SIAM J. SCI. COMPUT.
Vol. 41, No. 5, pp. S129–S148

AN UNCERTAINTY-WEIGHTED ASYNCHRONOUS ADMM METHOD FOR PARALLEL PDE PARAMETER ESTIMATION*

SAMY WU FUNG† AND LARS RUTHOTTO†

Abstract. We consider a global variable consensus alternating direction method of multipliers (ADMM) algorithm for estimating parameters of partial differential equations (PDEs) asynchronously and in parallel. Motivated by problems with many measurements, we partition the data and distribute the resulting subproblems among the available workers. Since each subproblem can be associated with different forward models and right-hand sides, this provides ample options for tailoring the method to different applications, including multisource and multiphysics PDE parameter estimation problems. We also consider an asynchronous variant of consensus ADMM to reduce communication and latency. Our key contribution is a novel weighting scheme that empirically increases the progress made in early iterations of the consensus ADMM scheme and is attractive when using a large number of subproblems. This makes consensus ADMM competitive for solving PDE parameter estimation, which incurs immense cost per iteration. The weights in our scheme are related to the uncertainty associated with the solutions of each subproblem. We exemplarily show that the weighting scheme combined with the asynchronous implementation reduces the time-to-solution and lowers the communication costs for a 3D single-physics and multiphysics PDE parameter estimation problems.

Key words. PDE-constrained optimization, parameter estimation, alternating direction method of multipliers, inverse problems, distributed optimization, multiphysics inversion

AMS subject classifications. 35R30, 90C06, 65N21, 65Y05

DOI. 10.1137/18M119166X

1. Introduction.

Recent technological advances have allowed us to collect data at massive scales with relative ease. This trend, often referred to as *big data*, has given rise to notoriously challenging high-dimensional parameter estimation problems. A common example is the computation of the maximum a posteriori (MAP) estimate [8, 44] of large-scale Bayesian inverse problems. In this case, the parameter estimation problem is solved iteratively using gradient-based optimization. This requires numerous simulations that may involve different physical models and commonly scale to millions of variables [5], leading to very high costs in both CPU time and memory. As a result, general approaches, such as model order reduction schemes [35, 3, 13], parallel and distributed schemes [6], and importance sampling and stochastic optimization schemes [49, 36], have become highly desirable, if not necessary, for solving these types of problems.

In this paper, we focus on parallel and distributed techniques. We consider the consensus alternating direction method of multipliers (ADMM) [6, 15, 27] as well as its asynchronous variant (async-ADMM), which aims at reducing latencies and thereby reduce the time-to-solution [52]. Consensus ADMM has previously been applied to high-dimensional inverse problems in data sciences [32, 34], statistical learning [6, 15, 47, 51], and imaging [14, 24, 28]. The algorithm tackles large-scale problems by partitioning the data into, say, N smaller batches that can be solved in parallel, and in

*Received by the editors June 1, 2018; accepted for publication (in revised form) March 29, 2019; published electronically October 29, 2019.
https://doi.org/10.1137/18M119166X
Funding: This work was supported by the U.S. National Science Foundation (NSF) through awards DMS 1522599 and DMS 1751636.
†Departments of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (samy.wu@emory.edu, lruthotto@emory.edu).

some cases, explicitly. This often leads to an improved ratio of local computation and communication. More specifically, each iteration of the algorithm breaks down into (1) N subproblems using parts of the data that are solved locally and independently, (2) an averaging step that is performed once all N subproblems have been solved, and (3) an explicit update of the dual variable. The main change in the async-ADMM variant is that the averaging step is performed once $N_a < N$ subproblems have been solved, reducing the overall latency.

As we demonstrate in our numerical experiments, a straightforward implementation of consensus ADMM converges slowly, in particular when the information contained in the split data sets is complementary and the number of batches, N, is large. One problem in these cases is that the averaging step in consensus ADMM gives equal weight to all the solutions corresponding to each batch, leading to an uninformed averaged reconstruction. In large-scale problems, such as partial differential equation (PDE) parameter estimation, this renders consensus ADMM prohibitive since often only a few iterations are affordable.

To increase the performance of consensus ADMM, particularly in early iterations, we introduce a novel weighting scheme that improves the convergence of consensus ADMM. The weights are obtained in a systematic and efficient way using the framework of uncertainty quantification (UQ) proposed in [12]. We demonstrate the effect of the weights on a collection of linear inverse problems. We also outline the potential of our method by comparing it to the Gauss–Newton (GN) method [17] and the nonlinear conjugate gradient (NLCG) method [20] on a single-physics PDE parameter estimation problem involving a travel time tomography survey, and a multiphysics parameter estimation problem involving direct current resistivity (DCR) and travel time tomography [48] surveys.

The remainder of the paper is organized as follows. In section 2, we review the mathematical framework for MAP estimation and UQ as well as numerical optimization algorithms for their computations. In section 3, we present the weighted consensus ADMM method and its asynchronous variant. In section 4, we outline the potential of our method for a series of numerical experiments, and finally, we summarize the paper in section 5.

2. Mathematical background and numerical implementation. In this section, we briefly review the computation of the MAP estimate and principles of UQ in the context of large-scale Bayesian inverse problems (see, e.g., [8, 44] for a detailed overview). We limit the discussion to the finite-dimensional case since we follow the discretize-then-optimize approach; however, an overview of the infinite-dimensional case can be found in [44]. We also review optimization techniques for computing the MAP estimate and their associated challenges.

2.1. MAP Estimation and UQ. We consider additive noise-corrupted observations

$$Y = \mathcal{F}(X) + E, \tag{2.1}$$

where $\mathcal{F}: \mathbb{R}^n \mapsto \mathbb{R}^m$ is the parameter-to-observable map, and Y, X, and E are random vectors corresponding to the observations, the model parameter, and the measurement noise, respectively. In the following, we denote their corresponding realizations by $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^n$, and $\boldsymbol{\epsilon} \in \mathbb{R}^m$.

We employ the prior probability density function (PDF), $\pi_{\text{prior}}: \mathbb{R}^n \mapsto \mathbb{R}$, which describes prior information we may have about X, and the likelihood function $\pi(\mathbf{y}|\mathbf{x})$,

which describes the relationship between the measurements $\mathbf{y}$ and the unknown model parameters $\mathbf{x}$. We use Bayes' theorem to obtain the posterior PDF,

$$\pi_{\text{post}}(\mathbf{x}) \propto \pi_{\text{prior}}(\mathbf{x})\pi(\mathbf{y}|\mathbf{x}), \tag{2.2}$$

and compute the MAP point by maximizing the posterior PDF, that is,

$$\mathbf{x}_{\text{MAP}} = \operatorname*{argmax}_{\mathbf{x}} \pi_{\text{post}}(\mathbf{x}). \tag{2.3}$$

For simplicity, we assume that X and E are statistically independent, and we limit the discussion to the case where the prior PDF is Gaussian and E is a random vector whose entries are independently and identically distributed so that $E \sim \mathcal{N}(\mathbf{0}, \Gamma_{\text{noise}})$, where $\Gamma_{\text{noise}} \in \mathbb{R}^{m\times m}$ is the diagonal noise-covariance matrix. In this case, the likelihood and prior PDFs are given by

$$\pi(\mathbf{y}|\mathbf{x}) \propto \exp(-\Phi(\mathbf{x})) \quad \text{and} \quad \pi_{\text{prior}}(\mathbf{x}) \propto \exp(-\mathcal{R}(\mathbf{x})), \tag{2.4}$$

respectively, where, due to the assumptions above,

$$\Phi(\mathbf{x}) = \frac{1}{2}\|\mathcal{F}(\mathbf{x}) - \mathbf{y}\|^2_{\mathbf{\Gamma}^{-1}_{\text{noise}}} \quad \text{and} \quad \mathcal{R}(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_{\text{ref}}\|^2_{\mathbf{\Gamma}^{-1}_{\text{prior}}}. \tag{2.5}$$

Here, $\mathbf{x}_{\text{ref}}$ is the mean of the model parameter prior PDF, and $\Gamma_{\text{prior}} \in \mathbb{R}^{n\times n}$ is the covariance matrix of the prior PDF. Using (2.2) and (2.4), we can restate the posterior PDF in closed form as

$$\pi_{\text{post}}(\mathbf{x}) \propto \exp\Big(- \Phi(\mathbf{x}) - \mathcal{R}(\mathbf{x})\Big), \tag{2.6}$$

and the MAP estimate can then be found by solving

$$\mathbf{x}_{\text{MAP}} = \operatorname*{argmin}_{\mathbf{x}} \ \Big(\Phi(\mathbf{x}) + \mathcal{R}(\mathbf{x})\Big). \tag{2.7}$$

When $\mathcal{F}$ is a linear operator, that is, $\mathcal{F} = \mathbf{A} \in \mathbb{R}^{m\times n}$, the posterior PDF is also Gaussian, and we can write its covariance matrix $\Gamma_{\text{post}} \in \mathbb{R}^{n\times n}$ in closed form as

$$\mathbf{\Gamma}_{\text{post}} = (\mathbf{A}^\top \mathbf{\Gamma}^{-1}_{\text{noise}}\mathbf{A} + \mathbf{\Gamma}^{-1}_{\text{prior}})^{-1}, \tag{2.8}$$

which can be used for quantifying uncertainties of the model parameter $\mathbf{x}$. In the context of large-scale PDE parameter estimation, however, the matrix $\mathbf{A}$, let alone its inverse, is seldom constructed. The computation of $\mathbf{\Gamma}_{\text{post}}$ is intractable, and we therefore follow [12] and use an iterative method to obtain an approximation in section 3.2.

