A Block Coordinate Descent Proximal Method for Simultaneous Filtering and Parameter Estimation

Ramin Raziperchikolaei [1 2] **Harish S. Bhat** [3 4]

Abstract

We propose and analyze a block coordinate descent proximal algorithm (BCD-prox) for simultaneous filtering and parameter estimation of ODE models. As we show on ODE systems with up to $d = 40$ dimensions, as compared to state-of-the-art methods, BCD-prox exhibits increased robustness (to noise, parameter initialization, and hyperparameters), decreased training times, and improved accuracy of both filtered states and estimated parameters. We show how BCD-prox can be used with multistep numerical discretizations, and we establish convergence of BCD-prox under hypotheses that include real systems of interest.

1. Introduction

Though ordinary differential equations (ODE) are used extensively in science and engineering, the task of learning ODE states and parameters from data still presents challenges. This is especially true for nonlinear ODE that do not have analytical solutions. For such problems, several published and widely used methods—including Bayesian, spline-based, and extended Kalman filter methods—work well with data with a high signal-to-noise ratio. As the magnitude of noise increases, these methods break down, leading to unreliable estimates of states and parameters. Problem domains such as biology commonly feature both nonlinear ODE models and highly noisy observations, motivating the present work.

Motivated by recent advances in alternating minimization (Chatterji & Bartlett, 2017; Li et al., 2016; Yi et al., 2014), block coordinate descent (BCD) (Xu & Yin, 2013; Zhang & Brand, 2017), and proximal methods (Parikh & Boyd, 2014; Sun et al., 2015), we study a BCD proximal algorithm (BCD-prox) to solve the simultaneous filtering and parameter estimation problem. Here filtering means recovering clean ODE states from noisy observations. BCD-prox works by minimizing a unified objective function that directly measures how well the states and parameters satisfy the ODE system, in contrast to other methods that use separate objectives. BCD-prox learns the states directly in the original space, instead of learning them indirectly by fitting a smoothed function to the observations. Under hypotheses that include systems of real interest, BCD-prox is provably convergent. In comparison with other methods, BCD-prox is more robust with respect to noise, parameter initialization, and hyperparameters. BCD-prox is also easy to implement and runs quickly.

There have been several different approaches to the filtering and estimation problem. Nonlinear least squares methods start with an initial guess for the parameters that is iteratively updated to bring the model's predictions close to measurements (Bard, 1973; Benson, 1979; Himmelblau et al., 1967; Hosten, 1979). These methods diverge when the initial parameters are far from the true parameters.

Of more recent interest are spline-based methods, in which filtered, clean states are computed via (cubic) splines fit to noisy data. As splines are differentiable, parameter estimation then reduces to a regression problem (Cao & Zhao, 2008; Cao et al., 2011; Poyton et al., 2006; Ramsay et al., 2007; Varah, 1982). Estimators other than splines, such as smoothing kernels and local polynomials, are also used (Dattner & Klaassen, 2015; Gugushvili & Klaassen, 2012; Liang & Wu, 2008). These methods are sensitive to numerous hyperparameters (such as smoothing parameters and the numbers/positions of knots), to parameter initialization, and to the magnitude/type of noise that contaminates the data.

Bayesian approaches (Calderhead et al., 2009; Dondelinger et al., 2013; Girolami, 2008; Gorbach et al., 2017) must set hyperparameters (prior distributions, variances, kernel widths, etc.) very carefully to produce reasonable results. Bayesian methods also feature large training times. Another disadvantage of these methods, mentioned by Gorbach et al.

[1]Rakuten Institute of Technology, San Mateo, CA, USA [2]Department of Computer Science, University of California, Merced, USA [3]Department of Mathematics, University of Utah, USA [4]Department of Applied Mathematics, University of California, Merced, USA. Correspondence to: Ramin Raziperchikolaei <ramin.raziperchikola@rakuten.com>, Harish S. Bhat <hbhat@ucmerced.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

(2017), is that they cannot simultaneously learn clean states and parameters. Gorbach et al. (2017) uses a variational inference approach to overcome this problem, but the method is not applicable to all ODE.

BCD-prox learns parameters and states jointly, but it does not fit a smooth function to the observations. Via this approach, BCD-prox reduces the number of hyperparameters to one. BCD-prox avoids assumptions (i.e., spline or other smooth estimator) regarding the shape of the filtered states. Furthermore, both the BCD and proximal components of the algorithm enable it to step slowly away from a poor initial choice of parameters. In this way, BCD-prox remedies the problems of other methods.

Many other well-known nonlinear ODE filtering methods, including extended and ensemble Kalman filters as well as particle filters, are online methods that make Gaussian assumptions. In contrast, BCD-prox is a distribution-free, batch method.

The rest of the paper is organized as follows. In Section 2 we define both the problem and the BCD-prox algorithm. In Section 3, we compare BCD-prox at a conceptual level against a competing method from the literature. We discuss the convergence of BCD-prox in Section 4. We show the advantages of BCD-prox with several experiments in Section 5. Further experiments and details are given in the supplementary material.

2. Problem and Proposed Solution

Consider a dynamical system in $\mathbb{R}^d$, depending on a parameter $\boldsymbol{\theta} \in \mathbb{R}^p$, with state $\mathbf{x}(t)$ at time t:

$$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}). \tag{1}$$

At T distinct times $\{t_i\}_{i=1}^T$, we have noisy observations $\mathbf{y}(t_i) \in \mathbb{R}^d$:

$$\mathbf{y}(t_i) = \mathbf{x}(t_i) + \mathbf{z}(t_i), \quad i = 1, \ldots, T \tag{2}$$

where $\mathbf{z}(t_i) \in \mathbb{R}^d$ is the noise of the observation at time t_i. We represent the set of T d-dimensional states, noises, and observations by $\mathbf{X}, \mathbf{Z}$, and $\mathbf{Y} \in \mathbb{R}^{d \times T}$, respectively. For concision, in what follows, we write the time t_i as a subscript, i.e., $\mathbf{x}_{(t_i)}$ instead of $\mathbf{x}(t_i)$.

In this paper, we assume that the form of the vector field $\mathbf{f}(\cdot)$ is known. The simultaneous parameter estimation and filtering problem is to use $\mathbf{Y}$ to estimate $\boldsymbol{\theta}$ and $\mathbf{X}$. Examples of $\mathbf{f}(\cdot)$ and $\boldsymbol{\theta}$ can be found in Section 5.

