Accuracy v.s. Implementability in Algorithmic Design —- An Example of Operator Splitting Methods for Convex Optimization

Xiaoming Yuan

Hong Kong Baptist University

September 02, 2014

Outline

1. Backgrounds
2. Accuracy v.s. Implementability – An Easier Case
3. Accuracy v.s. Implementability – A More Complicated Case
4. Conclusions

Outline

What Do I Want to Say?

- Accuracy:
  - The fidelity to the original model.

What Do I Want to Say?

- Accuracy:
  - The fidelity to the original model.
  - Able to solve a subproblem EXACTLY.

What Do I Want to Say?

- Accuracy:
  - The fidelity to the original model.
  - Able to solve a subproblem EXACTLY.
  - Maintain the convergence (or faster convergence) of an algorithm.

What Do I Want to Say?

- Accuracy:
    - The fidelity to the original model.
    - Able to solve a subproblem EXACTLY.
    - Maintain the convergence (or faster convergence) of an algorithm.
- Implementability:

What Do I Want to Say?

- Accuracy:
  - The fidelity to the original model.
  - Able to solve a subproblem EXACTLY.
  - Maintain the convergence (or faster convergence) of an algorithm.
- Implementability:
  - Easy to solve a subproblem

What Do I Want to Say?

- Accuracy:
  - The fidelity to the original model.
  - Able to solve a subproblem EXACTLY.
  - Maintain the convergence (or faster convergence) of an algorithm.
- Implementability:
  - Easy to solve a subproblem
  - Ready for coding

What Do I Want to Say?

- Accuracy:
  - The fidelity to the original model.
  - Able to solve a subproblem EXACTLY.
  - Maintain the convergence (or faster convergence) of an algorithm.
- Implementability:
  - Easy to solve a subproblem
  - Ready for coding
- They are both important (I hope you also agree).

What Do I Want to Say?

- Accuracy:
  - The fidelity to the original model.
  - Able to solve a subproblem EXACTLY.
  - Maintain the convergence (or faster convergence) of an algorithm.
- Implementability:
  - Easy to solve a subproblem
  - Ready for coding
- They are both important (I hope you also agree).
- Yet, they are usually conflicted (to be proved later).

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b,\, x \in \mathcal{X}\},$$

with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b, \, x \in \mathcal{X}\},$$

  with $A \in \Re^{m \times n}$, $b \in \Re^{m}$, $\mathcal{X} \subseteq \Re^{n}$ a closed convex set, $\theta : \Re^{n} \to \Re$ a convex but not necessarily smooth function.

- Solving the original model — thus with 100% accuracy.

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b,\ x \in \mathcal{X}\},$$

  with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.
- Solving the original model — thus with 100% accuracy. But how?

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b, x \in \mathcal{X}\},$$

  with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.
- Solving the original model — thus with 100% accuracy. But how? — in general, not possible.

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b, \, x \in \mathcal{X}\},$$

with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.

- Solving the original model — thus with 100% accuracy. But how? — in general, not possible. — not implementable.

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b,\, x \in \mathcal{X}\},$$

  with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.
- Solving the original model — thus with 100% accuracy. But how? — in general, not possible. — not implementable.
- The penalty method:

$$x^{k+1} = \arg\min\{\theta(x) + \frac{\beta}{2}\|Ax - b\|^2 \mid x \in \mathcal{X}\}$$

  which solves an easier problem without linear constraints

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b,\ x \in \mathcal{X}\},$$

with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.

- Solving the original model — thus with 100% accuracy. But how? — in general, not possible. — not implementable.
- The penalty method:

$$x^{k+1} = \arg\min\{\theta(x) + \frac{\beta}{2}\|Ax - b\|^2 \mid x \in \mathcal{X}\}$$

which solves an easier problem without linear constraints — with much more implementability.

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b,\, x \in \mathcal{X}\},$$

  with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.
- Solving the original model — thus with 100% accuracy. But how? — in general, not possible. — not implementable.
- The penalty method:

$$x^{k+1} = \arg\min\{\theta(x) + \frac{\beta}{2}\|Ax - b\|^2 \,|\, x \in \mathcal{X}\}$$

  which solves an easier problem without linear constraints — with much more implementability.
- Of course, with much less accuracy

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b, x \in \mathcal{X}\},$$

with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.

- Solving the original model — thus with 100% accuracy. But how? — in general, not possible. — not implementable.
- The penalty method:

$$x^{k+1} = \arg\min\{\theta(x) + \frac{\beta}{2}\|Ax - b\|^2 \,|\, x \in \mathcal{X}\}$$

which solves an easier problem without linear constraints — with much more implementability.