2.2. Numerical optimization. In large-scale PDE parameter estimation, we are concerned with the case where massive amounts of data are available, leading to numerous right-hand sides and potentially multiple PDEs [19, 48]. Here, we split the misfit function in (2.5) into N terms, i.e.,

$$\Phi(\mathbf{x}) = \sum_{j=1}^{N} \Phi_j(\mathbf{x}), \quad \text{where} \quad \Phi_j(\mathbf{x}) = \frac{1}{2}\|\mathcal{F}_j(\mathbf{x}) - \mathbf{y}_j\|^2_{\mathbf{\Gamma}^{-1}_{j,\text{noise}}}, \tag{2.9}$$

Algorithm 2.1 Gauss–Newton

- initialize $\mathbf{x}^{(0)}$
- for $k = 0, 1, 2, \ldots$
  1. compute $\Phi_1(\mathbf{x}^{(k)}), \ldots, \Phi_N(\mathbf{x}^{(k)})$ and $\nabla_{\mathbf{x}}\Phi_1(\mathbf{x}^{(k)}), \ldots, \nabla_{\mathbf{x}}\Phi_N(\mathbf{x}^{(k)})$
  2. solve system (2.11) to obtain $\delta\mathbf{x}$ using PCG
  3. set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \gamma\partial\mathbf{x}$, where γ is set by Armijo linesearch
  4. check convergence criteria

where $\mathcal{F}_j : \mathbb{R}^n \mapsto \mathbb{R}^{m_j}$ and $\mathbf{y}_j \in \mathbb{R}^{m_j}$ correspond to the jth forward operator and right-hand side, respectively, and $\Gamma_{j,\text{noise}} \in \mathbb{R}^{m_j \times m_j}$ is the noise covariance matrix corresponding to the jth misfit term. This allows us to rephrase (2.7) as

$$\mathbf{x}_{\text{MAP}} = \operatorname*{argmin}_{\mathbf{x}} \sum_{j=1}^{N} \Phi_j(\mathbf{x}) + \mathcal{R}(\mathbf{x}), \tag{2.10}$$

where, for simplicity, we assume that the different right-hand sides are statistically independent, which allows us to decouple the posterior. There are many ways to exploit the structure of (2.10), including stochastic optimization methods, e.g., stochastic approximation [37], stochastic average approximation [29], and the method of simultaneous sources [18]. Here, we are interested in deterministic optimization methods (potentially applied to a stochastic average approximation or the reduced problem in [18]). Common choices include steepest descent (SD), quasi-Newton methods such as l-BFGS [50], nonlinear conjugate gradient (NLCG) methods [20, 21], and GN methods [17, 50]. We limit the discussion in this section to the GN-PCG and NLCG methods.

When applying the GN-PCG algorithm to (2.10), the Hessian of an approximated objective function with linearized parameter-to-observable map is used as the coefficient matrix in the Newton system. The search direction, $\partial\mathbf{x} \in \mathbb{R}^n$, is computed approximately by applying a preconditioned conjugate gradient (PCG) method (see, e.g., [40]) to the linear system

$$\left(\sum_{j=1}^{N} \mathbf{J}_j^\top \boldsymbol{\Gamma}_{j,\text{noise}}^{-1} \mathbf{J}_j + \nabla_{\mathbf{x}}^2 \mathcal{R}(\mathbf{x}) \right) \partial\mathbf{x} = - \sum_{j=1}^{N} \nabla_{\mathbf{x}} \Phi_j(\mathbf{x}) - \nabla_{\mathbf{x}} \mathcal{R}(\mathbf{x}) \tag{2.11}$$

(see Algorithm 2.1, step 2), where $\mathbf{J}_j \in \mathbb{R}^{m_j \times n}$ is the Jacobian matrix of the jth parameter-to-observable map $\mathcal{F}_j$. Although the gradients and matrix vector products with the approximated Hessian in (2.11) can be computed in parallel, solving the linear system efficiently is nontrivial. To limit the communication overhead, one can use the static scheduling approach described in [39]. Here, the model and a number of meshes, sources, receivers, and forward problems are assigned to all the workers in the offline phase. Then, to evaluate the misfit and the gradient, and to perform a Hessian matrix-vector product, each worker computes its corresponding batch of gradients and Hessian matrix-vector products, and communicates it to the main process. Consequently, every inner PCG iteration used to solve (2.11) requires sending the current iterate for the search direction to the workers and receiving the results from local matrix-vector products. For large-scale problems this can result in a nontrivial amount of communication, especially when many PCG iterations are needed. Moreover, if the data is divided unevenly among the workers, the algorithm

Algorithm 2.2 NLCG

- initialize $\mathbf{x}^{(0)}$
- set $\mathbf{p}^{(0)} = -\nabla_{\mathbf{x}} f(\mathbf{x}^{(0)})$, where $f(\mathbf{x}) = \sum_{j=1}^{N} \Phi_j(\mathbf{x}) + \mathcal{R}(\mathbf{x})$
- for $k = 0, 1, 2, \ldots$
  1. update $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \gamma \partial \mathbf{x}^{(k)}$, where γ is set by a linesearch
  2. compute $\nabla_{\mathbf{x}} f(\mathbf{x}^{(k+1)})$
  3. set $\mathbf{d}^{(k)} = \nabla_{\mathbf{x}} f(\mathbf{x}^{(k+1)}) - \nabla_{\mathbf{x}} f(\mathbf{x}^{(k)})$
  4. compute $\beta^{(k)} = \dfrac{1}{(\partial \mathbf{x}^{(k)})^\top \mathbf{d}^{(k)}} \left(\mathbf{d}^{(k)} - \partial \mathbf{x}^{(k)} \dfrac{2\|\mathbf{d}^{(k)}\|^2}{(\partial \mathbf{x}^{(k)})^\top \mathbf{d}^{(k)}} \right) \nabla_{\mathbf{x}} f(\mathbf{x}^{(k+1)})$
  5. update $\partial \mathbf{x}^{(k+1)} = -\nabla_{\mathbf{x}} f(\mathbf{x}^{(k+1)}) + \beta^{(k)} \partial \mathbf{x}^{(k)}$
  6. check convergence criteria

may lead to large latencies in each PCG iteration [39]. This motivates us to consider more scalable distributed algorithms, especially when the size and dimension of the problem are very large.

The NLCG algorithm requires substantially less communication per outer iteration. NLCG performs explicit steps using gradients to update the model (see Algorithm 2.2) and therefore avoids the communication that comes with solving the GN system using an iterative method. In our experience, however, the method typically requires more iterations than the GN-PCG method in order to achieve a similar level of accuracy (see sections 4.2 and 4.3). Since each gradient and objective function evaluation requires at least N PDE solves, the large number of outer iterations renders the NLCG method less attractive for most large-scale PDE parameter estimation problems.

3. Uncertainty-weighted consensus ADMM. In this section, we introduce our uncertainty-weighted ADMM method. First, we present the general formulation of the weighted ADMM, which involves rephrasing (2.10) as a global variable consensus problem [6], and review the asynchronous implementation presented in [52]. We propose a novel scheme for selecting the weights, which is based on approximate uncertainty information of the local subproblems that we obtain similarly to the framework in [12]. Finally, we use a numerical example to illustrate the intuition behind the weights.

3.1. Weighted consensus ADMM. Motivated by the discussion in the previous section, we reformulate the optimization problem (2.10) as an equivalent weighted global variable consensus problem

$$\begin{aligned} \mathbf{x}_{\text{MAP}} = \underset{\mathbf{x}_1,\ldots,\mathbf{x}_N,\mathbf{z}}{\operatorname{argmin}} & \sum_{j=1}^{N} \left(\Phi_j(\mathbf{x}_j) + \mathcal{R}(\mathbf{x}_j) \right) \\ \text{s.t.} \quad & \mathbf{W}_j(\mathbf{x}_j - \mathbf{z}) = \mathbf{0}, \quad j = 1, \ldots, N, \end{aligned} \tag{3.1}$$

where, in contrast to (2.10), the objective function is now separable and the coupling is enforced in the constraints. Here, $\mathbf{x}_j \in \mathbb{R}^n$ are the local variables that are brought into consensus via the global variable $\mathbf{z} \in \mathbb{R}^n$, and $\mathbf{W}_j \in \mathbb{R}^{n \times n}$ are nonsingular weight matrices. For ease of presentation and to obtain an efficient optimization scheme, this work uses diagonal weight matrices. In the standard global consensus formulation [6], the identity matrix is assigned as the weight matrices. This reformulation allows each of the objective terms in (3.1) to be handled by its corresponding worker via the consensus ADMM algorithm.