For ease of exposition, we first describe a BCD-prox algorithm based on the explicit Euler discretization of (1). Later, we will describe how to incorporate higher-order multistep methods into BCD-prox. The explicit Euler method discretizes the ODE (1) for the T time points as follows:

$$\mathbf{x}_{(t_{i+1})} - \mathbf{x}_{(t_i)} = \mathbf{f}(\mathbf{x}_{(t_i)}, \boldsymbol{\theta})\Delta_i, \quad i = 1, ..., T-1 \tag{3}$$

where $\Delta_i = t_{i+1} - t_i$. In (3), both states $\mathbf{X}$ and parameters $\boldsymbol{\theta}$ are unknown; we are given only the noisy observations $\mathbf{Y}$. With this discretization, let us define

$$E(\mathbf{X}, \boldsymbol{\theta}) = \sum_{i=1}^{T-1} \left\| \mathbf{x}_{(t_{i+1})} - \mathbf{x}_{(t_i)} - \mathbf{f}(\mathbf{x}_{(t_i)}, \boldsymbol{\theta})\Delta_i \right\|^2 \tag{4}$$

Note that E measures the time-discretized mismatch between the left- and right-hand sides of (1). We refer to E as fidelity, the degree to which the estimated states $\mathbf{X}$ and parameters $\boldsymbol{\theta}$ actually satisfy the ODE. Let us now envision a sequence of iterates $\{\mathbf{X}^{*(n)}, \boldsymbol{\theta}^{*(n)}\}_{n \geq 0}$. For $n \geq 1$, we define the Euler BCD-prox objective function:

$$F_n^{\text{Euler}}(\mathbf{X}, \boldsymbol{\theta}) = E(\mathbf{X}, \boldsymbol{\theta}) + \lambda \left\| \mathbf{X} - \mathbf{X}^{*(n-1)} \right\|^2. \tag{5}$$

We can now succinctly describe the Euler version of BCD-prox as block coordinate descent (first on $\boldsymbol{\theta}$, then on $\mathbf{X}$) applied to (5)*, initialized with the noisy data via $\mathbf{X}^{*(0)} = \mathbf{Y}$, and repeated iteratively until convergence criteria are met..*

The Euler method is a first-order method. To understand this, let $\Delta_i = h$ (independent of i) and $t_N = Nh$. Then the global error between the numerical and true solution of the ODE (1), $\left\| \mathbf{x}^{\text{numerical}}(t_N) - \mathbf{x}^{\text{true}}(t_N) \right\|$, is $O(h)$. If we seek a more accurate discretization, we can apply a multistep method. The idea of multistep (m-step) methods is to use the previous m states to predict the next state, yielding a method with $O(h^m)$ global error. Let us consider the general formulation of the explicit linear m-step method to discretize (1):

$$\mathbf{x}_{(t_{i+1})} = \sum_{j=0}^{m-1} a_j \mathbf{x}_{(t_{i-j})} + \Delta_i \sum_{j=0}^{m-1} b_j \mathbf{f}(\mathbf{x}_{(i-j)}, \boldsymbol{\theta}), \tag{6}$$

where Δ_i is the time step. There are several strategies to determine the coefficients $\{a_j\}_{j=0}^{m-1}$ and $\{b_j\}_{j=0}^{m-1}$. For example, over the interval Δ_i, the Adams-Bashforth method approximates $\mathbf{f}(\cdot)$ with a polynomial of order m; this leads to a method with $O(h^m)$ global error. When $m = 1$, this method reduces to the explicit Euler method considered above. For further information on multistep methods, consult Iserles (2009); Palais & Palais (2009).

Note that to use m-step methods to predict the state at time i, we need its previous m states. To predict the states $\{\mathbf{x}_i\}_{i=2}^m$ (the first few states), the maximum order we can use is $i-1$, because there are only $i-1$ states before the state $\mathbf{x}_i$. In general, to predict $\mathbf{x}_i$ we use a multistep method of order $\min(i-1, m)$.

Algorithm 1 Pseudo-code of our proposed method

Input: Noisy observations $\mathbf{Y} = [\mathbf{y}_{(t_1)}, \ldots, \mathbf{y}_{(t_T)}] \in \mathbb{R}^{d \times T}$, time differences $\{\Delta_i = t_{i+1} - t_i\}_{i=1}^{T-1}$, form of $\mathbf{f}(\cdot)$ in Eq. (1), hyperparameter λ, initial guess $\boldsymbol{\theta}^{*(0)}$, and order m of the m-step method.

1: $\mathbf{X}^{*(0)} = \mathbf{Y}$
2: $n = 0$
3: **repeat**
4: $\quad n = n + 1$
5: $\quad$ • Compute $\boldsymbol{\theta}^{*(n)} = \operatorname{argmin}_{\boldsymbol{\theta}} F_n(\mathbf{X}^{*(n-1)}, \boldsymbol{\theta})$.
6: $\quad$ • Compute $\mathbf{X}^{*(n)} = \operatorname{argmin}_{\mathbf{X}} F_n(\mathbf{X}, \boldsymbol{\theta}^{*(n)})$.
7: **until** convergence
8: Compute predicted states $\hat{\mathbf{X}}$ by repeatedly applying Eq. (6), where $\boldsymbol{\theta} = \boldsymbol{\theta}^{*(n)}$ and $\mathbf{x}_{(t_1)} = \mathbf{x}^{*(n)}_{(t_1)}$.
9: **return** $\boldsymbol{\theta}^{*(n)}$ and $\hat{\mathbf{X}}$ as the estimated parameters and predicted states.

When using a general m-step discretization method, we define our objective function as follows:

$$E_{\text{m-step}}(\mathbf{X}, \boldsymbol{\theta}) = \sum_{i=1}^{T-1} \Big\| \mathbf{x}_{(t_{i+1})} - \textstyle\sum_{j=0}^{k-1} a_j \mathbf{x}_{(t_{i-j})} - \Delta_i \sum_{j=0}^{k-1} b_j \mathbf{f}(\mathbf{x}_{(i-j)}, \boldsymbol{\theta}) \Big\|^2, \quad (7)$$

where $k = \min(i-1, m)$ is the order of the discretization method to predict the state $\mathbf{x}_i$. We can then reformulate the BCD-prox objective as

$$F_n(\mathbf{X}, \boldsymbol{\theta}) = E_{\text{m-step}}(\mathbf{X}, \boldsymbol{\theta}) + \lambda \left\| \mathbf{X} - \mathbf{X}^{*(n-1)} \right\|^2. \quad (8)$$

We now regard (5) as a special case of (8) for $m = 1$, i.e., in the case where the m-step method reduces to Euler. With these definitions, *BCD-prox is block coordinate descent (first on $\boldsymbol{\theta}$, then on $\mathbf{X}$) applied to $F_n(\mathbf{X}, \boldsymbol{\theta})$, initialized with the noisy data via $\mathbf{X}^{*(0)} = \mathbf{Y}$, and repeated iteratively until convergence criteria are met.*. We detail this algorithm in Alg. 1.