- Of course, with much less accuracy —- indeed, not necessarily convergent if $\beta \nrightarrow +\infty$.

A Canonical Convex Optimization Model

- A canonical convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b, x \in \mathcal{X}\},$$

  with $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \subseteq \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function.
- Solving the original model — thus with 100% accuracy. But how? — in general, not possible. — not implementable.
- The penalty method:

$$x^{k+1} = \arg\min\{\theta(x) + \frac{\beta}{2}\|Ax - b\|^2 \mid x \in \mathcal{X}\}$$

  which solves an easier problem without linear constraints — with much more implementability.
- Of course, with much less accuracy —- indeed, not necessarily convergent if $\beta \nrightarrow +\infty$.
- With sufficient implementability while too little accuracy.

The augmented Lagrangian method

- How can we keep both the implementability (as the penalty method) and accuracy (with convergence)?

The augmented Lagrangian method

- How can we keep both the implementability (as the penalty method) and accuracy (with convergence)?
- Answer: The augmented Lagrangian method (H. Hestenes and M. Powell in 1969, individually)

The augmented Lagrangian method

- How can we keep both the implementability (as the penalty method) and accuracy (with convergence)?
- Answer: The augmented Lagrangian method (H. Hestenes and M. Powell in 1969, individually)

$$\left\{ \begin{array}{lcl} x^{k+1} & = & \arg\min\{\theta(x) - (\lambda^k)^T(Ax - b) + \frac{\beta}{2}\|Ax - b\|^2 \mid x \in \mathcal{X}\} \\ \lambda^{k+1} & = & \lambda^k - \beta(Ax^{k+1} - b) \end{array} \right.$$

where $\lambda \in \Re^m$ is the Lagrange multiplier and $\beta > 0$ is a penalty parameter.

The augmented Lagrangian method

- How can we keep both the implementability (as the penalty method) and accuracy (with convergence)?
- Answer: The augmented Lagrangian method (H. Hestenes and M. Powell in 1969, individually)

$$\left\{ \begin{array}{rcl} x^{k+1} & = & \arg\min\{\theta(x) - (\lambda^k)^T(Ax-b) + \frac{\beta}{2}\|Ax-b\|^2 \,\big|\, x \in \mathcal{X}\} \\ \lambda^{k+1} & = & \lambda^k - \beta(Ax^{k+1} - b) \end{array} \right.$$

where $\lambda \in \Re^m$ is the Lagrange multiplier and $\beta > 0$ is a penalty parameter.

- The subproblem is as difficult as that of the penalty method (the same level of implementability)

The augmented Lagrangian method

- How can we keep both the implementability (as the penalty method) and accuracy (with convergence)?
- Answer: The augmented Lagrangian method (H. Hestenes and M. Powell in 1969, individually)

$$\begin{cases} x^{k+1} &= \arg\min\{\theta(x) - (\lambda^k)^T(Ax-b) + \frac{\beta}{2}\|Ax-b\|^2 \mid x \in \mathcal{X}\} \\ \lambda^{k+1} &= \lambda^k - \beta(Ax^{k+1} - b) \end{cases}$$

  where $\lambda \in \Re^m$ is the Lagrange multiplier and $\beta > 0$ is a penalty parameter.
- The subproblem is as difficult as that of the penalty method (the same level of implementability)
- It is convergent with any fixed $\beta > 0$ (higher accuracy)

Some Comments on ALM

The ALM:

$$\begin{cases} x^{k+1} & = & \arg\min\{\theta(x) - (\lambda^k)^T(Ax-b) + \frac{\beta}{2}\|Ax-b\|^2 \mid x \in \mathcal{X}\} \\ \lambda^{k+1} & = & \lambda^k - \beta(Ax^{k+1} - b) \end{cases}$$

- ALM has an augmented term and it updates the dual variable iteratively

Some Comments on ALM

The ALM:

$$\begin{cases} x^{k+1} = \arg\min\{\theta(x) - (\lambda^k)^T(Ax-b) + \frac{\beta}{2}\|Ax-b\|^2 \mid x \in \mathcal{X}\} \\ \lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} - b) \end{cases}$$

- ALM has an augmented term and it updates the dual variable iteratively
- In 1976, T. Rockafellar showed that ALM is an application of the proximal point algorithm (B. Martinet, 1970, or even earlier, J. Moreau, 1965) to the dual problem of the model above.

Some Comments on ALM

The ALM:

$$\begin{cases} x^{k+1} & = & \arg\min\{\theta(x) - (\lambda^k)^T(Ax - b) + \frac{\beta}{2}\|Ax - b\|^2 \mid x \in \mathcal{X}\} \\ \lambda^{k+1} & = & \lambda^k - \beta(Ax^{k+1} - b) \end{cases}$$

- ALM has an augmented term and it updates the dual variable iteratively
- In 1976, T. Rockafellar showed that ALM is an application of the proximal point algorithm (B. Martinet, 1970, or even earlier, J. Moreau, 1965) to the dual problem of the model above.
- It can be regarded as a dual ascent method over the dual variable λ.

