A Flexible ADMM for Big Data Applications

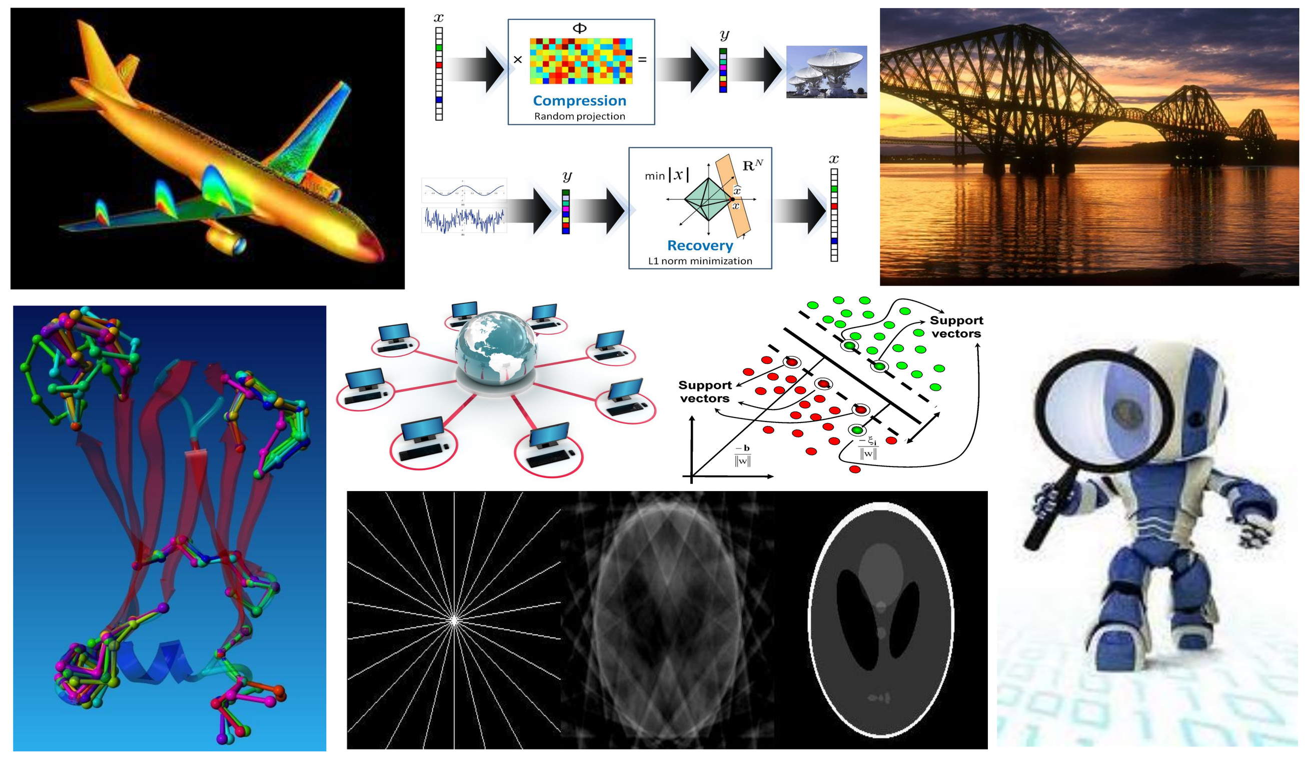
Daniel Robinson and Rachael Tappenden, Johns Hopkins University

Introduction

We study the optimization problem:

$$\underset{x_1,\dots,x_n}{\text{minimize}} \quad f(x) := \sum_{i=1}^{n} f_i(x_i) \quad \text{subject to} \quad \sum_{i=1}^{n} A_i x_i = b. \tag{1}$$

- Functions $f_i : \mathbf{R}^{N_i} \to \mathbf{R} \cup \{\infty\}$ are strongly convex with constant $\mu_i > 0$
- Vector $b \in \mathbf{R}^m$ and matrices $A_i \in \mathbf{R}^{m \times N_i}$ are problem data.
- We can write $x = [x_1, \dots, x_n] \in \mathbf{R}^N$, $A = [A_1, \dots, A_n]$, with $x_i \in \mathbf{R}^{N_i}$ and $N = \sum_{i=1}^{n} N_i$
- We assume a solution to (1) exists



What's new?!