3. Conceptual Comparison with iPDA

Though BCD-prox may seem straightforward, we cannot find prior work that utilizes precisely this approach. Since of the closest relatives is the successful iPDA (iterated principal differential analysis) method (Poyton et al., 2006; Ramsay et al., 2007), we explain iPDA and offer a conceptual comparison between iPDA and BCD-prox. In iPDA, the parameter estimation error is defined as

$$E_{\text{cont}}(\mathbf{x}_{(t)}, \boldsymbol{\theta}) = \int \left\| \frac{d\mathbf{x}_{(t)}}{dt} - \mathbf{f}(\mathbf{x}_{(t)}, \boldsymbol{\theta}) \right\|^2 dt, \quad (9)$$

which we can regard as the continuous-time ($\Delta_i \to 0$) limit of either (4) or (7), our mismatch/fidelity terms. The iPDA objective function is then

$$J(\mathbf{x}_{(t)}, \boldsymbol{\theta}) = E_{\text{cont}}(\mathbf{x}_{(t)}, \boldsymbol{\theta}) + \lambda \left\| \mathbf{X} - \mathbf{Y} \right\|^2, \quad (10)$$

the sum of the parameter estimation error with a regularization term. Initialized with $\boldsymbol{\theta}^{(0)}$, the iPDA method proceeds by iterating over the following two minimization steps:

1. Set $\mathbf{x}^{(n)}_{(t)} = \operatorname{argmin}_{\mathbf{x}_{(t)}} J(\mathbf{x}_{(t)}, \boldsymbol{\theta}^{(n-1)})$. In this step, $\mathbf{x}_{(t)}$ is constrained to be a smooth spline.

2. Set $\boldsymbol{\theta}^{(n)} = \operatorname{argmin}_{\boldsymbol{\theta}} J(\mathbf{x}^{(n)}_{(t)}, \boldsymbol{\theta})$. Note that the optimization only includes the parameter estimation term since the regularization term does not depend on $\boldsymbol{\theta}$.

The main issue with the objective in (10) is the regularization term. This term determines how far the clean states are going to be from the noisy observations. If we set λ to a large value, then $\mathbf{x}_{(t)}$ remains close to the data $\mathbf{y}_{(t)}$, potentially causing a large parameter estimation error. If we set λ to a small value, then $\mathbf{x}_{(t)}$ might wander far from the observed data. It is a challenging task to set λ to the right value for two reasons: 1) the optimal λ depends on both the noise $\mathbf{Z}$ and the vector field $\mathbf{f}$, and 2) in a real problem, we do not have access to the clean states $\mathbf{X}$ (all we have are the noisy observations $\mathbf{Y}$), so we cannot find the right λ by cross-validation. We return to this point below.

Before continuing, it is worth pointing out a crucial fact regarding all the mismatch/fidelity objectives E that we have seen thus far.

Theorem 1. *The objective functions E defined in* (4), (7), *and* (9) *all have an infinite number of zeros, i.e., an infinite number of global minima that result in $E = 0$.*

Proof. Assign arbitrary real vectors to $\boldsymbol{\theta}$ and the initial condition $\mathbf{x}_{(t_1)}$. Note that (4) is the special case of (7) for $m = 1$ so we need only discuss (7). Starting from $\mathbf{x}_{(t_1)}$, step forward in time via (6). By computing the states $\mathbf{x}_{(t_2)}, \ldots, \mathbf{x}_{(t_T)}$ in this way, we ensure that each term in (7) vanishes. For the continuous E function (9), we use the existence/uniqueness theorem for ODE to posit a unique solution $\mathbf{x}_{(t)}$ passing through $\mathbf{x}_{(t_1)}$ at time $t = t_1$. By definition of a solution of an ODE, this will ensure that (9) vanishes. Because, in all cases, $E \geq 0$, we achieve a global minimum. Because $\boldsymbol{\theta}$ and $\mathbf{x}_{(t_1)}$ are arbitrary, an infinite number of such minima exist. □

Note that BCD-prox always produces a (state,parameter) pair that results in $E = 0$ for (7). In fact, step 8 of Alg. 1 uses the idea from the proof of Theorem (1) to generate a sequence of predicted states $\hat{\mathbf{X}}$ such that $E(\hat{\mathbf{X}}, \boldsymbol{\theta}^{*(n)}) = 0$.

Let us reconsider Step 6 in Alg. 1:

$$\mathbf{X}^{*(n)} = \underset{\mathbf{X}}{\operatorname{argmin}} F_n(\mathbf{X}, \boldsymbol{\theta}^{*(n)}). \tag{11}$$

By definition of F_n and using the notion of proximal operators (Parikh & Boyd, 2014), we can write

$$\mathbf{X}^{*(n)} = \operatorname{prox}_{(2\lambda)^{-1}E}(\mathbf{X}^{*(n-1)}),$$

with the understanding here and in what follows that $\boldsymbol{\theta}$ is fixed at $\boldsymbol{\theta}^{*(n)}$. In general for $\Delta_i > 0$ and arbitrary f, E defined in (7) will not be convex. In this case, we view the proximal operator above as a set-valued operator as in (Li et al., 2017); any element of the set will do. Conceptually, this proximal step approximates a gradient descent step:

$$\begin{aligned}\mathbf{X}^{*(n)} &= \mathbf{X}^{*(n-1)} \\ &\quad - (2\lambda)^{-1}\nabla_{\mathbf{X}} E(\mathbf{X}^{*(n-1)}, \boldsymbol{\theta}^{*(n)}) + o((2\lambda)^{-1}).\end{aligned} \tag{12}$$

It is now clear that λ plays the role of an inverse step size—our experiments later will confirm that there is little harm in choosing λ too large. With this in mind, we can now contrast BCD-prox with iPDA. In BCD-prox, we use the data $\mathbf{Y}$ to initialize the algorithm; subsequently, the algorithm may take many proximal steps of the form (12) to reach a desired optimum. If the data $\mathbf{Y}$ is heavily contaminated with noise, it may be wise to move far away from $\mathbf{Y}$ as we iterate.

In contrast, iPDA's regularization term is $\lambda\|\mathbf{X} - \mathbf{Y}\|^2$. Roughly speaking, iPDA searches for $\mathbf{X}$ in a neighborhood of $\mathbf{Y}$; the diameter of this neighborhood is inversely related to λ. When the magnitude of the noise $\mathbf{Z}$ is small, searching for $\mathbf{X}$ in a small neighborhood of $\mathbf{Y}$ is reasonable. For real data problems in which the magnitude of $\mathbf{Z}$ is unknown, however, choosing λ *a priori* becomes difficult.