Some Comments on ALM

The ALM:

$$\begin{cases} x^{k+1} & = & \arg\min\{\theta(x) - (\lambda^k)^T(Ax - b) + \frac{\beta}{2}\|Ax - b\|^2 \mid x \in \mathcal{X}\} \\ \lambda^{k+1} & = & \lambda^k - \beta(Ax^{k+1} - b) \end{cases}$$

- ALM has an augmented term and it updates the dual variable iteratively
- In 1976, T. Rockafellar showed that ALM is an application of the proximal point algorithm (B. Martinet, 1970, or even earlier, J. Moreau, 1965) to the dual problem of the model above.
- It can be regarded as a dual ascent method over the dual variable λ.
- A significant difference from the penalty method — the penalty parameter of ALM can theoretically be fixed as any positive scalar.

Outline

A Separable Model

- For many applications, the last model can be specified as a separable form

$$\min\{\theta_1(x_1) + \theta_2(x_2) \mid A_1x_1 + A_2x_2 = b,\ x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2\},$$

where $A_1 \in \Re^{m\times n_1}$, $A_2 \in \Re^{m\times n_2}$, $b \in \Re^m$, $\mathcal{X}_i \subseteq \Re^{n_i}$ $(i = 1, 2)$ and $\theta_i : \Re^{n_i} \to \Re$ $(i = 1, 2)$.

A Separable Model

- For many applications, the last model can be specified as a separable form

$$\min\{\theta_1(x_1) + \theta_2(x_2) \mid A_1x_1 + A_2x_2 = b,\ x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2\},$$

  where $A_1 \in \Re^{m\times n_1}$, $A_2 \in \Re^{m\times n_2}$, $b \in \Re^m$, $\mathcal{X}_i \subseteq \Re^{n_i}$ $(i = 1, 2)$ and $\theta_i : \Re^{n_i} \to \Re$ $(i = 1, 2)$.
- This model corresponds to the last model with $\theta(x) = \theta_1(x_1) + \theta_2(x_2)$, $x = (x_1, x_2)$, $A = (A_1, A_2)$, $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$ and $n = n_1 + n_2$.

A Separable Model

- For many applications, the last model can be specified as a separable form

$$\min\{\theta_1(x_1)+\theta_2(x_2) \mid A_1x_1+A_2x_2=b,\ x_1\in\mathcal{X}_1, x_2\in\mathcal{X}_2\},$$

  where $A_1\in\Re^{m\times n_1}$, $A_2\in\Re^{m\times n_2}$, $b\in\Re^m$, $\mathcal{X}_i\subseteq\Re^{n_i}$ $(i=1,2)$ and $\theta_i:\Re^{n_i}\to\Re$ $(i=1,2)$.
- This model corresponds to the last model with $\theta(x)=\theta_1(x_1)+\theta_2(x_2)$, $x=(x_1,x_2)$, $A=(A_1,A_2)$, $\mathcal{X}=\mathcal{X}_1\times\mathcal{X}_2$ and $n=n_1+n_2$.
- A typical application of the widely-used l_1-l_2 model

$$\min\{\mu\|x\|_1+\frac{1}{2}\|Ax-b\|^2\}$$

  where the least-square term $\frac{1}{2}\|Ax-b\|^2$ represents a data-fidelity term and the l_1-norm term $\|x\|_1$ is a regularization term for inducing spare solutions, and $\mu>0$ is a trade-off parameter.

Using ALM Directly with 100% Accuracy

Applying ALM directly:

$$\begin{cases} (x_1^{k+1}, x_2^{k+1}) = \arg\min\{\theta_1(x_1) + \theta_2(x_2) - (\lambda^k)^T(A_1x_1 + A_2x_2 - b) + \frac{\beta}{2}\|A_1x_1 + A_2x_2 - b\|^2 \mid (x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2\} \\ \lambda^{k+1} = \lambda^k - \beta(A_1x^{k+1} + A_2x_2^{k+1} - b); \end{cases}$$

Using ALM Directly with 100% Accuracy

Applying ALM directly:

$$\begin{cases} (x_1^{k+1}, x_2^{k+1}) = \arg\min\{\theta_1(x_1) + \theta_2(x_2) - (\lambda^k)^T(A_1x_1 + A_2x_2 - b) + \frac{\beta}{2}\|A_1x_1 + A_2x_2 - b\|^2 \mid (x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2\} \\ \lambda^{k+1} = \lambda^k - \beta(A_1x^{k+1} + A_2x_2^{k+1} - b); \end{cases}$$

How about its implementability?