1. Present a new Flexible Alternating Direction Method of Multipliers (F-ADMM) to solve (1).
   - For strongly convex f_i; for general $n \geq 2$; uses a *Gauss-Seidel* updating scheme.
2. Incorporate (very general) regularization matrices $\{P_i\}_{i=1}^n$
   - Stabilize the iterates; make subproblems easier to solve
   - **Assume:** $\{P_i\}_{i=1}^n$ are symmetric and sufficiently positive definite.
3. Prove that F-ADMM is *globally convergent*.
4. Introduce a Hybrid ADMM variant (H-ADMM) that is *partially parallelizable*.
5. Special case of H-ADMM can be applied to convex functions

A Flexible ADMM (F-ADMM)

Algorithm 1 F-ADMM for solving problem (1).

1: **Initialize:** $x^{(0)}$, $y^{(0)}$, parameters $\rho > 0$, $\gamma \in (0, 2)$, and matrices $\{P_i\}_{i=1}^n$
2: **for** $k = 0, 1, 2, \dots$ **do**
3: **for** $i = 1, \dots, n$ **do** Update the primal variables in a Gauss-Seidel fashion:

$$x_i^{(k+1)} \leftarrow \arg\min_{x_i} \left\{ f_i(x_i) + \frac{\rho}{2} \| A_i x_i + \sum_{j=1}^{i-1} A_j x_j^{(k+1)} + \sum_{l=i+1}^{n} A_l x_l^{(k)} - b - \frac{y^{(k)}}{\rho} \|_2^2 + \frac{1}{2} \| x_i - x_i^{(k)} \|_{P_i}^2 \right\}$$

4: **end for**
5: Update dual variables: $y^{(k+1)} \leftarrow y^{(k)} - \gamma\rho(Ax^{(k+1)} - b)$
6: **end for**

Theorem 1. *Under our assumptions the sequence* $\{(x^{(k)}, y^{(k)})\}_{k \geq 0}$ *generated by Algorithm 1 converges to some vector* (x^*, y^*) *that is a solution to problem* (1).

A Hybrid ADMM (H-ADMM)

F-ADMM is inherently serial (it has a Gauss-Seidel-type updating scheme)
Jacobi-type (parallel) methods are imperative for big data problems
Goal: find a balance between algorithm speed via parallelization (Jacobi) and allowing up-to-date information to be fed back into the algorithm (Gauss-Seidel).

- Apply F-ADMM to "grouped data" and choose the regularization matrix carefully
- New Hybrid Gauss-Seidel/Jacobi ADMM method (H-ADMM) that is *partially parallelizable*

Group the data: Form ℓ groups of p blocks where $n = \ell p$

$$x = [\underbrace{x_1, \dots, x_p}_{\mathbf{x}_1} \mid \underbrace{x_{p+1}, \dots, x_{2p}}_{\mathbf{x}_2} \mid \dots \mid \underbrace{x_{(\ell-1)p+1} \cdots x_n}_{\mathbf{x}_\ell}]$$

and

$$A = [\underbrace{A_1, \dots, A_p}_{\mathcal{A}_1} \mid \underbrace{A_{p+1}, \dots, A_{2p}}_{\mathcal{A}_2} \mid \dots \mid \underbrace{A_{(\ell-1)p+1} \cdots A_n}_{\mathcal{A}_\ell}]$$

Problem (1) becomes:

$$\underset{x \in \mathbf{R}^N}{\text{minimize}} \quad f(x) \equiv \sum_{j=1}^{\ell} \mathbf{f}_j(\mathbf{x}_j) \quad \text{subject to} \quad \sum_{j=1}^{\ell} \mathcal{A}_j \mathbf{x}_j = b. \tag{2}$$

Choice of 'group' regularization matrices $\{\mathcal{P}_i\}_{i=1}^n$ is crucial: Choosing

$$\mathcal{P}_i := \text{blkdiag}(P_{\mathcal{S}_{i,1}}, \dots, P_{\mathcal{S}_{i,p}}) - \rho \mathcal{A}_i^T \mathcal{A}_i \tag{3}$$

(index set $\mathcal{S}_i = \{(i-1)p, \dots, ip\}$ for $1 \leq i \leq \ell$) makes group iterations separable/parallelizable!