An additional important difference between BCD-prox and iPDA has to do with convexity, which we discuss next.

4. Convergence

In practice, we implement the argmin steps in Alg.1 using the LBFGS algorithm, implemented in Python via scipy.optimize.minimize. Throughout this work, when using LBFGS, we use automatic differentiation to supply the optimizer with gradients of the objective function. We stop Alg.1 when the error E changes less than 10^{-8} from one iteration to the next.

To see when this happens, we take another look at the optimization over the states $\mathbf{X}$ in (11) at iteration n. This objective function F_n has two parts. The optimal solution of the first part (E) is the predicted states $\hat{\mathbf{X}}^{(n)}$. The optimal solution of the proximal part is $\mathbf{X}^{*(n-1)}$. When we optimize this objective function to find $\mathbf{X}^{*(n)}$, there are three cases: 1) $\mathbf{X}^{*(n)} = \hat{\mathbf{X}}^{(n)}$, 2) $\mathbf{X}^{*(n)} = \mathbf{X}^{*(n-1)}$, and 3) $\mathbf{X}^{*(n)}$ is neither $\hat{\mathbf{X}}^{(n)}$ nor $\mathbf{X}^{*(n-1)}$. Our algorithm stops when we are in case 1 or 2 since further optimization over $\boldsymbol{\theta}$ and $\mathbf{X}$ changes nothing. In case 3, the algorithm continues, leading to further optimization steps to decrease error.

Indeed, let us note that steps 5 and 6 in Alg. 1 together imply

$$E(\mathbf{X}^{*(n)}, \boldsymbol{\theta}^{*(n)}) \leq E(\mathbf{X}^{*(n-1)}, \boldsymbol{\theta}^{*(n-1)}). \tag{13}$$

The function E, bounded below by 0, is non-increasing along the trajectory $\{(\mathbf{X}^{*(n)}, \boldsymbol{\theta}^{*(n)})\}_{n\geq 1}$. Hence $\{E(\mathbf{X}^{*(n)}, \boldsymbol{\theta}^{*(n)})\}_{n\geq 1}$ must converge to some $E^* \geq 0$.

Next we offer convergence theory for the Euler version of BCD-prox. We believe this theory can also be established for the general m-step version of BCD-prox; however, the calculations will be lengthier. In this subsection, we let $\mathrm{x}_i = \mathrm{x}_{(t_i)} \in \mathbb{R}^d$. For T even, set

$$\mathrm{x}^+ = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{T/2}\}, \quad \mathrm{x}^- = \{\mathrm{x}_{T/2+1}, \ldots, \mathrm{x}_T\}.$$

For T odd, replace $T/2$ by $(T-1)/2$ in the above definitions. In words, x^+ is the first half of the state series while x^- is the second half of the state series. Note that $\mathbf{X} = (\mathrm{x}^+, \mathrm{x}^-)$.

Assume that f is at most linear in $\boldsymbol{\theta}$, so that $\mathrm{f}(\mathrm{x}, \boldsymbol{\theta}) = \mathrm{f}_0(\mathrm{x}) + \mathrm{f}_1(\mathrm{x})\boldsymbol{\theta}$, with $\mathrm{f}_1 : \mathbb{R}^d \to \mathbb{R}^{d\times p}$ assumed to have full column rank for all x.

Now initialize $\mathbf{X}^0 = \mathbf{Y}$ and proceed sequentially with the following steps for $n \geq 1$:

$$\boldsymbol{\theta}^n = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} F_n(\mathbf{X}^{n-1}, \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} E(\mathbf{X}^{n-1}, \boldsymbol{\theta}) \tag{14a}$$

$$(\mathrm{x}^-)^n = \underset{\mathrm{x}^-}{\operatorname{argmin}} F_n((\mathrm{x}^+)^{n-1}, \mathrm{x}^-, \boldsymbol{\theta}^n) \tag{14b}$$

$$(\mathrm{x}^+)^n = \underset{\mathrm{x}^+}{\operatorname{argmin}} F_n(\mathrm{x}^+, (\mathrm{x}^-)^n, \boldsymbol{\theta}^n) \tag{14c}$$

$$\mathbf{X}^n = ((\mathrm{x}^+)^n, (\mathrm{x}^-)^n) \tag{14d}$$

We now seek to apply the results of Xu & Yin (2013). In order to do so, we will establish strong convexity of each of the steps in (14). We begin by noting that

$$E(\mathbf{X}, \boldsymbol{\theta}) = \sum_{i=1}^{T-1} \|\mathrm{x}_{i+1} - \mathrm{x}_i - \mathrm{f}_0(\mathrm{x}_i)\Delta_i + \mathrm{f}_1(\mathrm{x}_i)\boldsymbol{\theta}\Delta_i\|^2.$$

We compute the $p \times p$ Hessian

$$\nabla_{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}} E = 2\sum_{i=1}^{T-1} (\mathrm{f}_1(\mathrm{x}_i))^T \mathrm{f}_1(\mathrm{x}_i)\Delta_i^2.$$

Since f_1 has full column rank, it follows that $E(\mathbf{X}, \boldsymbol{\theta})$ is strongly convex in $\boldsymbol{\theta}$ with $\mathbf{X}$ held fixed.

Next, suppose all Δ_i are zero. Then (4) reduces to $E(\mathbf{X}, \boldsymbol{\theta}; \Delta_i = 0) = \sum_{i=1}^{T-1} \|\mathrm{x}_{i+1} - \mathrm{x}_i\|^2$. This is a

quadratic form written as a sum of squares; hence it is positive semidefinite. We sharpen this to positive definiteness by examining derivatives. First we hold $\mathbf{x}^-$ and $\boldsymbol{\theta}$ fixed and consider $A = \nabla_{\mathbf{x}^+}\nabla_{\mathbf{x}^+}E(\mathbf{X}, \boldsymbol{\theta}; \Delta_i = 0)$, the Hessian with respect to $\mathbf{x}^+$ only. We obtain

$$A = \begin{bmatrix} 2I & -2I & & & \\ -2I & 4I & -2I & & \\ & -2I & 4I & \ddots & \\ & & \ddots & \ddots & -2I \\ & & & -2I & 4I \end{bmatrix}$$

Here each I is a $d \times d$ identity block. The positive semidefiniteness established above implies that all eigenvalues of A are nonnegative. By an induction argument, we can show that $\det A = 2^{dT/2}$, implying that the eigenvalues of A are bounded away from zero. Hence the quadratic form $E(\mathbf{X}, \boldsymbol{\theta}; \Delta_i = 0)$ restricted to $\mathbf{x}^+$ (with $\mathbf{x}^-$ held fixed) is strongly convex. In an analogous way, we can show that $E(\mathbf{X}, \boldsymbol{\theta}; \Delta_i = 0)$ restricted to $\mathbf{x}^-$ (with $\mathbf{x}^+$ held fixed) is strongly convex. Both of these properties hold at $\Delta_i = 0$. Because the eigenvalues of both restrictions are continuous functions of Δ_i, there exists $\delta > 0$ such that for $\Delta_i \in (0, \delta)$, the eigenvalues remain bounded away from zero.