Consider the augmented Lagrangian defined by

$$\begin{aligned}&\mathcal{L}_\rho(\mathbf{x}_1,\dots,\mathbf{x}_N,\mathbf{u}_1,\dots,\mathbf{u}_N,\mathbf{z})\\ &\qquad=\sum_{j=1}^{N}\Phi_j(\mathbf{x}_j)+\mathcal{R}(\mathbf{x}_j)+\mathbf{u}_j^\top(\mathbf{W}_j(\mathbf{x}_j-\mathbf{z}))+\frac{\rho}{2}\|\mathbf{W}_j(\mathbf{x}_j-\mathbf{z})\|_2^2.\end{aligned}\tag{3.2}$$

Consensus ADMM aims at solving problem (3.1) by finding a saddle point of $\mathcal{L}_\rho$ via the following iterations:

$$\begin{aligned}\mathbf{x}_j^{(k+1)}&=\operatorname*{argmin}_{\mathbf{x}_j}\mathcal{L}_\rho(\mathbf{x}_1^{(k)},\dots,\mathbf{x}_{j-1}^{(k)},\mathbf{x}_j,\mathbf{x}_{j+1}^{(k)},\dots,\mathbf{x}_N^{(k)},\mathbf{u}_1^{(k)},\dots,\mathbf{u}_N^{(k)},\mathbf{z}^{(k)})\\ &=\operatorname*{argmin}_{\mathbf{x}_j}\left(\Phi_j(\mathbf{x}_j)+\mathcal{R}(\mathbf{x}_j)+(\mathbf{u}_j^{(k)})^\top\mathbf{W}_j\mathbf{x}_j+\frac{\rho}{2}\|\mathbf{W}_j(\mathbf{x}_j-\mathbf{z}^{(k)})\|_2^2\right),\\ &\qquad j=1,\dots,N,\end{aligned}\tag{3.3}$$

$$\begin{aligned}\mathbf{z}^{(k+1)}&=\operatorname*{argmin}_{\mathbf{z}}\mathcal{L}_\rho(\mathbf{x}_1^{(k+1)},\dots,\mathbf{x}_N^{(k+1)},\mathbf{u}_1^{(k)},\dots,\mathbf{u}_N^{(k)},\mathbf{z})\\ &=\left(\sum_{j=1}^{N}\mathbf{W}_j^\top\mathbf{W}_j\right)^{-1}\sum_{j=1}^{N}\left(\mathbf{W}_j^\top\mathbf{W}_j\mathbf{x}_j^{(k+1)}+(1/\rho)\mathbf{W}_j\mathbf{u}_j^{(k)}\right),\end{aligned}\tag{3.4}$$

$$\mathbf{u}_j^{(k+1)}=\mathbf{u}_j^{(k)}+\rho\mathbf{W}_j(\mathbf{x}_j^{(k+1)}-\mathbf{z}^{(k+1)}),\quad j=1,\dots,N,\tag{3.5}$$

where k denotes the current iteration, $\mathbf{u}_j\in\mathbb{R}^n$ are the dual variables, and $\rho>0$ is the penalty parameter associated with the augmented Lagrangian term. In the first two steps, we have simplified the augmented Lagrangian by dropping all terms that enter the subproblems as constants. We note that the last step is a dual ascent step.

The minimization steps in (3.3) require PDE solves per function, gradient, and Hessian evaluations, and are the most computationally challenging part of the algorithm. However, they correspond to the local subproblems that are solved independently by each worker. Another advantage is that the local subproblem can be solved using any optimization algorithm, which provides an easy way to tailor the method to different subproblems, e.g., subproblems containing different PDEs for which highly optimized algorithms already exist. Consequently, ADMM sits at a higher level of abstraction than classical optimization algorithms, such as those mentioned in section 2.2. The global variable $\mathbf{z}$ attempts to bring the local variables $\mathbf{x}_j$ into consensus by averaging them in (3.4), and finally, the dual variables are updated via a gradient ascent step in (3.5).

We use the stopping criteria in [6] to define the primal and dual residuals as

$$\mathbf{r}^{(k+1)}=\left(\mathbf{W}_1(\mathbf{x}_1^{(k+1)}-\mathbf{z}^{(k+1)}),\dots,\mathbf{W}_N(\mathbf{x}_N^{(k+1)}-\mathbf{z}^{(k+1)})\right)\ \text{ and}\tag{3.6}$$

$$\mathbf{s}^{(k+1)}=-\rho\left(\mathbf{W}_1(\mathbf{z}^{(k+1)}-\mathbf{z}^{(k)}),\dots,\mathbf{W}_N(\mathbf{z}^{(k+1)}-\mathbf{z}^{(k)})\right),\tag{3.7}$$

respectively, and stop whenever

$$\|\mathbf{r}^{(k)}\|_2\leq\epsilon^{\text{pri}}\ \text{ and }\ \|\mathbf{s}^{(k)}\|_2\leq\epsilon^{\text{dual}}\tag{3.8}$$

for some chosen primal and dual tolerances ϵ^{pri} and ϵ^{dual}. It is also common to adaptively choose the penalty parameter ρ. We use the scheme in [6], i.e.,

$$\rho^{(k+1)}=\begin{cases}\tau^{\text{incr}}\rho^{(k)} & \text{if}\quad \|\mathbf{r}^{(k)}\|_2>\mu\|\mathbf{s}^{(k)}\|_2,\\ \rho^{(k)}/\tau^{\text{decr}} & \text{if}\quad \|\mathbf{s}^{(k)}\|_2>\mu\|\mathbf{r}^{(k)}\|_2,\\ \rho^{(k)} & \text{otherwise,}\end{cases}\tag{3.9}$$

Algorithm 3.1 Consensus ADMM

- initialize $\mathbf{x}_j^{(0)}, \mathbf{z}^{(0)}$, and $\mathbf{u}_j^{(0)}$ for $j = 1, \ldots, N$
- for $k = 0, 1, 2, \ldots$ until (3.8) holds
  1. obtain $\mathbf{x}_j^{(k+1)}$ by solving local problems in (3.3) for $j = 1, \ldots, N$
  2. obtain $\mathbf{z}^{(k+1)}$ using the averaging step (3.4)
  3. obtain $\mathbf{u}_j^{(k+1)}$ through dual update (3.5) for $j = 1, \ldots, N$

Algorithm 3.2 Consensus async-ADMM

- initialize $\mathbf{x}_j^{(0)}, \mathbf{z}^{(0)}$, and $\mathbf{u}_j^{(0)}$ for $j = 1, \ldots, N$
- initialize N_a and k_a
- while (3.8) not satisfied
  1. solve (3.3) locally
  2. perform averaging step (3.4) when N_a workers report their solutions
  3. update the corresponding N_a dual variables (3.5)

where $\mu > 1, \tau^{\text{incr}} > 1$, and $\tau^{\text{decr}} > 1$ are parameters commonly chosen to be $10, 2$, and 2, respectively [6]. This updating scheme aims at balancing the primal and dual residual norms within a factor of μ of each other as they both converge to zero.

Parallelization of consensus ADMM is much more straightforward than that of the GN-PCG described in section 2.2. The amount of communication per outer iteration is reduced as we only communicate one set of models, $\mathbf{x}_1, \ldots, \mathbf{x}_N$ per outer ADMM iteration. In the synchronous parallel implementation, the master processor must wait for all the workers to finish solving their corresponding subproblems in (3.3) before performing the averaging step (3.4) per iteration, which may lead to high latencies when some of the workers are much slower than others. The async-ADMM method in [52] aims at reducing these latencies in star network topologies. Here, the global averaging step (3.4) is performed when $N_a < N$ workers report their results. A *bounded delay* condition is also enforced, where every worker has to report at least once every k_a iterations to ensure sufficient "freshness" of all updates. We note that here we have better control of the overall amount of communication and latency since we can administer how many forward problems to assign to any given worker and how accurately to solve each subproblem.

Convergence results have been established for the synchronous ADMM algorithm in the case where the local subproblems are convex. In this case, the algorithm converges regardless of the initial choice $\rho^{(0)}$ [10, 25]. Even when solving (3.3) inexactly, ADMM convergence can be shown [25, sect. 4]. For the asynchronous case, convergence is ensured via the bounded delay condition. For nonconvex subproblems, it has been shown that ADMM converges to a local minimum under some modest assumptions, most importantly requiring ρ to be sufficiently large [26, 33, 50]. These assumptions ensure that the Hessian of the Lagrangian of (3.1) remains positive definite throughout the ADMM iterations.

3.2. Weight selection. We choose the weights to be approximately equal to the inverse of the diagonals of the posterior covariance $\Gamma_{j,\text{post}} \in \mathbb{R}^{n \times n}$ corresponding to the jth objective term in (3.1). This is one way to assign higher weights to elements of $\mathbf{x}_j$ for which the jth subproblem contains more information. It also reduces the impact of elements for which the data of the subproblem is uninformative. Clearly, there are other options for transforming uncertainties into weights. Since we are mostly

interested in encoding large differences in the uncertainties between subproblems, we do not compute the uncertainties with high accuracy.

As seen in (2.8), construction of the posterior covariance may not be tractable, especially for large-scale PDE parameter estimation problems and when the forward model is nonlinear. As a result, we follow the works of [12] for approximating the posterior covariance of each objective term in a tractable way. This is done via a low-rank approximation of the approximate Hessian of the misfit Φ_j in the following manner:

1. We linearize the residual in Φ_j and obtain the GN approximation
$$\mathbf{H}_{j,\text{mis}} \approx \mathbf{J}_j^\top (\mathbf{\Gamma}_{j,\text{noise}}^{-1}) \mathbf{J}_j, \tag{3.10}$$
where $\mathbf{J}_j \in \mathbb{R}^{m_j \times n}$ is the Jacobian matrix of $\mathcal{F}_j$ evaluated at some reference model parameter, e.g., $\mathbf{x}_{\text{ref}}$. We note that explicit construction of $\mathbf{H}_{j,\text{mis}}$ is not necessary as we only need the action of $\mathbf{J}_j$ and $\mathbf{J}_j^\top$ on a vector.
2. Denoting the prior-conditioned approximate Hessian by
$$\tilde{\mathbf{H}}_{j,\text{mis}} = \mathbf{\Gamma}_{\text{prior}}^{1/2} \mathbf{H}_{j,\text{mis}} \mathbf{\Gamma}_{\text{prior}}^{1/2},$$
we rewrite the jth posterior covariance in (2.8) as
$$\mathbf{\Gamma}_{j,\text{post}} = \mathbf{\Gamma}_{\text{prior}}^{1/2} \left(\tilde{\mathbf{H}}_{j,\text{mis}} + \mathbf{I} \right)^{-1} \mathbf{\Gamma}_{\text{prior}}^{1/2}. \tag{3.11}$$
3. We then construct a low-rank approximation of the prior-conditioned Hessian using, e.g., randomized SVD [41] or Lanczos bidiagonalization [16] to obtain
$$\tilde{\mathbf{H}}_{j,\text{mis}} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top \approx \mathbf{V}_r \mathbf{\Lambda}_r \mathbf{V}_r^\top, \tag{3.12}$$
where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n \times n}$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n \times n}$ denote the matrix of eigenvalues and eigenvectors of $\tilde{\mathbf{H}}_{j,\text{mis}}$, respectively, and $\mathbf{\Lambda}_r = \text{diag}(\lambda_1, \ldots, \lambda_r) \in \mathbb{R}^{r \times r}$ and $\mathbf{V}_r = [\mathbf{v}_1, \ldots, \mathbf{v}_r] \in \mathbb{R}^{n \times r}$ are their corresponding truncations retaining only the r largest eigenvalues and eigenvectors.
4. We plug this approximation into (3.11) and use the Sherman–Morrison–Woodbury formula [43] to obtain an expression for the inverse term,
$$\left(\tilde{\mathbf{H}}_{j,\text{mis}} + \mathbf{I} \right)^{-1} \approx \mathbf{I} - \mathbf{V}_r \mathbf{D}_r \mathbf{V}_r^\top + \mathcal{O}\left(\sum_{i=r+1}^{n} \frac{\lambda_i}{\lambda_i + 1} \right), \tag{3.13}$$
where $\mathbf{D} \in \mathbb{R}^{r \times r} = \text{diag}(\lambda_1/(\lambda_1+1), \ldots, \lambda_r/(\lambda_r+1))$.
5. Finally, we obtain the following manageable approximation of the posterior covariance that does not involve the inverse of the Hessian but that instead involves the square root of the prior:
$$\mathbf{\Gamma}_{j,\text{post}} \approx \mathbf{\Gamma}_{\text{prior}}^{1/2} (\mathbf{I} - \mathbf{V}_r \mathbf{D}_r \mathbf{V}_r^\top) \mathbf{\Gamma}_{\text{prior}}^{1/2}. \tag{3.14}$$

We choose the weights to be the inverse of the diagonals of $\mathbf{\Gamma}_{j,\text{post}}$,

$$\begin{aligned} \mathbf{W}_j &= \text{diag}(\mathbf{\Gamma}_{j,\text{post}})^{-1}, \\ &\approx \text{diag}\left(\mathbf{\Gamma}_{\text{prior}} - \mathbf{\Gamma}_{\text{prior}}^{1/2} \left(\mathbf{V}_r \mathbf{D}_r \mathbf{V}_r^\top \right) \mathbf{\Gamma}_{\text{prior}}^{1/2} \right)^{-1}, \\ &= \left[\text{diag}\left(\mathbf{\Gamma}_{\text{prior}} \right) - \text{diag}\left(\mathbf{\Gamma}_{\text{prior}}^{1/2} \left(\mathbf{V}_r \mathbf{D}_r \mathbf{V}_r^\top \right) \mathbf{\Gamma}_{\text{prior}}^{1/2} \right) \right]^{-1}, \quad j = 1, \ldots, N, \end{aligned} \tag{3.15}$$

so that we get higher weights in parts of the model where we are more certain, and vice versa. We note that to compute the diagonals of $\mathbf{\Gamma}_{j,\text{post}}$, we need to be able to multiply by $\mathbf{\Gamma}_{\text{prior}}^{1/2}$ and compute the diagonals of $\mathbf{\Gamma}_{\text{prior}} = (\mathbf{L}^\top\mathbf{L})^{-1}$ efficiently. In our experiments, we are mainly concerned with the case when $\mathbf{L}^\top\mathbf{L}$ is either a diagonal or a biharmonic operator. In the diagonal case, multiplying by $\mathbf{\Gamma}_{\text{prior}}^{1/2}$ and computing the diagonals of $\mathbf{\Gamma}_{\text{prior}}$ is trivial. In the case where we use the biharmonic operator, we have access to the spectral decomposition of $\mathbf{L}^\top\mathbf{L}$ using Fourier transforms [23], which allows us to efficiently multiply by $\mathbf{\Gamma}_{\text{prior}}^{1/2}$. We can also quickly estimate the diagonals of $\mathbf{\Gamma}_{\text{prior}}$ using probing methods [46], extrapolation methods [11], stochastic methods [2], and domain decomposition methods [30, 31, 45]. We may also update the weights throughout the ADMM scheme so that we instead employ local approximations of our posterior PDF [7]; however, convergence in this case is not guaranteed. When the nonzero diagonal elements of $\mathbf{W}_j$ are equal to one, the weighted ADMM method corresponds to the standard unweighted ADMM scheme, which is known to converge slowly [6]. One reason is because the averaging step in (3.4) gives equal weight to all elements of $\mathbf{x}_j$ for all $j = 1, \ldots, N$, leading to poor reconstructions of $\mathbf{z}$, especially in the early iterations. To illustrate this, we perform the following example.

Example 1. Consider solving the trivial linear system $\mathbf{Ix} = \mathbf{y}$ with the weighted and unweighted consensus ADMM with $N = 4$ splittings, where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix, and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are the model and the observed data, respectively. We formulate the least-squares problem as

$$\underset{\mathbf{x}_j,\mathbf{z}}{\text{argmin}} \sum_{j=1}^{4} \left(\frac{1}{2}\|\mathbf{I}_j\mathbf{x}_j - \mathbf{y}_j\|_2^2 + \frac{\alpha}{2}\|\mathbf{x}_j\|_2^2 \right) \tag{3.16}$$

$$\text{s.t.} \quad \mathbf{W}_j(\mathbf{x}_j - \mathbf{z}) = \mathbf{0}, \quad j = 1, \ldots, 4, \tag{3.17}$$

where $\alpha = 10^{-2}$, and $\mathbf{I}_j \in \mathbb{R}^{(n/4)\times n}$ and $\mathbf{y}_j \in \mathbb{R}^{n/4}$ are subsets of the data obtained by partitioning the rows of $\mathbf{I}$ and $\mathbf{y}$ corresponding to the pixels in the top left, top right, bottom left, and bottom right quadrant of the domain as seen in Figure 3.1. We show the averaged reconstruction of both methods during the first iteration in Figure 3.1.

We note that the forward model in Example 1 is separable, and the weights can intuitively and easily be designed by hand. This example shows that we have a principled way to construct the weighted scheme that corresponds to manually choosing the weights in the cases that are as obvious as this example. In general, however, it is not always possible to manually design the weights, as the forward operators are not always separable.

4. Numerical experiments. In this section, we outline the potential of the weighted scheme for consensus ADMM as well as its asynchronous variant on a series of linear and nonlinear inverse problems. We first experiment on a deblurring and a tomography problem from `Regtools`, a MATLAB package containing discrete ill-posed inverse problems [22], as well as from a collection of linear least-squares problems from the UF (University of Florida) Sparse Matrix Collection [9]. We then test our method on the following larger 3D PDE parameter estimation problems: a single-physics parameter estimation problem involving a travel time tomography survey, and a multiphysics parameter estimation problem involving DCR and travel time tomography. We conclude this section with a comparison in the communication among the algorithms used for the multiphysics problem.

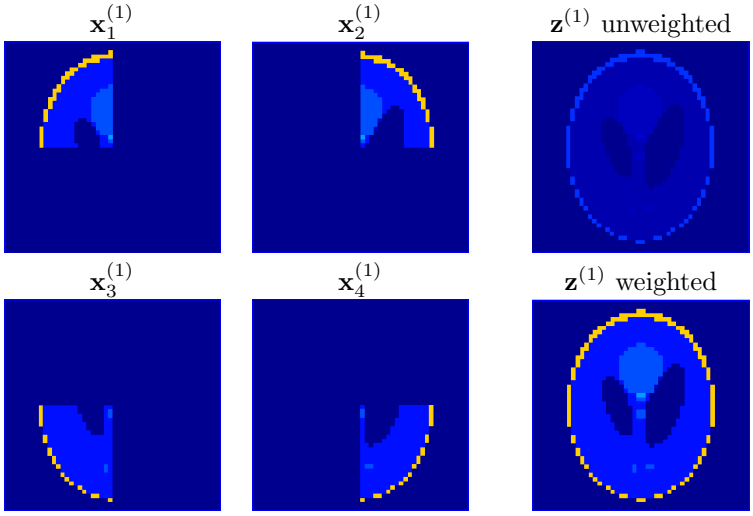

FIG. 3.1. *Averaging step of the weighted and unweighted consensus ADMM for Example* 1. *In this case,* $\mathbf{W}_1$ *assigns higher weights to the pixels in the upper left quadrant of* $\mathbf{x}_1$, $\mathbf{W}_2$ *assigns higher weights to the upper right quadrant of* $\mathbf{x}_2$, *etc. As a result, the weights educate the averaging step, leading to a better reconstruction of the image.*

4.1. Least-squares. We begin by comparing the weighted and unweighted consensus ADMM on a series of linear least-squares problems from `Regtools` [22] and the UF Library Sparse Matrix Collection [9]. For these problems, we use $N = 4$ splittings and solve

$$\begin{aligned} \underset{\mathbf{x}_j,\mathbf{z}}{\text{argmin}} \quad & \sum_{j=1}^{4} \left(\frac{1}{2}\|\mathbf{A}_j\mathbf{x}_j - \mathbf{y}_j\|_2^2 + \frac{\alpha}{2}\|\mathbf{x}_j\|_2^2 \right) \\ \text{s.t.} \quad & \mathbf{W}_j(\mathbf{x}_j - \mathbf{z}) = \mathbf{0}, \quad j = 1, \ldots, 4, \end{aligned} \tag{4.1}$$

where similarly to Example 1, $\mathbf{A}_j \in \mathbb{R}^{(m/4)\times n}$ and $\mathbf{y}_j \in \mathbb{R}^{m/4}$, $j = 1, \ldots, 4$, are chosen by partitioning the rows of the original matrix and the data, $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{y} \in \mathbb{R}^m$, respectively. For the deblurring and tomography problems from `Regtools`, we use the same splittings as in Figure 3.1, where we split the rows corresponding to the different quadrants of the image. For the non–image-based problems from the UF library, $\mathbf{A}_1$ and $\mathbf{y}_1$ correspond to the first $m/4$ rows of $\mathbf{A}$ and $\mathbf{y}$, respectively, $\mathbf{A}_2$ and $\mathbf{y}_2$ correspond to the second $m/4$ rows of $\mathbf{A}$ and $\mathbf{y}$, respectively, and so on. In the case when the number of rows, m, is not divisible by 4, we round accordingly.