Using ALM Directly with 100% Accuracy

Applying ALM directly:

$$\begin{cases} (x_1^{k+1}, x_2^{k+1}) = \arg\min\{\theta_1(x_1) + \theta_2(x_2) - (\lambda^k)^T(A_1x_1 + A_2x_2 - b) + \frac{\beta}{2}\|A_1x_1 + A_2x_2 - b\|^2 \mid (x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2\} \\ \lambda^{k+1} = \lambda^k - \beta(A_1x^{k+1} + A_2x_2^{k+1} - b); \end{cases}$$

How about its implementability?

Is it easy to solve the ALM subproblem exactly?

Splitting the ALM with Less Accuracy?

Splitting the ALM with Less Accuracy?

- Parallel (Jacobian) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^k + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

Splitting the ALM with Less Accuracy?

- Parallel (Jacobian) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^k + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Sequential (Gauss-Seidel) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

Splitting the ALM with Less Accuracy?

- Parallel (Jacobian) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^k + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Sequential (Gauss-Seidel) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Both lose accuracy but gain implementability — less accurate but more implementable cases compared to the original ALM.

Splitting the ALM with Less Accuracy?

- Parallel (Jacobian) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^k + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Sequential (Gauss-Seidel) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Both lose accuracy but gain implementability — less accurate but more implementable cases compared to the original ALM.
- They are equally implementable, and Sequential Splitting is more accurate.

Splitting the ALM with Less Accuracy?

- Parallel (Jacobian) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^k + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Sequential (Gauss-Seidel) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Both lose accuracy but gain implementability — less accurate but more implementable cases compared to the original ALM.
- They are equally implementable, and Sequential Splitting is more accurate.
- Parallel Splitting is not convergent (He/Hou/Y, 2013).

Splitting the ALM with Less Accuracy?

- Parallel (Jacobian) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^k + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Sequential (Gauss-Seidel) Splitting:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- Both lose accuracy but gain implementability — less accurate but more implementable cases compared to the original ALM.
- They are equally implementable, and Sequential Splitting is more accurate.
- Parallel Splitting is not convergent (He/Hou/Y, 2013).
- Sequential Splitting is convergent — the Alternating Direction Method of Multipliers (ADMM) originally proposed by R. Glowinski and Marrocco in 1975.

Comments on ADMM

The ADMM scheme:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- ADMM represents an inexact version of ALM, because the (x_1, x_2)-subproblem in ALM is decomposed into two smaller ones.

Comments on ADMM

The ADMM scheme:

$$\left\{\begin{array}{l} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{array}\right.$$

- ADMM represents an inexact version of ALM, because the (x_1, x_2)-subproblem in ALM is decomposed into two smaller ones.
- It is possible to take advantage of the properties of θ_1 and θ_2 individually — the decomposed subproblems are potentially much easier than the aggregated subproblem in (the original subproblem of) ALM.

Comments on ADMM

The ADMM scheme:

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- ADMM represents an inexact version of ALM, because the (x_1, x_2)-subproblem in ALM is decomposed into two smaller ones.
- It is possible to take advantage of the properties of θ_1 and θ_2 individually — the decomposed subproblems are potentially much easier than the aggregated subproblem in (the original subproblem of) ALM.
- For the mentioned l_1-l_2 model, all subproblems are even easy enough to have closed-form solutions (to be delineated).

Cont'd

- A “renaissance" of ADMM in many application domains such as image processing, statistical learning, computer vision, and so on.

Cont'd

- A “renaissance" of ADMM in many application domains such as image processing, statistical learning, computer vision, and so on.
- In 2011, we proved ADMM's convergence rate.

Cont'd

- A “renaissance" of ADMM in many application domains such as image processing, statistical learning, computer vision, and so on.
- In 2011, we proved ADMM's convergence rate.
- Review papers: Boyd *et al.* 2010, Glowinski 2012, Eckstein and Yao 2012.

Accuracy of ADMM

Certainly, acquiring implementability does not mean no care about the accuracy.

[1] Ng/Wang/Y., Inexact alternating direction methods for image recovery, SIAM Journal on Scientific Computing, 33(4), 1643-1668, 2011.

Accuracy of ADMM

Certainly, acquiring implementability does not mean no care about the accuracy.

- The accuracy of ADMM's subproblems should be considered seriously.

$$\begin{cases} x_1^{k+1} \approx \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} \approx \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

[1] Ng/Wang/Y., Inexact alternating direction methods for image recovery, SIAM Journal on Scientific Computing, 33(4), 1643-1668, 2011.