Then we have the following first convergence result.

Theorem 2. *Suppose all $\Delta_i \in (0, \delta)$ for the δ established above. Suppose* f *is linear in $\boldsymbol{\theta}$ with the full-rank condition described above. Then there exists an interval of λ values for which the algorithm* (14) *converges to a Nash equilibrium $(\overline{\mathbf{X}}, \overline{\boldsymbol{\theta}})$ of the objective E defined in* (4).

Proof. The result follows directly from Theorem 2.3 from Xu & Yin (2013); we have verified all hypotheses. In particular, when all $\Delta_i \in (0, \delta)$, $E(\mathbf{X}, \boldsymbol{\theta})$ is strongly convex in $\mathbf{x}^+$ (with $\mathbf{x}^-$ and $\boldsymbol{\theta}$ held fixed) and strongly convex in $\mathbf{x}^-$ (with $\mathbf{x}^+$ and $\boldsymbol{\theta}$ held fixed). □

Let us further assume that f satisfies the Kurdyka-Lojasiewicz (KL) property described in Section 2.2 of Xu & Yin (2013). In particular, if each component of f is real analytic, the KL property will be satisfied. Together with linearity of f in $\boldsymbol{\theta}$, this includes numerous vector fields of interest, including all ODE in our experimental results. (For FitzHugh–Nagumo, a change of variables renders the system linear in the parameters.) Then we have a second convergence result.

Theorem 3. *Suppose in addition to the hypotheses of Theorem 2,* f *is smooth and satisfies the KL property. Then assuming the algorithm defined by* (14) *begins sufficiently close to a global minimizer, it will converge to a global minimizer of E defined in* (4).

Proof. The result follows directly from Corollary 2.7 and Theorem 2.8 of Xu & Yin (2013); we have verified all hypotheses. □

5. Experiments

We briefly explain the datasets (models) that we used in our experiments here. In the supplementary material, we detail the ODEs and true parameter values for 1) Lotka–Volterra with two-dimensional states and four unknown parameters. 2) FitzHugh–Nagumo with two-dimensional states and four unknown parameters. 3) Rössler attractor with three-dimensional states and three unknown parameters. 4) Lorenz-96 with 40 nonlinear equations and one unknown parameter, the largest ODE we found in the literature.

We create the clean states using a Runge-Kutta method of order 5. In all of our experiments, unless otherwise stated, we use the three-step Adams-Bashforth method to discretize the ODE. Also, unless otherwise stated, we added Gaussian noise with mean 0 and variance σ^2 to each of the clean states to create the noisy observations.

Advantages of our approach. Before detailing our experimental results, let us give an overview of our findings. BCD-prox is robust with respect to its only hyperparameter λ. We will show below that for a broad range of λ values, BCD-prox works well. We fix it to $\lambda = 1$ in our later experiments. As explained before, previous methods have a large number of hyperparameters, which are difficult to set.

BCD-prox can be trained quickly. On a standard laptop, it takes around 20 seconds for BCD-prox to learn the parameters and states jointly on ODE problems with 400 states. The spline-based methods take a few minutes and Bayesian methods take a few hours to converge on the same problem.

Because BCD-prox, unlike Bayesian methods, does not make assumptions about the type of the noise or distribution of the states, it performs well under different noise and state distributions. In particular, as the magnitude of noise in the observations increases, BCD-prox clearly outperforms the extended Kalman filter.

As our experiments confirm, both spline-based and Bayesian approaches are very sensitive to the initialization of the ODE parameters. If we initialize them far away from the true values, they do not converge. BCD-prox is much more robust. This robustness stems from simultaneously learning states and parameters. Even if the estimated and true parameters differ at some iteration, they can converge later, as the estimated states converge to the clean states.

Evaluation metrics. Let $\boldsymbol{\theta}$ and $\mathbf{X}$ denote the true parameters and the clean states, respectively. Let $\boldsymbol{\theta}^*$ and $\hat{\mathbf{X}}$ denote the estimated parameters and the predicted states. We report the Frobenius norm of $\mathbf{X} - \hat{\mathbf{X}}$ as the prediction error.

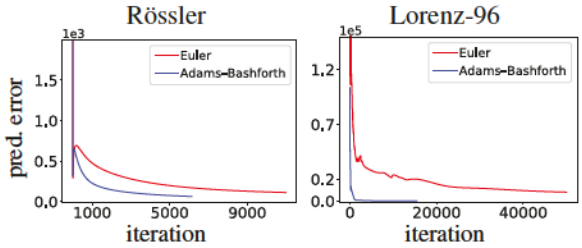

Figure 1. Prediction error at different iterations of our algorithm with different discretization methods. The noise variance of observations is $\sigma^2 = 1$. Our learning strategy decreases the error significantly.

We also consider $|\theta_l - \theta_l^*|$ as the lth parameter error. To compute predicted states, we first take $\boldsymbol{\theta}^*$ as the parameter and $\mathbf{x}^*_{(t_1)}$ as the initial state; we then repeatedly apply either Euler (3) or multistep (6) numerical integration.

Optimization of objective (4) leads to better estimation. At each iteration n of our optimization, we compute the predicted states $\hat{\mathbf{X}}^{(n)}$ and report the prediction error. In Fig. 1, we consider two kinds of discretization: 1) one-step Euler method, and 2) three-step Adams-Bashforth method. Note that as we increase the order, we expect to see more accurate results.

The variance of the noisy observations is $\sigma^2 = 1$. The supplementary material contains the results for the FitzHugh–Nagumo model and also for the case of $\sigma^2 = 0.5$. Fig. 1 shows that at the first iteration the error is significant in all models. The error is $\sim 10^3$ for FitzHugh–Nagumo and Rössler, and $\sim 1.5 \times 10^5$ for Lorenz-96 model.

After several iterations of our algorithm, the error decreases significantly, no matter what kind of discretization we use. Three-step Adams-Bashforth performs better than Euler in general: it converges faster and achieves a smaller final error. This is especially clear for the Lorenz-96 model: the final error is near zero for three-step Adams-Bashforth, but near 10^4 for Euler.