Algorithm 2 H-ADMM for solving problem (1).

1: **Initialize:** $x^{(0)}$, $y^{(0)}$, $\rho > 0$ and $\gamma \in (0, 2)$, index sets $\{\mathcal{S}_i\}_{i=1}^{\ell}$, matrices $\{P_i\}_{i=1}^n$.
2: **for** $k = 0, 1, 2, \dots$ **do**
3: **for** $i = 1, \dots, \ell$ (in a serial Gauss-Seidel fashion solve) **do**
4: Set $\mathbf{b}_i \leftarrow b - \sum_{q=1}^{i-1} \mathcal{A}_q \mathbf{x}_q^{(k+1)} - \sum_{s=i}^{\ell} \mathcal{A}_s \mathbf{x}_s^{(k)} + \frac{y^{(k)}}{\rho}$
5: **for** $j \in \mathcal{S}_i$ (in parallel Jacobi fashion solve) **do**

$$x_j^{(k+1)} \leftarrow \arg\min_{x_j} \left\{ f_j(x_j) + \frac{\rho}{2} \| A_j(x_j - x_j^{(k)}) - \mathbf{b}_i \|_2^2 + \frac{1}{2} \| x_j - x_j^{(k)} \|_{P_j}^2 \right\}$$

6: **end for**
7: **end for**
8: Update the dual variables: $y^{(k+1)} \leftarrow y^{(k)} - \gamma\rho(Ax^{(k+1)} - b)$.
9: **end for**

Key features of H-ADMM

- H-ADMM is a special case of F-ADMM (Convergence is automatic from F-ADMM theory)
- Groups updated in Gauss-Seidel fashion; Blocks within group updated in Jacobi fashion
  - Decision variables x_j for $j \in \mathcal{S}_i$ are solved for *in parallel*!
- Regularization matrices $\mathcal{P}_i$:
  - Never explicitly form $\mathcal{P}_i$ for $i = 1, \dots, \ell$. Only need $P_1, \dots, P_n$
  - Regularization matrix $\mathcal{P}_i$ makes subproblem for group $\mathbf{x}_i^{(k+1)}$ separable into p blocks
  - **Assume:** Matrices $\mathcal{P}_i$ are symmetric and sufficiently positive definite
- Computational considerations
  - "Optimize" H-ADMM to number of processors: For machine with p processors, set group size to p too!
  - Still using "new" information, just not as much as in 'individual block' setting
  - Special Case: Hybrid ADMM with $\ell = 2$, $\Rightarrow$ convergence holds for convex f_i

Numerical Experiments

Determine the solution to an underdetermined system of equations with the smallest 2-norm:

$$\underset{x \in \mathbf{R}^N}{\text{minimize}} \quad \frac{1}{2}\|x\|_2^2 \quad \text{subject to} \quad Ax = b. \tag{4}$$