Accuracy of ADMM

Certainly, acquiring implementability does not mean no care about the accuracy.

- The accuracy of ADMM's subproblems should be considered seriously.

$$\begin{cases} x_1^{k+1} \approx \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} \approx \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- How to define "$\approx$" rigorously above?

[1] Ng/Wang/Y., Inexact alternating direction methods for image recovery, SIAM Journal on Scientific Computing, 33(4), 1643-1668, 2011.

Accuracy of ADMM

Certainly, acquiring implementability does not mean no care about the accuracy.

- The accuracy of ADMM's subproblems should be considered seriously.

$$\begin{cases} x_1^{k+1} \approx \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} \approx \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases}$$

- How to define "≈" rigorously above?
- For a general case, we need to analyze rigorously the inexactness criterion for solving these subproblems [1].

[1] Ng/Wang/Y., Inexact alternating direction methods for image recovery, SIAM Journal on Scientific Computing, 33(4), 1643-1668, 2011.

Two ADMM Applications

(1) Compressive Sensing (Donoho, Candes, Tao, $\cdots$)

Two ADMM Applications

(1) Compressive Sensing (Donoho, Candes, Tao, $\cdots$)

- Allowing us to go beyond the Shannon limit to exploit the sparsity of a signal.

Two ADMM Applications

(1) Compressive Sensing (Donoho, Candes, Tao, $\cdots$)

- Allowing us to go beyond the Shannon limit to exploit the sparsity of a signal.
- Acquiring important information of a signal efficiently (e.g., storage-saving, speed-improving).

Two ADMM Applications

(1) Compressive Sensing (Donoho, Candes, Tao, $\cdots$)

- Allowing us to go beyond the Shannon limit to exploit the sparsity of a signal.
- Acquiring important information of a signal efficiently (e.g., storage-saving, speed-improving).

Two ADMM Applications

(1) Compressive Sensing (Donoho, Candes, Tao, $\cdots$)

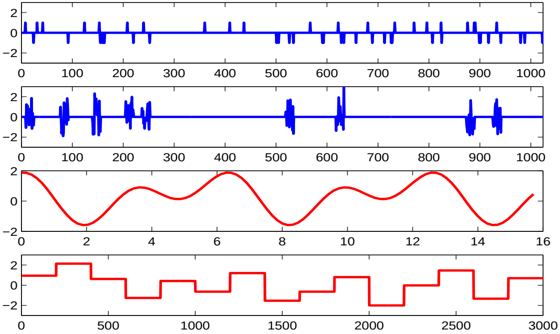
- Allowing us to go beyond the Shannon limit to exploit the sparsity of a signal.
- Acquiring important information of a signal efficiently (e.g., storage-saving, speed-improving).

Ideal model: $Ax = b$

x — original signal, A — sensing matrix (a fat matrix), b — observation (with noise)

The Sparsity of a Signal

Some signals are large-scale but sparse (maybe under some transform domain)

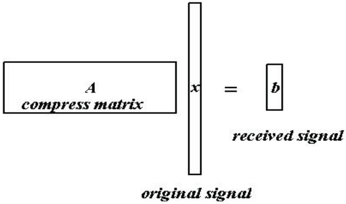


Mathematical Model

Find a sparse solution of a system of linear equations

$$\min \ \left\{ \|x\|_0 \ \middle| \ Ax = b, \ x \in \mathcal{R}^n \right\},$$

where $\|x\|_0 =$ number of nonzeros of x and $A \in \mathcal{R}^{m \times n}$ with $m \ll n$.

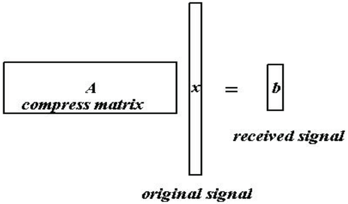


Mathematical Model

Find a sparse solution of a system of linear equations

$$\min \ \{\|x\|_0 \mid Ax = b, \ x \in \mathcal{R}^n\},$$

where $\|x\|_0 =$ number of nonzeros of x and $A \in \mathcal{R}^{m\times n}$ with $m \ll n$.



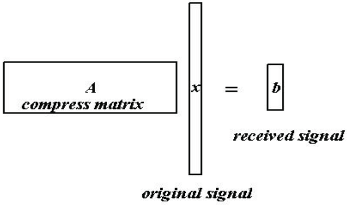
- The solution is in general not unique.

Mathematical Model

Find a sparse solution of a system of linear equations

$$\min \ \left\{ \|x\|_0 \ \middle| \ Ax = b, \ x \in \mathcal{R}^n \right\},$$

where $\|x\|_0 =$ number of nonzeros of x and $A \in \mathcal{R}^{m \times n}$ with $m \ll n$.