The last point about Fig. 1 is that, as expected, the prediction error increases at times; the error does not decrease monotonically. This mainly happens at the first few iterations. The main reason for this behavior is that our objective function in (4) is different from the prediction error. We cannot directly optimize the prediction error because we do not have access to the clean states. Still, the fact that our algorithm eventually brings the prediction error close to zero suggests that minimizing the objective in (4) has the same effect as minimizing the prediction error.

Robustness to the hyperparameter λ. The only hyperparameter in our algorithm is λ. In Fig. 2, we set λ in turn to a set of values from 0 to 20, run our algorithm, and report the results after convergence. In both models, we generate observations with the variance $\sigma^2 = 0.5$. Because of randomness included in creating noisy observations, we create 10 sets of observations, run our algorithm once for each of them, and report the mean in Fig. 2. We also show the standard deviation in prediction errors, but not in parameter values (to avoid clutter).

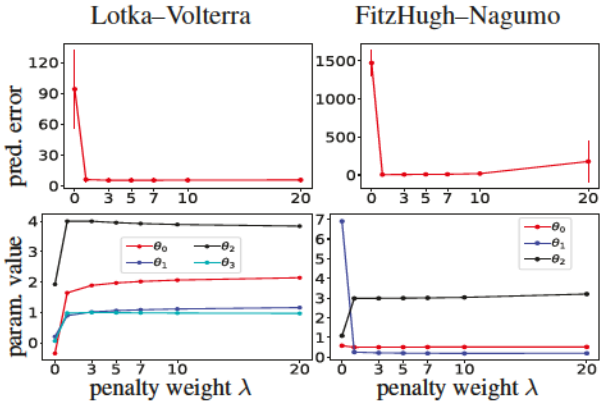

Figure 2. Robustness to the hyperparameter λ. The true parameters are $\theta_0 = .5, \theta_1 = .3$, and $\theta_2 = 3$ in the FitzHugh–Nagumo and $\theta_0 = 2, \theta_1 = 1, \theta_2 = 4$, and $\theta_3 = 1$ in the Lotka–Volterra. For each λ, we report the mean error and parameter value in 10 experiments.

In Fig. 2 we report the prediction error and the estimated parameters for each value of λ. The true values for the FitzHugh–Nagumo are $\theta_0 = .5, \theta_1 = .3$, and $\theta_2 = 3$. For the Lotka–Volterra model, the true values are $\theta_0 = 2, \theta_1 = 1, \theta_2 = 4$, and $\theta_3 = 1$.

We see in Fig. 2 that for $\lambda > 0$, BCD-prox correctly finds the parameters and brings the error close to zero. Also, in the range of $\lambda = 1$ to 20, the errors and the estimated parameters remain almost the same. We have found that increasing λ to 1 000 does not change the estimated parameters. The only disadvantage of increasing λ to a large value is that training time increases—as explained above, increasing λ is analogous to decreasing the step size in a gradient descent method. Large λ implies that states can change very little from one iteration to another, forcing the algorithm to run longer for convergence. The algorithm, as explained in detail before, does not work well when $\lambda = 0$; in this case, the algorithm stops after a single iteration, with the predicted states far from the clean states.

Comparison with other methods (robustness to initialization). As the first experiment, we compare BCD-prox with three other methods, each of them from a different category. Among the iPDA (spline-based) methods, we use a MATLAB code available online (Ramsay et al., 2007), denoted by "iPDA" in our experiments. Among the Bayesian approaches, we use an R code available online (Dondelinger et al., 2013), denoted "Bayes" in our experiments. We also implement a method that uses the iterative least square approach, denoted "lsq" in our experiments. This method considers the parameters and the initial state as the unknown

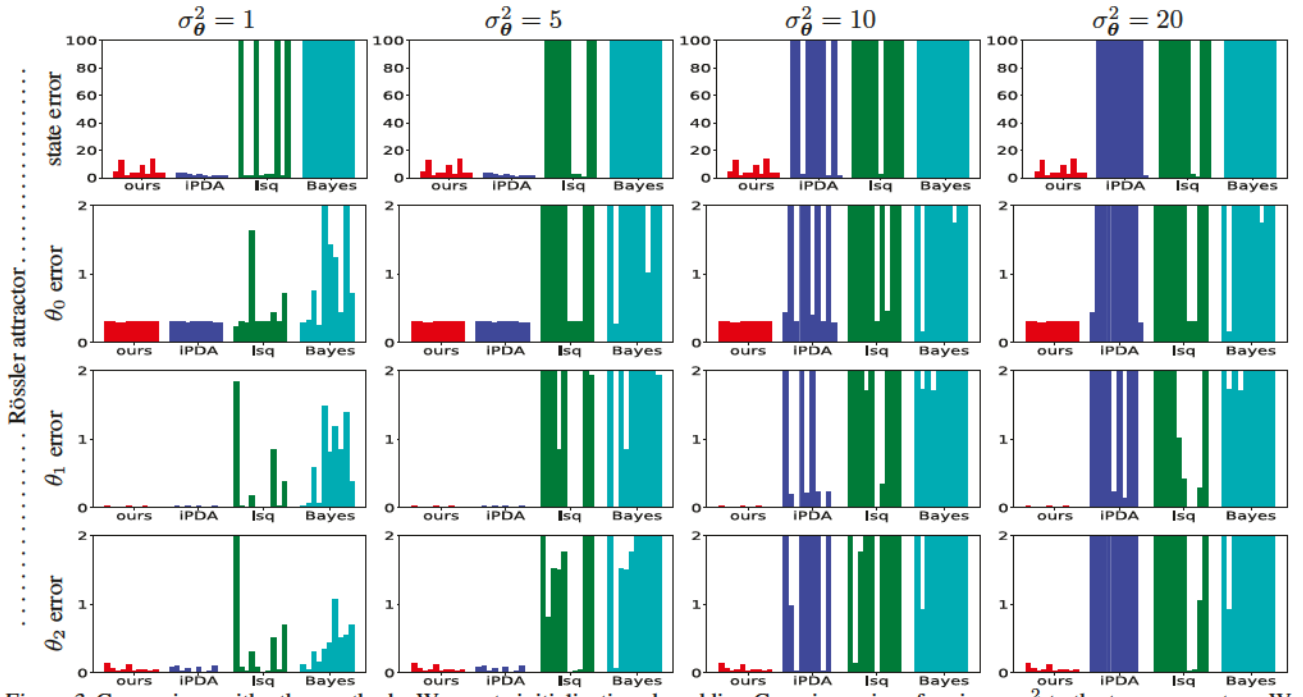

Figure 3. Comparison with other methods. We create initializations by adding Gaussian noise of variance σ^2_{θ} to the true parameters. We create 10 sets of observations and initializations per each σ^2_{θ} and report the errors. Each error bar corresponds to the error in one of the experiments. BCD-prox performs significantly better.

variables. To implement lsq, we use the Python LMFIT package (Newville et al., 2014). The variance of the noisy observations is $\sigma^2 = 0.5$.