We add a smallness regularization term with $\alpha = 10^{-2}$ since the splittings $\mathbf{A}_j$ in our experiments are underdetermined ($m/4 < n$), leading to rank-deficient coefficient matrices $\mathbf{A}_j^\top \mathbf{A}_j$ arising from the normal equations. We set the initial penalty parameter to $\rho^{(0)} = 5$ and use the adaptive scheme described in (3.9). We run the unweighted and weighted consensus ADMM for 10 iterations and show comparisons of the relative residuals and relative errors. To compute the weights, we follow the procedure in section 3.2 and compute a rank-10 approximation of the Hessian of the misfits using the MATLAB function `eigs`.

We observe larger performance gains through the weighted ADMM in the deblurring problem compared to the tomography problem. In the deblurring problem, the weights are concentrated in different non–overlapping parts of the domain (see Figure

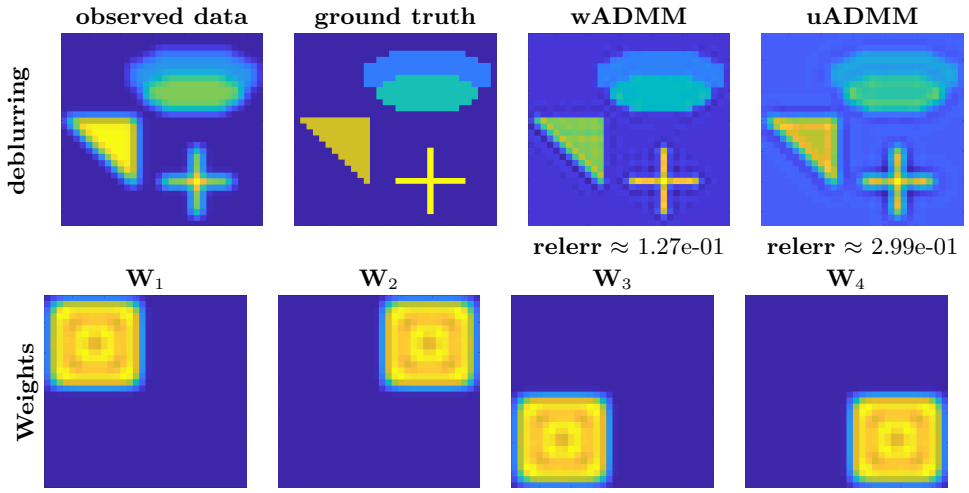

FIG. 4.1. *Observed data, ground truth, and reconstructions (first row) after* 10 *iterations and weights (second row) for the deblurring problem from* `Regtools` [22].

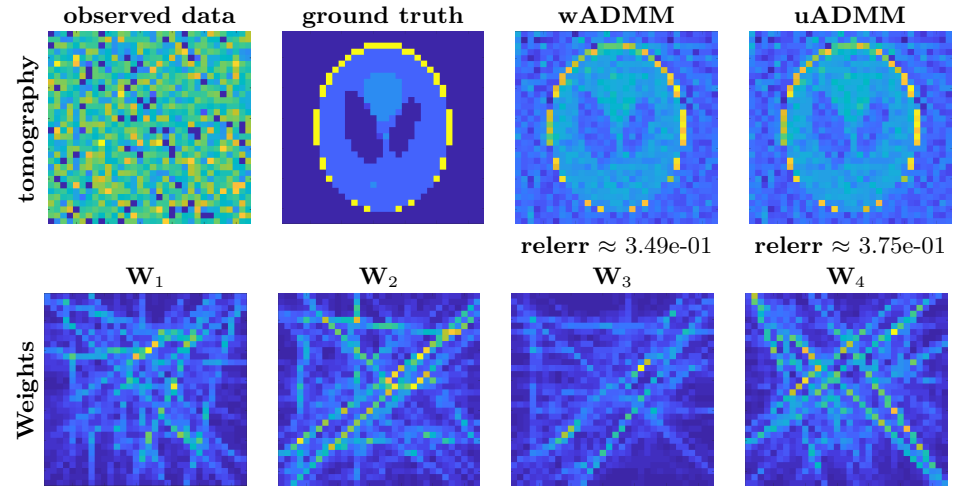

FIG. 4.2. *Observed data, ground truth, and reconstructions (first row) after* 10 *iterations and weights (second row) for the tomography problem from* `Regtools [22]`.

4.1), leading to more efficient averaging. For the tomography problem, however, the weights look similar and contain a substantial amount of overlap, leading to averaged reconstructions that are similar to those of the unweighted ADMM (see Figure 4.2).

Finally, for the UF matrices, we randomly take 30 matrices with dimensions $100 \leq m, n \leq 1000$ from the library and compare both methods in Table 4.1 after 10 iterations. We report their condition numbers, relative residuals, and relative errors. We obtain better results with the weighted ADMM after 10 iterations. We refrain from solving these problems in parallel since they are small 2D problems and are mainly used as a proof-of-concept.

4.2. Single-physics parameter estimation. As a more realistic test problem, we consider the 3D SEG/EAGE model [1] as the ground truth (see Figure 4.4b) and

TABLE 4.1

Comparison of the accuracy obtained using the unweighted and weighted ADMM applied to least-squares problems from the UF Sparse Matrix Collection [9]. *The first and second columns show the name and condition number of the matrices. The third and fourth columns show the relative residuals of the unweighted and weighted ADMM at iteration* 10, *respectively. The fifth and sixth columns show the relative errors of the unweighted and weighted ADMM at iteration* 10, *respectively.*

UF Sparse Matrix Collection Results					
		Unweighted ADMM		Weighted ADMM	
Matrix	Cond #	Residual	Relative error	Residual	Relative error
bcspwr03	5.01e+02	4.47e-02	2.58e-01	1.97e-02	1.63e-01
bcsstk03	6.79e+06	6.67e-01	9.99e-01	5.82e-01	9.91e-01
bcsstk19	1.34e+11	2.81e-01	9.01e-01	6.30e-02	8.62e-01
bfwb782	1.81e+01	4.94e-02	1.04e-01	2.83e-01	1.38e-01
can_229	4.01e+17	5.13e-02	2.10e-01	1.81e-02	1.46e-01
cavity02	8.12e+04	6.07e-01	9.33e-01	2.46e-01	7.83e-01
cavity03	5.85e+05	5.69e-01	9.01e-01	1.90e-01	7.34e-01
ch5-5-b4	1.00e+00	1.21e-01	9.85e-01	5.01e-03	9.84e-01
dwt_307	2.35e+18	8.92e-02	1.82e-01	2.23e-02	9.20e-02
football	3.74e+02	5.90e-02	4.88e-01	2.44e-02	3.58e-01
fs_183_3	3.27e+13	7.33e-02	1.00e+00	2.32e-02	1.00e+00
G23	1.00e+04	2.27e-02	2.63e-01	1.88e-02	2.55e-01
GD98_c	9.87e+16	7.78e-02	3.70e-01	5.13e-02	2.62e-01
gre_115	4.97e+01	2.77e-01	4.70e-01	7.64e-02	2.67e-01
gre_343	1.12e+02	1.18e-01	1.77e-01	4.48e-02	6.90e-02
grid1_dual	3.35e+16	3.74e-02	3.67e-01	2.56e-02	3.20e-01
impcol_d	2.06e+03	3.50e-01	7.07e-01	9.74e-02	3.94e-01
jpwh_991	1.42e+02	1.89e-01	8.81e-01	1.61e-01	8.66e-01
lowThrust_1	Inf	4.40e-01	9.96e-01	2.84e-01	9.82e-01
lund_a	2.80e+06	1.06e-01	5.99e-01	4.05e-02	5.65e-01
nos3	3.77e+04	5.27e-01	9.88e-01	2.22e-01	9.67e-01
odepa400	2.26e+05	4.91e-01	9.99e-01	1.86e-01	9.97e-01
pde900	1.53e+02	5.98e-01	9.82e-01	2.90e-01	9.39e-01
poisson2D	1.33e+02	3.33e-01	7.44e-01	8.03e-02	6.64e-01
polbooks	7.20e+02	3.72e-02	2.97e-01	2.30e-02	2.51e-01
problem1	3.11e+16	4.08e-01	9.11e-01	1.43e-01	8.24e-01
rdb200l	1.33e+02	8.68e-02	1.44e-01	1.91e-02	1.05e-01
str˙0	2.74e+02	8.46e-01	9.37e-01	4.04e-01	7.02e-01
TF10	7.34e+02	2.63e-01	4.82e-01	5.21e-02	2.85e-01
young1c	4.15e+02	6.39e-01	9.40e-01	3.59e-01	6.97e-01

test our method for a single-physics inversion involving the travel time tomography survey. The model contains a salt dome in which the velocity is significantly higher than in the background. The domain is of size 13.5 km × 13.5 km × 4.2 km and is divided into $64 \times 64 \times 32$ mesh cells equally sized at approximately $211\text{m} \times 211\text{m} \times 11\text{m}$. We implement our experiments in an extension of `jInv` [39], an open-source package for PDE parameter estimation written in Julia [4]. For brevity, since the travel time tomography problem is modeled by the Eikonal equation, we refer to it as the Eikonal problem for the remainder of the paper. We solve these problems in parallel and experiment on the effect of the asynchronous variant (async-ADMM) on the weighted and unweighted consensus ADMM.