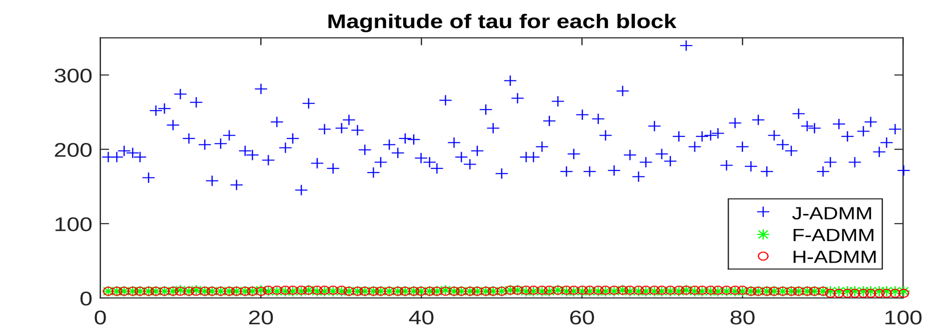
- $n = 100$ blocks; $p = 10$ processors; $\ell = 10$ groups; block size $N_i = 100 \; \forall i$;
- Convexity parameter $\mu_i = 1 \; \forall i$; A is $3 \cdot 10^3 \times 10^4$ and sparse; Noiseless data
- Algorithm parameters: $\rho = 0.1$, $\gamma = 1$; stopping condition: $\frac{1}{2}\|Ax - b\|_2^2 \leq 10^{-10}$.

Choosing $P_j = \tau_j I - \rho A_j^T A_j \; \forall j \in \mathcal{S}_i$, for some $\tau_j > 0$, gives

$$x_j^{(k+1)} = \frac{\tau_j}{\tau_j + 1} x_j^{(k)} + \frac{\rho}{\tau_j + 1} A_j^T \mathbf{b}_i \quad \forall j \in \mathcal{S}_i.$$

We compare F-ADMM and H-ADMM with Jacobi ADMM (J-ADMM) [1]. (J-ADMM is similar to F-ADMM, but the blocks are updated using a Jacobi-type scheme.)

Results using theoretical τ_j values



Method	Epochs
J-ADMM	4358.2
H-ADMM	214.1
F-ADMM	211.3

Left plot: A plot of the magnitude of τ_j for each block $1 \leq j \leq n$ for problem (4). The τ_j values are much smaller for H-ADMM and F-ADMM, than for J-ADMM. Larger τ_j corresponds to 'more positive definite' regularization matrices P_j. Right table: Number of epochs required by J-ADMM, F-ADMM, and H-ADMM for problem (4) using theoretical values of τ_j for $j = 1, \dots, n$ (averaged over 100 runs). H-ADMM and F-ADMM require significantly fewer epochs than J-ADMM using theoretical values of τ_j.

Results using parameter tuning

Parameter tuning can help practical performance! Assign the same value τ_j for all blocks $j = 1, \dots, n$ and all algorithms. (i.e., $\tau_1 = \tau_2 = \dots = \tau_n$.)

τ_j	J-ADMM	H-ADMM	F-ADMM
$\frac{\rho^2}{2}\|A\|^4$	530.0	526.3	526.2
$0.6 \cdot \frac{\rho^2}{2}\|A\|^4$	324.0	320.1	319.9
$0.4 \cdot \frac{\rho^2}{2}\|A\|^4$	217.7	214.5	214.1
$0.22 \cdot \frac{\rho^2}{2}\|A\|^4$	123.1	119.3	119.0
$0.2 \cdot \frac{\rho^2}{2}\|A\|^4$	—	95.8	95.5
$0.1 \cdot \frac{\rho^2}{2}\|A\|^4$	—	75.3	73.0

The table presents the number of epochs required by J-ADMM, F-ADMM, and H-ADMM on problem (4) as τ_j varies. For each τ_j we run each algorithm (J-ADMM, F-ADMM and H-ADMM) on 100 random instances of the problem formulation described above. Here, τ_j takes the same value for all blocks $j = 1, \dots, n$. All algorithms require fewer epochs as τ_j decreases, and F-ADMM requires the smallest number of epochs.

- As τ_j decreases, number of epochs decreases
- For fixed τ_j F-ADMM and H-ADMM outperform J-ADMM
- F-ADMM converge for very small τ_j values

References

1. Wei Deng, Ming-Jun Lai, Zhimin Peng, and Wotao Yin, "Parallel Multi-Block ADMM with o(1/k) Convergence", Technical Report, UCLA (2014)
2. Daniel Robinson and Rachael Tappenden, "A Flexible ADMM Algorithm for Big Data Applications", Technical Report, JHU, arXiv:1502.04391 (2015)

{daniel.p.robinson,rtappen1}@jhu.edu