All methods including ours need an initial guess for the unknown parameters. We add Gaussian noise with mean 0 and variance σ^2_{θ} to the true parameter and use the result to initialize the methods. Fig. 3 shows the results for the Rössler model and the supplementary material contains the results on the FitzHugh–Nagumo model. We change the variance from $\sigma^2_{\theta} = 1$ to 20. Since there is randomness in both initialization and observation, we repeat the experiment 10 times. Note that the comparisons are fair, with the same observations and initializations used across all methods.

In Fig. 3, each of the bars corresponds to the prediction or parameter error for one of the methods in one of the experiments. Hence there are 10 error bars for each of the methods in each plot. We set $\lambda = 1$ in BCD-prox for all the experiments. For the other methods, we chose the best hyperparameters that we could determine after careful experimentation.

The first point in Fig. 3 is that BCD-prox is robust with respect to the initialization, while the other methods are not. The total number of experiments per method is 80 (40 for the FitzHugh–Nagumo and 40 for Rössler). The prediction error of BCD-prox exceeds 100 in 4 experiments. The prediction error of iPDA (the second best method after ours) exceeds 100 in 39 experiments (nearly half the experiments). For lsq and Bayes, the errors are substantially worse.

Fig. 3 shows that almost all the methods work well when the initialization is close to the true parameters (small noise). In reality, we do not know what the real parameters are; it is reasonable to say that the last column of Fig. 3 (initialization with the largest noise) determines which method performs better in real-world applications. BCD-prox outperforms other methods in both prediction and parameter error.

In our second experiment, we compare BCD-prox with the mean-field variational Bayes method of Gorbach et al. (2017) on the Lotka–Volterra model. The mean-field method is only applicable to differential equations with a specific form—see Eq. (10) in Gorbach et al. (2017). While we cannot apply the mean-field method to the FitzHugh–Nagumo and Rössler models, we can apply it to the Lotka–Volterra model. In Fig. 4, we compare the methods by prediction and parameter errors.

Fig. 4 (first row) shows the results for $\sigma^2 = 1$ (results for other variances are in supplementary material). Similar to our previous experiments, we generate 10 sets of noisy observations and each bar corresponds to the error for one of the methods in one of the experiments.

Fig. 4 shows that the average error of BCD-prox is less than that of the mean-field method in almost all cases. The average parameter error of the mean-field method for θ_0 and θ_1 becomes around 3 and 8, respectively, but the average error of BCD-prox for both parameters remains less than 1. The results in the supplementary material show that as we increase the noise in the observations, the error of

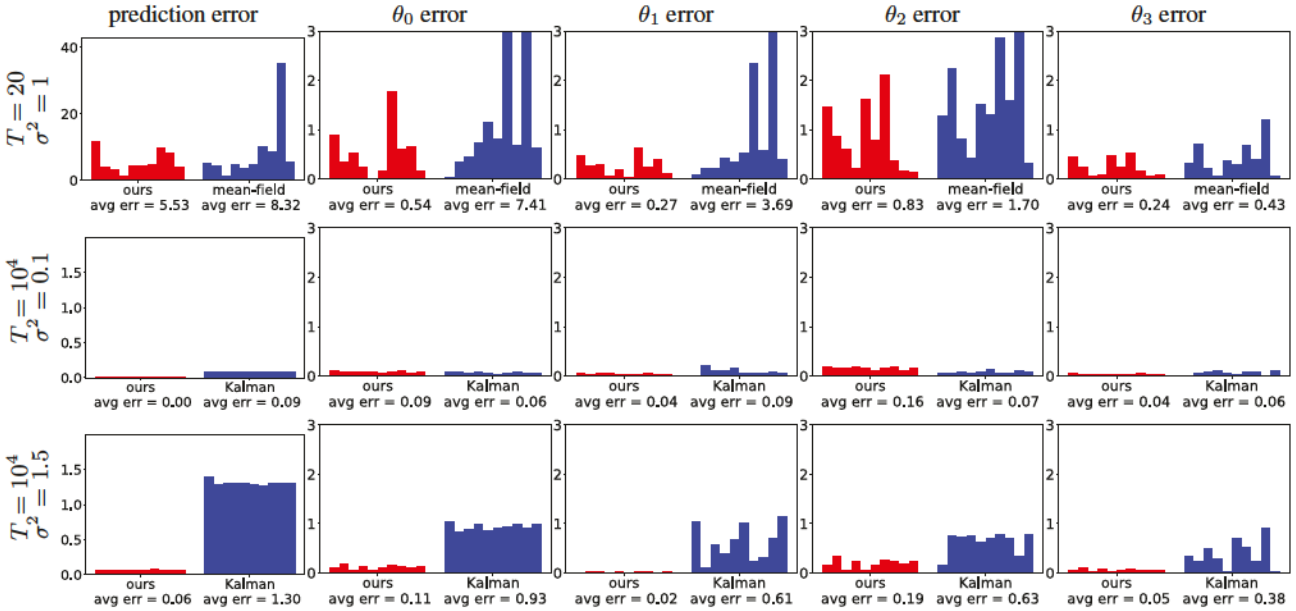

Figure 4. We generate 10 sets of observations for the Lotka–Volterra model and report the error for each of the experiments. The average error has been reported below each plot. *First row:* comparison with the mean-field method of Gorbach et al. (2017), where the noisy observations have the variance $\sigma^2 = 1$. *Second and third rows:* comparison with the extended Kalman filter (EKF) where the noisy observations have the variance $\sigma^2 = 0.1$ and 1.5, respectively.The number of observation is $T = 10\,000$.

both methods increases. Still, BCD-prox is more robust to observational noise than the mean-field method.

Comparison with extended Kalman filter (EKF). We follow Sitz et al. (2002) to apply EKF to our problem. We use an open-source Python code (Labbe, 2014) in our implementation. For details, see the supplementary material.

In the second and third rows of Fig. 4, we compare BCD-prox with EKF on the Lotka–Volterra model. We compare the methods in different settings by changing the amount of noise and the number of samples. The noise variances are $\sigma^2 = 0.1$ and $\sigma^2 = 1.5$ and the number of samples are $T = 20$ (time range $[0, 2]$) and $T = 10\,000$ (time range $[0, 1\,000]$). The results for $T = 20$ can be found in the supplementary material.