- The solution is in general not unique.
- It is NP-hard!

Basic Models for Compressive Sensing

- Basis-pursuit (BP):

$$\min\{\|x\|_1 \;\mid\; Ax = b\}$$

Basic Models for Compressive Sensing

- Basis-pursuit (BP):

$$\min \{\|x\|_1 \mid Ax = b\}$$

- l_1-regularized least-squares model:

$$\min \tau \|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2$$

A Reformulation of the $l^1 - l^2$ Model

$$\min_x \ \tau\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2$$

$$\Updownarrow \ \text{By introducing } y$$

$$\begin{array}{ll} \min & \tau\|x\|_1 + \frac{1}{2}\|Ay - b\|_2^2 \\ \text{s.t.} & x = y. \end{array}$$

Solutions of ADMM's Subproblems

$$\begin{array}{ll} \min & \tau\|x\|_1 + \frac{1}{2}\|Ay - b\|_2^2 \\ \text{s.t.} & x = y. \end{array}$$

Solutions of ADMM's Subproblems

$$\begin{array}{ll} \min & \tau\|x\|_1 + \frac{1}{2}\|Ay-b\|_2^2 \\ \text{s.t.} & x = y. \end{array}$$

1. $x^{k+1} = \arg\min_{x\in\mathcal{R}^n} \tau\|x\|_1 + \frac{\beta}{2}\left\|x - y^k - \frac{\lambda^k}{\beta}\right\|_2^2;$
2. y^{k+1}: $(\beta I + A^T A)y = A^T b + \beta x^{k+1} - \lambda^k;$
3. $\lambda^{k+1} = \lambda^k - \beta\left(x^{k+1} - y^{k+1}\right)$

Solutions of ADMM's Subproblems

$$\begin{array}{ll} \min & \tau\|x\|_1 + \frac{1}{2}\|Ay - b\|_2^2 \\ \text{s.t.} & x = y. \end{array}$$

1. $x^{k+1} = \arg\min_{x \in \mathcal{R}^n} \tau\|x\|_1 + \frac{\beta}{2}\left\|x - y^k - \frac{\lambda^k}{\beta}\right\|_2^2;$
2. y^{k+1}: $(\beta I + A^T A)y = A^T b + \beta x^{k+1} - \lambda^k;$
3. $\lambda^{k+1} = \lambda^k - \beta\left(x^{k+1} - y^{k+1}\right)$

P1 is a soft-shrinkage operator

P2 is a system of linear equations, efficient solvers (e.g. PCG or BB) are available

Another ADMM Application

(2) Image deblurring

A clean image could be degraded by blur — defocus of the camera's lens, the moving object, turbulence in the air, $\cdots$

Another ADMM Application

(2) Image deblurring

A clean image could be degraded by blur — defocus of the camera's lens, the moving object, turbulence in the air, $\cdots$

$$\min \||\nabla x|\|_1 + \frac{\mu}{2}\|Kx - x^0\|^2,$$

where x is the clean image, x^0 is the corrupted image by Gaussian noise, K is the point spread function (blur), ∇ is a gradient operator (by Rudin/Osher/Fatemi, 92') to preserve sharp edges of an image, and μ is a trade-off parameter.

Another ADMM Application

(2) Image deblurring

A clean image could be degraded by blur — defocus of the camera's lens, the moving object, turbulence in the air, $\cdots$

$$\min \||\nabla x|\|_1 + \frac{\mu}{2}\|Kx - x^0\|^2,$$

where x is the clean image, x^0 is the corrupted image by Gaussian noise, K is the point spread function (blur), ∇ is a gradient operator (by Rudin/Osher/Fatemi, 92') to preserve sharp edges of an image, and μ is a trade-off parameter.

original image

blurred image

restored image

Applying ADMM

Reformulate it as

$$\begin{aligned} \min \; & \| |y| \|_1 + \frac{\mu}{2} \|Kx - x^0\|^2 \\ \text{s.t. } & \nabla x = y, \end{aligned}$$

to which ADMM is applicable.

Applying ADMM

Reformulate it as

$$\begin{aligned} \min \; & \| |y| \|_1 + \frac{\mu}{2} \|Kx - x^0\|^2 \\ \text{s.t. } & \nabla x = y, \end{aligned}$$

to which ADMM is applicable.
The resulting subproblems are easy.

Applying ADMM

Reformulate it as

$$\min \||y\||_1 + \frac{\mu}{2}\|Kx - x^0\|^2$$
$$\text{s.t. } \nabla x = y,$$

to which ADMM is applicable.
The resulting subproblems are easy.