The PDE involved in the forward problem is the Eikonal equation (see Table 4.2), and it is solved using the Factored Eikonal Fast Marching Algorithm [48]. We solve the inversion using 36 sources and 3600 receivers located on the top surface of the domain. We compare the weighted and unweighted ADMM (wADMM and uADMM), their

TABLE 4.2

PDEs corresponding to two different geophysical imaging techniques: (a) *DCR, and* (b) *travel time tomography. Here,* $u_j : \Omega \to \mathbb{R}$ *is the potential field that evolves from the* j*th source,* $q_j : \Omega \to \mathbb{R}$*, which is a dipole placed on the earth's surface, and* x_0 *is the origin. In the travel time tomography experiment,* $|\cdot|$ *is the Euclidean norm, and* $\tau_j : \Omega \to \mathbb{R}$ *is the travel time of the wave that propagates from a point source located at* x_j*, for which the travel time is* 0*. To jointly fit travel time and DC-resistivity data, we assume known petrophysics* [42]*, so that we have a relation between the wave velocity* v *and the ground conductivity* σ *as shown in* (4.2).

(a) DC resistivity		(b) Travel time tomography	
$\nabla \cdot \Big(\sigma(v(x))\nabla u_j\Big) = q_j(x)$	in Ω	$\lvert\nabla\tau_j\rvert^2 = v(x)$	in Ω
$\nabla u_j \cdot \vec{n} = 0$	on $\partial\Omega$	$\tau_j(x_j) = 0$	
$u_j(x_0) = u_{j,0}$			

asynchronous variants (async-wADMM and async-uADMM), GN, and NLCG. For all six algorithms, we use a biharmonic regularization with regularization parameter $\alpha = 10^{-1}$ to enforce smoothness. We solve all inversions in parallel using 10 workers. Here, six workers solve forward problems containing four sources each, and the remaining four workers solve forward problems containing three sources each.

We run the GN inversion for a maximum of 30 outer iterations and use at most 10 PCG iterations with PCG stopping tolerance of 10^{-1} to solve the GN system. For the NLCG inversion, we set a maximum of 100 outer iterations since it is expected to take more iterations than GN to reach the same accuracy. In the ADMM inversions, we run a total of 10 outer iterations with three GN iterations used to solve the subproblems. This particular choice of inner GN and outer ADMM iterations aims to balance the runtime and computations performed with those of the GN inversion while avoiding solving the subproblems too inexactly, as this may lead to a lack of convergence. In the ADMM subproblems, each GN iteration also uses at most 10 PCG iterations with PCG stopping tolerance of 10^{-1} as in the GN inversion.

For the penalty parameter, we use the scheme described in (3.9) to vary ρ and use a lower bound of 10^{-12}. As expected, the performance of ADMM depends crucially on the initial choice of ρ; therefore, we report the best results obtained from initial values of $\rho^{(0)} \in [10^{-8}, 10^{2}]$. In our experiment, the optimal initial values are $\rho^{(0)} = 10^{-7}$ for uADMM and $\rho^{(0)} = 10^{-4}$ for wADMM. In the asynchronous case, we perform a global update whenever $N_a = 5$ workers report their solutions, and we enforce the bounded delay condition by requiring all workers to report results at least once every $k_a = 4$ iterations. To compute the weights, we follow the procedure described in section 3.2, where we use the Lanczos bidiagonalization algorithm from `KrylovMethods` [38] to compute a rank-5 approximation of the approximate Hessians of the data misfits. To estimate the diagonal of the prior covariance, we perform 1000 iterations of the stochastic estimator proposed in [2] based on Hutchinson's technique for estimating the trace of a matrix. These iterations can be performed very efficiently as we have the spectral decomposition of the biharmonic operator. Again, we note that highly accurate uncertainties are not necessary in our case, and a good guess is sufficient for our experiments. The computation of the weights took about 31 seconds.

In Figure 4.3(a),(b), we show the relative errors and misfits for the Eikonal problem. Although the relative error does not improve during the iterations for any of the methods, we point out that the reconstructions visually become similar to the ground truth; see Figure 4.4(c),(h). The impact of communication and latency on the difference in runtimes between asynchronous ADMM variants, which ran for about 15 minutes, and the GN-PCG, which ran for about 38 minutes, is evident. For the

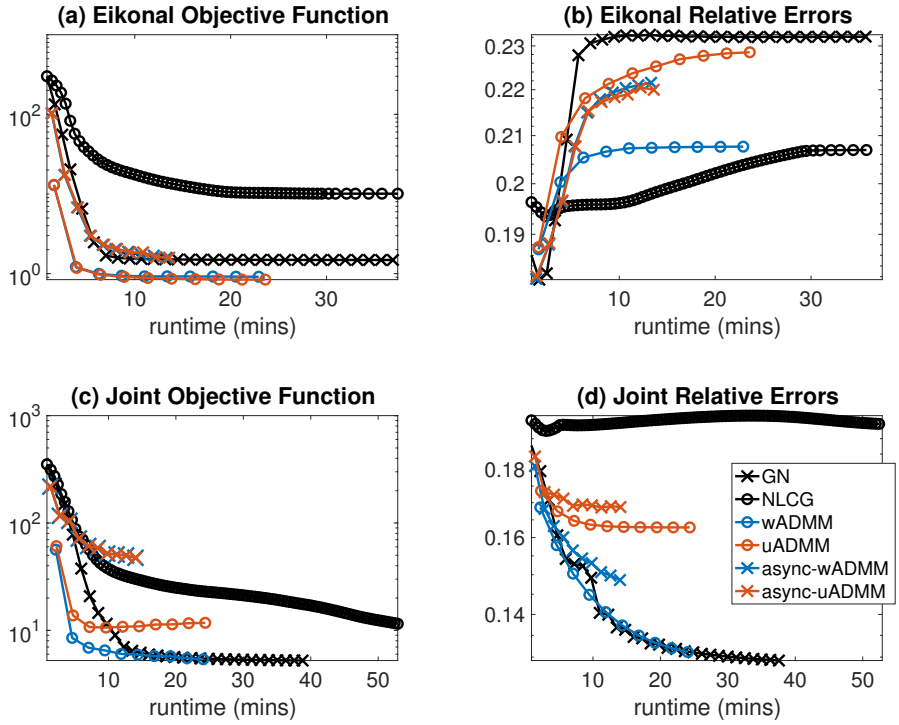

FIG. 4.3. *Objective function and relative errors for the Eikonal and joint inversions using six different algorithms: GN, NLCG, uADMM, wADMM, async-uADMM, and async-wADMM. Here, the x-axis represents runtime in minutes. The experiments were run on a shared memory computer operating Ubuntu* 14.04 *with two Intel Xeon E*5-2670 *v*32.3 *GHz CPUs, each using* 12 *cores and a total of* 128 *GB of RAM. Here, Julia is installed and compiled using the Intel Math Kernel Library.*

NLCG, a total of 68 iterations were performed before a linesearch fail was reached. As expected, an iteration from the NLCG method is much quicker than an iteration from the remaining five methods since each NLCG iteration only requires explicit steps to update the model.

4.3. Multiphysics parameter estimation. We now add a second modality to section 4.2, the DCR survey, which is modeled by the steady-state heterogeneous diffusion equation (see Table 4.2), and consider a multiphysics inversion. Here, we keep the same settings for the Eikonal problem and use 32 sources and 1682 receivers located on the top surface of the domain for the DCR survey. To solve the DCR forward problem, we use the finite volume method described in [17] to discretize the problem, and we solve the linear system using Julia's direct solver. For simplicity, we assume known petrophysics [42], which gives us an explicit relation between the ground conductivity σ and the wave velocity v given by

$$\sigma(v) = \left(2 - \frac{v}{c}\right)\left(\frac{b-a}{2}(\tanh(10(c-v)) + 1) + a\right). \tag{4.2}$$

Here, a and b are the conductivity values set to 0.1 and 1.0, respectively, and $c = 3.0$ is the velocity in which the contrast is centered. This setup was also used in [39].