In Fig. 4 we report the average estimation error instead of the prediction error. Estimation error is defined as the difference between the clean states and the estimated states $\mathbf{X}^*$. We report the estimation error because the prediction error of EKF goes to infinity. To see why this happens, note that to obtain reasonable predictions we need good estimations of the parameters and the initial state. Since EKF is an online method, it never updates the initial state. Given that the initial state is noisy, no matter how well parameters are estimated, the prediction error becomes very large. BCD-prox updates the initial state, yielding small prediction error.

We found that the only setting in which EKF performs comparably to BCD-prox is the case of $T = 10\,000$ and $\sigma^2 = 0.1$. In other words, EKF works fine when we have long time series with low noise. In more realistic settings, BCD-prox significantly outperforms EKF. A key difference between the two methods is that EKF is an online method while ours is a batch method, iterating over the entire data set repeatedly. Consequently, BCD-prox updates parameters based on information in all the states, leading to more robust updates than is possible with EKF, which updates parameters based on a single observation.

We also see that the error of both methods becomes smaller as we increase the number of samples T. This is expected because increasing T is equivalent to giving more information about the model to the methods. The average estimation error of BCD-prox becomes almost 0 for large T.

Conclusion. BCD-prox addresses issues of previous approaches to simultaneous parameter estimation and filtering, achieving fast training and robustness to noise, initialization, and hyperparameter tuning. We have shown how to use BCD-prox with multistep ODE integration methods. Additional features of BCD-prox include its connection to BCD and proximal methods, its unified objective function, and a convergence theory resulting from blockwise convexity. In ongoing/future work, we seek to extend BCD-prox to estimate the vector field $\mathbf{f}$ from noisy observations.

Acknowledgements

H. S. Bhat was partially supported by NSF award DMS-1723272. Both authors acknowledge use of the MERCED computational cluster, funded by NSF award ACI-1429783.

References

Bard, Y. *Nonlinear Parameter Estimation*. Academic Press, 1973.

Benson, M. Parameter fitting in dynamic models. *Ecological Modelling*, 6(2):97–115, 1979.

Calderhead, B., Girolami, M., and Lawrence, N. D. Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 217–224, 2009.

Cao, J. and Zhao, H. Estimating dynamic models for gene regulation networks. *Bioinformatics*, 24(14):1619–1624, 2008.

Cao, J., Wang, L., and Xu, J. Robust estimation for ordinary differential equation models. *Biometrics*, 67(4):1305–1313, 2011.

Chatterji, N. S. and Bartlett, P. L. Alternating minimization for dictionary learning with random initialization. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1994–2003, 2017.

Dattner, I. and Klaassen, C. A. J. Optimal rate of direct estimators in systems of ordinary differential equations linear in functions of the parameters. *Electronic Journal of Statistics*, 9(2):1939–1973, 2015.

Dondelinger, F., Husmeier, D., Rogers, S., and Filippone, M. ODE parameter inference using adaptive gradient matching with Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 216–228, 2013.

Girolami, M. Bayesian inference for differential equations. *Theoretical Computer Science*, 408(1):4–16, 2008.

Gorbach, N. S., Bauer, S., and Buhmann, J. M. Scalable variational inference for dynamical systems. In *Advances in Neural Information Processing Systems*, pp. 4809–4818, 2017.

Gugushvili, S. and Klaassen, C. A. $\sqrt{n}$-consistent parameter estimation for systems of ordinary differential equations: bypassing numerical integration via smoothing. *Bernoulli*, 18(3):1061–1098, 2012.

Himmelblau, D. M., Jones, C. R., and Bischoff, K. B. Determination of rate constants for complex kinetics models. *Industrial & Engineering Chemistry Fundamentals*, 6(4):539–543, 1967.

Hosten, L. A comparative study of short cut procedures for parameter estimation in differential equations. *Computers & Chemical Engineering*, 3(1-4):117–126, 1979.

Iserles, A. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, second edition, 2009.

Labbe, R. Kalman and Bayesian filters in Python, 2014. URL `https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python`.

Li, Q., Zhou, Y., Liang, Y., and Varshney, P. K. Convergence analysis of proximal gradient with momentum for nonconvex optimization. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 2111–2119, 2017.

Li, Y., Liang, Y., and Risteski, A. Recovery guarantee of weighted low-rank approximation via alternating minimization. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2358–2367, 2016. URL `http://jmlr.org/proceedings/papers/v48/lii16.html`.

Liang, H. and Wu, H. Parameter estimation for differential equation models using a framework of measurement error in regression models. *Journal of the American Statistical Association*, 103(484):1570–1583, 2008.

Newville, M., Stensitzki, T., Allen, D. B., and Ingargiola, A. LMFIT: Non-linear least-square minimization and curve-fitting for Python, 2014.

Palais, R. S. and Palais, R. A. *Differential Equations, Mechanics, and Computation*. Number 51 in Student Mathematical Library. American Math. Soc., Providence, RI, 2009.

Parikh, N. and Boyd, S. P. Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

Poyton, A., Varziri, M., McAuley, K., McLellan, P., and Ramsay, J. Parameter estimation in continuous-time dynamic models using principal differential analysis. *Computers & Chemical Engineering*, 30(4):698–708, 2006.

Ramsay, J. O., Hooker, G., Campbell, D., and Cao, J. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

Sitz, A., Schwarz, U., Kurths, J., and Voss, H. U. Estimation of parameters and unobserved components for nonlinear systems from noisy time series. *Phys. Rev. E*, 66(1):016210, 2002. doi: 10.1103/PhysRevE.66.016210.

Sun, J., Lu, J., Xu, T., and Bi, J. Multi-view sparse co-clustering via proximal alternating linearized minimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 757–766, 2015. URL `http://jmlr.org/proceedings/papers/v37/sunb15.html`.

Varah, J. M. A spline least squares method for numerical parameter estimation in differential equations. *SIAM Journal on Scientific and Statistical Computing*, 3(1):28–46, 1982.

Xu, Y. and Yin, W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM J. Imaging Sciences*, 6(3):1758–1789, 2013. doi: 10.1137/120887795. URL `https://doi.org/10.1137/120887795`.

Yi, X., Caramanis, C., and Sanghavi, S. Alternating minimization for mixed linear regression. In *Proceedings of the 31st International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 613–621, 2014. URL `http://jmlr.org/proceedings/papers/v32/yia14.html`.

Zhang, Z. and Brand, M. Convergent block coordinate descent for training Tikhonov regularized deep neural networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1719–1728, 2017.