- The x-subproblem (via a DFT):

$$\tilde{x}^k = \arg\min_x \left\{ \frac{\mu}{2}\|Kx - x^0\|^2 - (\lambda^k)^T(\nabla x - y^k) + \frac{\beta}{2}\|\nabla x - y^k\|^2 \right\}.$$

Applying ADMM

Reformulate it as

$$\begin{aligned} \min\ & |||y|||_1 + \frac{\mu}{2}\|Kx - x^0\|^2 \\ \text{s.t. } & \nabla x = y, \end{aligned}$$

to which ADMM is applicable.
The resulting subproblems are easy.

- The x-subproblem (via a DFT):

$$\tilde{x}^k = \arg\min_x \left\{ \frac{\mu}{2}\|Kx - x^0\|^2 - (\lambda^k)^T(\nabla x - y^k) + \frac{\beta}{2}\|\nabla x - y^k\|^2 \right\}.$$

- The y-subproblem (via a shrinkage):

$$\tilde{y}^k = \arg\min_y \left\{ |||y|||_1 - (\lambda^{k+1})^T(\nabla x^{k+1} - y) + \frac{\beta}{2}\|\nabla x^{k+1} - y\|^2 \right\}.$$

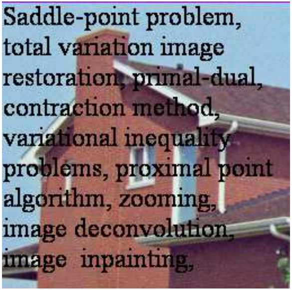
Image Inpainting

Problem: Some pixels are missing in image. Partial information of image is available

$\mathbf{g} = S\,\mathbf{f}, \quad S$ — mask

Model: $\min \ \{\|\nabla \mathbf{f}\|_1 \ | \ S\,\mathbf{f} = \mathbf{g}\}$

original image



missing pixel image

restored image

Image Decomposition

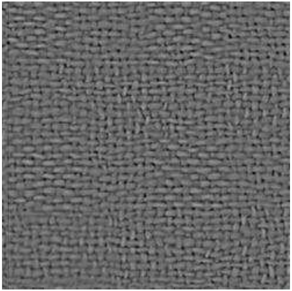
Problem: Separate the sketch (cartoon) and oscillating component (texture) of image

$\mathbf{f} = \mathbf{u} + \mathbf{v}$, $\mathbf{u}$ — cartoon part, $\mathbf{v}$ — texture part

Model: $\min \left\{ \tau \|\nabla \mathbf{u}\|_1 + \|\mathbf{v}\|_{-1,\infty} \;\middle|\; \mathbf{u} + \mathbf{v} = \mathbf{f} \right\}$

original image

cartoon part



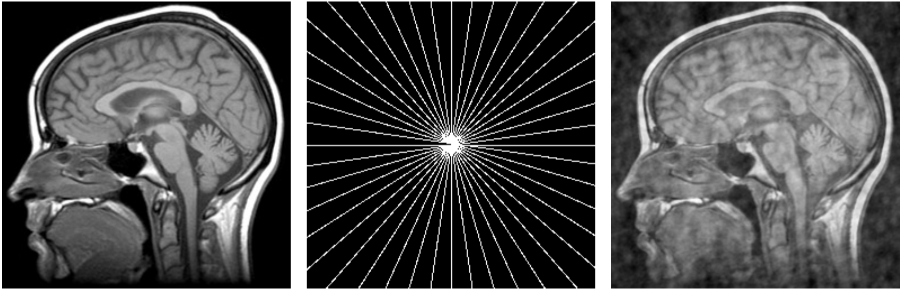
texture part

Magnetic Resonance Imaging (MRI)

Problem: Reconstruct a medical image by sampling its Fourier coefficients partially

$\mathcal{F}\mathbf{g} = P\mathcal{F}\mathbf{f}$, P — sampling mask, $\mathcal{F}$ — Fourier transform

Model: min $\{\|\nabla \mathbf{f}\|_1 \mid \mathcal{F}\mathbf{g} = P\mathcal{F}\mathbf{f}\}$



medical image sampling mask reconstruction

Outline

A More Complicated Model with Higher Degree of Separability

A more complicated multi-block separable convex optimization model:

$$\min \left\{ \sum_{i=1}^{m} \theta_i(x_i) \;\middle|\; \sum_{i=1}^{m} A_i x_i = b,\; x_i \in \mathcal{X}_i,\; i = 1, 2, \cdots, m \right\},$$

with $m \geq 3$.

