, 2001, pp. 793–804. DOI: 10.1093/biomet/88.3.793.

[87] M. Wytock and J. Kolter, "Contextually supervised source separation with application to energy disaggregation", *Twenty-Eighth AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2013, pp. 486–492. DOI: 10.1609/aaai.v28i1.8769.

[88] A. Yurtsever, V. M., and S. Sra, "Three operator splitting with a nonconvex loss function", 2021. DOI: 10.48550/arXiv.2103.04568.

[89] K. Zhao, M. A. Wulder, T. Hu, R. Bright, Q. Wu, H. Qin, Y. Li, E. Toman, B. Mallick, X. Zhang, and M. Brown, "Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm", *Remote Sensing of Environment*, vol. 232, 2019, pp. 111–181. DOI: 10.1016/j.rse.2019.04.034.

[90] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net", *Journal of the royal statistical society, series B (statistical methodology)*, vol. 67, no. 2, 2005, pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x.