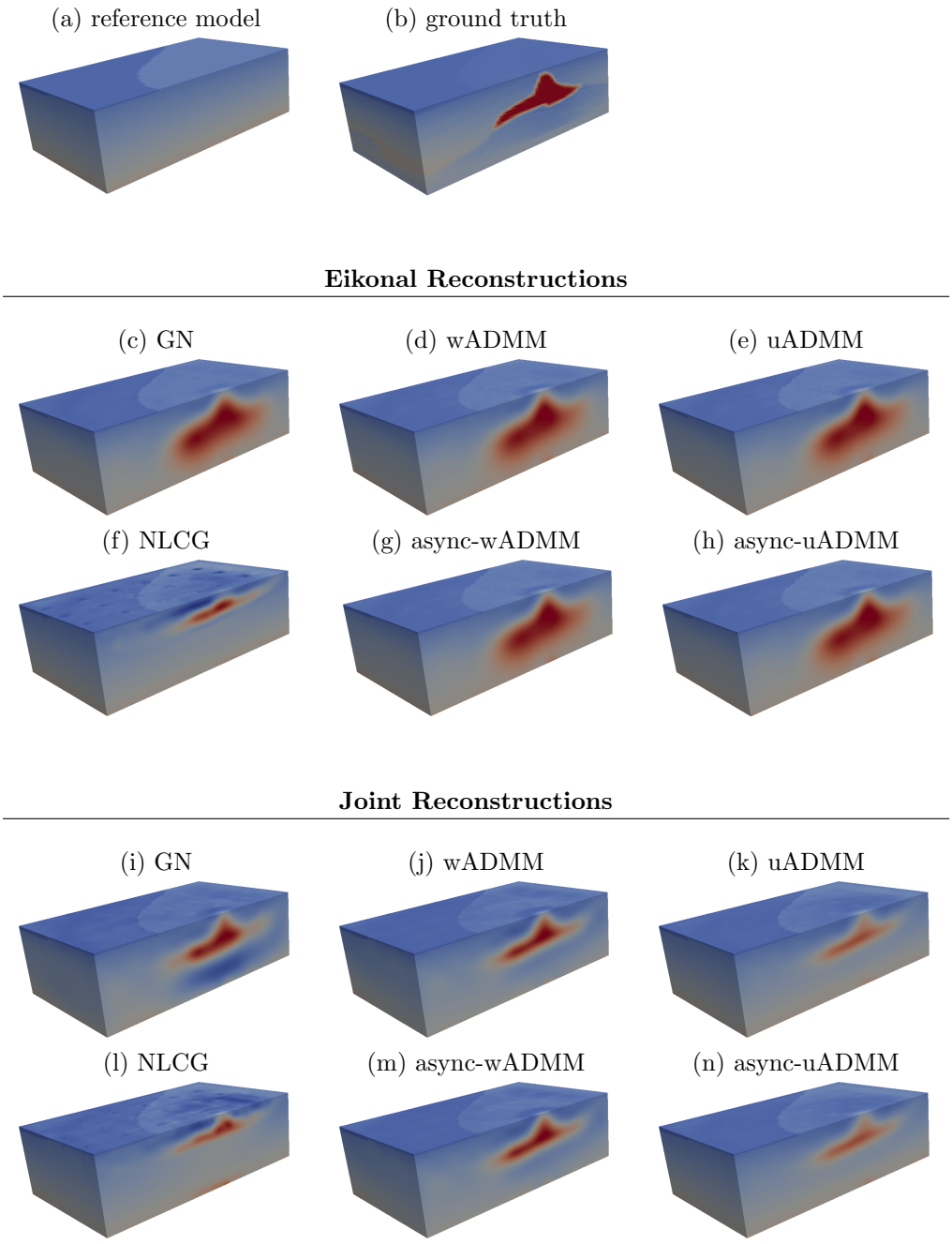

FIG. 4.4. *Reconstructions of SEG model with single-physics and multiphysics experiments.*

As in section 4.2, we compare six algorithms: wADMM, uADMM, async-wADMM, async-uADMM, GN-PCG, and NLCG. We solve all the inversions in parallel using ten workers. The PDE operator in the DCR experiment is small enough to be factorized in a single worker with a direct solver. In this case, it is not worth parallelizing the problem since communication takes longer than solving for all sources in one worker. For larger problems, where the DCR problem must instead be solved iteratively, however, distributing the DCR sources among different workers will lead to faster time-to-solution. In contrast, the Eikonal problems are solved with a sequential fast marching scheme [48], and thus we distribute the problems among the remaining nine workers. The nine workers in charge of the Eikonal problem solve forward problems, each containing four sources. The inversion settings are also the same as in section 4.2 except for the choice of initial penalty parameter, where we find the optimal initial values to be $\rho^{(0)} = 10^{-3}$ for uADMM and $\rho^{(0)} = 1.0$ for wADMM. We also follow the same procedure as in section 4.2 to compute the weights for this setup, which took about 54 seconds.

We show the results for the relative errors and objective function values vs. runtime for the joint inversions in Figure 4.3 and the reconstructions in Figure 4.4. We see that the weighted schemes improve the convergence and reconstruction quality of its unweighted counterparts. In fact, we obtain similar reconstruction accuracy between the GN scheme and the synchronous weighted ADMM (see Figure 4.3(d)); we highlight the considerable progress made by the ADMM method in the first few iterations, as this is not very common for small problems that can be held in the memory of a few machines. We also obtain faster convergence using the asynchronous ADMM variants, with the weighted asynchronous ADMM leading to a similar quality of reconstruction, as can be seen in Figure 4.4. As expected, the joint inversions enhance the quality of the reconstruction since the different physics involved capture different properties of the model [39].

4.4. Communication costs. We use the multiphysics example to exemplify the differences in terms of communication costs for the GN, NLCG, and ADMM methods. As discussed in section 4.3, we assign worker 1 the DCR problem containing all of its 32 forward models, and assign each of the remaining nine workers (2–10) four Eikonal forward models. In the comparison below, the vectors communicated between the workers and the master process are of size 131072×1.

GN. We use 30 GN iterations, each of which involves up to 10 PCG iterations per GN iteration. In each GN iteration, the master process sends the current model to all ten workers. Worker 1 then returns the accumulated gradient vector corresponding to the 32 DCR forward problems, and each of workers 2–10 returns the accumulated gradient vector for the four local sources. This allows for the computation of the full gradient shown in (2.11).

Moreover, *in each PCG iteration*, the master process sends one vector to all workers. The workers then return the accumulated matrix-vector product of this vector with the approximated Hessians associated with the local subproblems (DCR for worker 1 and Eikonal for workers 2–10). This leads to a total of 20 vectors communicated between the master and the workers per PCG iteration. Overall, the GN method consists of

$$10 \text{ workers} \times (2 \text{ gradient vectors} + 20 \text{ PCG vectors}) \times 30 \text{ GN iterations} = 6600$$

vectors communicated between the workers and the master process.

NLCG. We run NLCG for a maximum of 100 iterations. The communication pattern is the same as in the GN method except that no inner PCG iterations are performed. Thus, the NLCG inversion consists of

$$10 \text{ workers} \times 2 \text{ gradient vectors} \times 100 \text{ NLCG iterations} = 2000$$

vectors communicated between the workers and the master process.

sync-ADMM. We run a total of 10 outer ADMM iterations. In each ADMM iteration, the master process sends the current global variable $\mathbf{z}^{(k)}$ to all workers. Each worker then returns its corresponding local variable $\mathbf{x}_j^{(k+1)}$ to the main process. This leads to a total of two vectors communicated between each worker and the master process per ADMM iteration. The sync-ADMM inversion thus consists of

$$10 \text{ workers} \times 2 \text{ vectors} \times 10 \text{ ADMM iterations} = 200$$

total vectors communicated between the workers and the master process.

async-ADMM. As in sync-ADMM, we run a total of 10 ADMM iterations. However, we update the global variable whenever four workers report their solution. As a result, the async-ADMM inversion consists of

$$4 \text{ workers} \times 2 \text{ vectors} \times 10 \text{ ADMM iterations} = 80$$

total vectors communicated between the workers and the master process.

The communication comparison described above confirms that ADMM dramatically reduces the amount of communication in the inversion. This is also seen in the reduced runtimes in Figure 4.3. A similar comparison can be made for the single-physics parameter estimation problem in section 4.2.

5. Conclusion. We propose a weighted asynchronous consensus ADMM (async-wADMM) method for solving large-scale PDE parameter estimation problems in parallel. To this end, the data involved in the problem is divided among the available workers. Our scheme is geared toward applications such as PDE parameter estimation, where only a few iterations can be afforded. Our proposed weighting scheme improves the convergence of the standard ADMM. Since our weights are informed by an approximate uncertainty quantification for the subproblems in (3.3), we formulate the parameter estimation problem in a Bayesian setting. It is important to note that our scheme can also be applied in the frequentist setting as long as weights are available. To obtain an overall efficient scheme, we follow the work of [12] to quantify the uncertainties in a tractable manner.

As test problems, we use a collection of linear least-squares problems for proof-of-concept, a more realistic single-physics problem involving the travel time tomography survey, and a multiphysics parameter estimation problem involving the DCR and travel time tomography survey. Our numerical results show that our method accelerates the convergence of consensus ADMM, particularly in the early iterations. The quality of the parameter reconstructions obtained by the weighted async-ADMM scheme is comparable to that of the GN-PCG method; however, the weighted async-ADMM method requires substantially less communication among workers and has smaller latencies, resulting in reduced inversion runtimes and less communication. Moreover, since we can choose any optimization scheme to solve the subproblems in async-ADMM, the method sits at a higher level of abstraction and provides additional flexibility. Each subproblem can, therefore, be solved with a tailored solver, making the weighted async-ADMM especially attractive for large-scale multiphysics

PDE parameter estimation problems. For brevity, we do not show the case where the weights are computed in every iteration; however, in this case, we obtain reconstructions that are indiscernible from those shown in Figure 4.4. We intend to further explore our method for the case where the weights are correlated (nondiagonal), for large-scale problems where the GN-PCG method cannot be used, and on computational environments with small communication bandwidth such as cloud computing platforms.

Acknowledgments. We would like to thank the anonymous referees for their insightful comments which helped improve the manuscript.

REFERENCES

[1] F. Aminzadeh, B. Jean, and T. Kunz, 3-*D Salt and Overthrust Models*, Society of Exploration Geophysicists, Kansas City, 1997.

[2] C. Bekas, E. Kokiopoulou, and Y. Saad, *An estimator for the diagonal of a matrix*, Appl. Numer. Math., 57 (2007), pp. 1214–1229.

[3] P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Rev., 57 (2015), pp. 483–531, https://doi.org/10.1137/130932715.

[4] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *Julia: A fresh approach to numerical computing*, SIAM Rev., 59 (2017), pp. 65–98, https://doi.org/10.1137/141000671.

[5] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, *Large-scale PDE-constrained optimization: An introduction*, in Large-Scale PDE-Constrained Optimization, Springer, 2003, pp. 3–13.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn., 3 (2011), pp. 1–122.

[7] T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler, *A computational framework for infinite-dimensional Bayesian inverse problems part* I: *The linearized case, with application to global seismic inversion*, SIAM J. Sci. Comput., 35 (2013), pp. A2494–A2523, https://doi.org/10.1137/12089586X.

[8] D. Calvetti and E. Somersalo, *Large-scale statistical parameter estimation in complex systems with an application to metabolic models*, Multiscale Model. Simul., 5 (2006), pp. 1333–1366, https://doi.org/10.1137/050644860.

[9] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Softw. (TOMS), 38 (2011), 1.

[10] J. Eckstein and D. P. Bertsekas, *On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Programming, 55 (1992), pp. 293–318.

[11] P. Fika, M. Mitrouli, and P. Roupa, *Estimating the diagonal of matrix functions*, Math. Methods Appl. Sci., 41 (2018), pp. 1083–1088.

[12] H. P. Flath, L. C. Wilcox, V. Akçelik, J. Hill, B. van Bloemen Waanders, and O. Ghattas, *Fast algorithms for Bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial Hessian approximations*, SIAM J. Sci. Comput., 33 (2011), pp. 407–432, https://doi.org/10.1137/090780717.

[13] S. W. Fung and L. Ruthotto, *A multiscale method for model order reduction in PDE parameter estimation*, J. Comput. Appl. Math., 350 (2019), pp. 19–34.

[14] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, *Fast alternating direction optimization methods*, SIAM J. Imaging Sci., 7 (2014), pp. 1588–1623, https://doi.org/10.1137/120896219.

[15] T. Goldstein, G. Taylor, K. Barabin, and K. Sayre, *Unwrapping ADMM: Efficient distributed computing via transpose reduction*, in Proceedings of Machine Learning Research, Vol. 51: Artificial Intelligence and Statistics, Cadiz, Spain, 2016, pp. 1151–1158.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed., JHU Press, Baltimore, 2013.

[17] E. Haber, *Computational Methods in Geophysical Electromagnetics*, Math. Ind. 1, SIAM, Philadelphia, 2015, https://doi.org/10.1137/1.9781611973808.

[18] E. Haber, M. Chung, and F. Herrmann, *An effective method for parameter estimation with PDE constraints with multiple right-hand sides*, SIAM J. Optim., 22 (2012), pp. 739–757, https://doi.org/10.1137/11081126X.

[19] E. Haber and M. H. Gazit, *Model fusion and joint inversion*, Surveys Geophys., 34 (2013), pp. 675–695.

[20] W. W. Hager and H. Zhang, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM J. Optim., 16 (2005), pp. 170–192, https://doi.org/10.1137/030601880.

[21] W. W. Hager and H. Zhang, *A survey of nonlinear conjugate gradient methods*, Pacific J. Optim., 2 (2006), pp. 35–58.

[22] P. C. Hansen, *Regularization tools: A MATLAB package for analysis and solution of discrete ill-posed problems*, Numer. Algorithms, 6 (1994), pp. 1–35.

[23] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring Images: Matrices, Spectra, and Filtering*, Fund. Algorithms 3, SIAM, Philadelphia, 2006, https://doi.org/10.1137/1.9780898718874.

[24] J. Heredia-Juesas, A. Molaei, L. Tirado, W. Blackwell, and J. Á. Martínez-Lorenzo, *Norm-1 regularized consensus-based ADMM for imaging with a compressive antenna*, IEEE Antennas Wireless Propagation Lett., 16 (2017), pp. 2362–2365.

[25] M. Hong and Z.-Q. Luo, *On the linear convergence of the alternating direction method of multipliers*, Math. Programming, 162 (2017), pp. 165–199.

[26] M. Hong, Z.-Q. Luo, and M. Razaviyayn, *Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems*, SIAM J. Optim., 26 (2016), pp. 337–364, https://doi.org/10.1137/140990309.

[27] K. Huang and N. D. Sidiropoulos, *Consensus-ADMM for general quadratically constrained quadratic programming*, IEEE Trans. Signal Process., 64 (2016), pp. 5297–5310.

[28] J. H. Juesas, G. Allan, A. Molaei, L. Tirado, W. Blackwell, and J. A. M. Lorenzo, *Consensus-based imaging using ADMM for a compressive reflector antenna*, in Proceedings of the 2015 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting, IEEE, 2015, pp. 1304–1305.

[29] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello, *The sample average approximation method for stochastic discrete optimization*, SIAM J. Optim., 2 (2002), pp. 479–502, https://doi.org/10.1137/S1052623499363220.

[30] S. Li, S. Ahmed, G. Klimeck, and E. Darve, *Computing entries of the inverse of a sparse matrix using the find algorithm*, J. Comput. Phys., 227 (2008), pp. 9408–9427.

[31] L. Lin, J. Lu, L. Ying, R. Car, Weinan E, *Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems*, Commun. Math. Sci., 7 (2009), pp. 755–777.

[32] M. Ma, A. N. Nikolakopoulos, and G. B. Giannakis, *Fast decentralized learning via hybrid consensus ADMM*, in Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2018, pp. 3829–3833.

[33] J. Macdonald and L. Ruthotto, *Improved susceptibility artifact correction of echo-planar MRI using the alternating direction method of multipliers*, J. Math. Imaging Vision, 60 (2018), pp. 268–282.

[34] H. Miao, X. Liu, B. Huang, and L. Getoor, *A hypergraph-partitioned vertex programming approach for large-scale consensus optimization*, in Proceedings of the 2013 IEEE International Conference on Big Data, IEEE, 2013, pp. 563–568.

[35] M. Ohlberger and K. Smetana, *A dimensional reduction approach based on the application of reduced basis methods in the framework of hierarchical model reduction*, SIAM J. Sci. Comput., 36 (2014), pp. A714–A736, https://doi.org/10.1137/130939122.

[36] B. T. Polyak and A. B. Juditsky, *Acceleration of stochastic approximation by averaging*, SIAM J. Control Optim., 30 (1992), pp. 838–855, https://doi.org/10.1137/0330046.

[37] H. Robbins and S. Monro, *A Stochastic Approximation Method*, Ann. Math. Statist., 22 (1951), pp. 400–407.

[38] L. Ruthotto, *KrylovMethods.jl*, https://github.com/lruthotto/KrylovMethods.jl, 2019.

[39] L. Ruthotto, E. Treister, and E. Haber, *jInv–a flexible Julia package for PDE parameter estimation*, SIAM J. Sci. Comput., 39 (2017), pp. S702–S722, https://doi.org/10.1137/16M1081063.

[40] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, 2003, https://doi.org/10.1137/1.9780898718003.

[41] A. K. Saibaba, J. Lee, and P. K. Kitanidis, *Randomized algorithms for generalized Hermitian eigenvalue problems with application to computing Karhunen–Loève expansion*, Numer. Linear Algebra Appl., 23 (2016), pp. 314–339.

[42] J. H. Schön, ed., *Physical Properties of Rocks: Fundamentals and Principles of Petrophysics*, Developments in Petroleum Science 65, Elsevier, Amsterdam, 2015.

[43] J. Sherman and W. J. Morrison, *Adjustment of an inverse matrix corresponding to a change*

in one element of a given matrix, Ann. Math. Statist., 21 (1950), pp. 124–127.

[44] A. M. Stuart, *Inverse problems: A Bayesian perspective*, Acta Numer., 19 (2010), pp. 451–559.

[45] J. M. Tang and Y. Saad, *Domain-decomposition-type methods for computing the diagonal of a matrix inverse*, SIAM J. Sci. Comput., 33 (2011), pp. 2823–2847, https://doi.org/10.1137/100799939.

[46] J. M. Tang and Y. Saad, *A probing method for computing the diagonal of a matrix inverse*, Numer. Linear Algebra Appl., 19 (2012), pp. 485–501.

[47] G. Taylor, Z. Xu, and T. Goldstein, *Scalable classifiers with ADMM and transpose reduction*, in Proceedings of the Workshops at the Thirty-First AAAI Conference on Artificial Intelligence, AAAI, 2017, https://www.aaai.org/ocs/index.php/WS/AAAIW17/paper/view/15174.

[48] E. Treister and E. Haber, *A fast marching algorithm for the factored eikonal equation*, J. Comput. Phys., 324 (2016), pp. 210–225.

[49] J. Tsitsiklis, D. Bertsekas, and M. Athans, *Distributed asynchronous deterministic and stochastic gradient optimization algorithms*, IEEE Trans. Automat. Control, 31 (1986), pp. 803–812.

[50] S. J. Wright and J. Nocedal, *Numerical Optimization*, Springer Ser. Oper. Res., Springer, 1999.

[51] Z. Xu, G. Taylor, H. Li, M. A. Figueiredo, X. Yuan, and T. Goldstein, *Adaptive consensus ADMM for distributed optimization*, in Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, Vol. 70, 2017, pp. 3841–3850.

[52] R. Zhang and J. Kwok, *Asynchronous distributed ADMM for consensus optimization*, in Proceedings of the 31st International Conference on on Machine Learning, Beijing, China, Vol. 32, 2014, pp. 1701–1709.