A More Complicated Model with Higher Degree of Separability

A more complicated multi-block separable convex optimization model:

$$\min \left\{ \sum_{i=1}^{m} \theta_i(x_i) \;\middle|\; \sum_{i=1}^{m} A_i x_i = b,\ x_i \in \mathcal{X}_i,\ i = 1, 2, \cdots, m \right\},$$

with $m \geq 3$.

Applications include

- Image alignment problem
- The robust principal component analysis model with noisy and incomplete data
- The latent variable Gaussian graphical model selection
- The quadratic discriminant analysis model

Splitting Versions with Less Accuracy while More Implementability

- Obviously, the parallel (Jacobian) splitting:

$$\begin{cases} x_1^{k+1} = \operatorname{argmin}\{\theta_1(x_1) - (\lambda^k)T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + \sum_{j=2}^{m} A_jx_j^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ \cdots\cdots \\ x_i^{k+1} = \operatorname{argmin}\{\theta_i(x_i) - (\lambda^k)^T(A_ix_i) + \frac{\beta}{2}\| \sum_{j=1}^{i-1} A_jx_j^k + A_ix_i + \sum_{j=i+1}^{m} A_jx_j^k - b\|^2 \mid x_i \in \mathcal{X}_i\}, \\ \cdots\cdots \\ x_m^{k+1} = \operatorname{argmin}\{\theta_m(x_m) - (\lambda^k)^T(A_mx_m) + \frac{\beta}{2}\| \sum_{j=1}^{m-1} A_jx_j^k + A_mx_m - b\|^2 \mid x_m \in \mathcal{X}_m\}, \\ \lambda^{k+1} = \lambda^k - \beta(\sum_{i=1}^{m} A_ix_i^{k+1} - b). \end{cases}$$

does not work (more details are coming).

Cont'd

- Can we extend ADMM straightforwardly (by splitting ALM into m subproblems sequentially)?

$$
\left\{
\begin{array}{l}
x_1^{k+1} = \operatorname{argmin}\{\theta_1(x_1) - (\lambda^k)T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + \sum_{j=2}^{m} A_jx_j^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\
\cdots\cdots \\
x_i^{k+1} = \operatorname{argmin}\{\theta_i(x_i) - (\lambda^k)^T(A_ix_i) + \frac{\beta}{2}\| \sum_{j=1}^{i-1} A_jx_j^{k+1} + A_ix_i + \sum_{j=i+1}^{m} A_jx_j^k - b\|^2 \mid x_i \in \mathcal{X}_i\}, \\
\cdots\cdots \\
x_m^{k+1} = \operatorname{argmin}\{\theta_m(x_m) - (\lambda^k)^T(A_mx_m) + \frac{\beta}{2}\| \sum_{j=1}^{m-1} A_jx_j^{k+1} + A_mx_m - b\|^2 \mid x_m \in \mathcal{X}_m\}, \\
\lambda^{k+1} = \lambda^k - \beta(\sum_{i=1}^{m} A_ix_i^{k+1} - b).
\end{array}
\right.
$$

Cont'd

- Can we extend ADMM straightforwardly (by splitting ALM into m subproblems sequentially)?

$$
\begin{cases}
x_1^{k+1} = \text{argmin}\{\theta_1(x_1) - (\lambda^k)T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + \sum_{j=2}^{m} A_jx_j^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\
\cdots\cdots \\
x_i^{k+1} = \text{argmin}\{\theta_i(x_i) - (\lambda^k)^T(A_ix_i) + \frac{\beta}{2}\|\sum_{j=1}^{i-1} A_jx_j^{k+1} + A_ix_i + \sum_{j=i+1}^{m} A_jx_j^k - b\|^2 \mid x_i \in \mathcal{X}_i\}, \\
\cdots\cdots \\
x_m^{k+1} = \text{argmin}\{\theta_m(x_m) - (\lambda^k)^T(A_mx_m) + \frac{\beta}{2}\|\sum_{j=1}^{m-1} A_jx_j^{k+1} + A_mx_m - b\|^2 \mid x_m \in \mathcal{X}_m\}, \\
\lambda^{k+1} = \lambda^k - \beta(\sum_{i=1}^{m} A_ix_i^{k+1} - b).
\end{cases}
$$

- This direct extension of the ADMM has been widely used in the literature; and it does work very well for many applications!

Cont'd

- Can we extend ADMM straightforwardly (by splitting ALM into m subproblems sequentially)?

$$
\left\{
\begin{array}{l}
x_1^{k+1} = \text{argmin}\{\theta_1(x_1) - (\lambda^k)T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + \sum_{j=2}^{m} A_jx_j^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\
\cdots\cdots \\
x_i^{k+1} = \text{argmin}\{\theta_i(x_i) - (\lambda^k)^T(A_ix_i) + \frac{\beta}{2}\| \sum_{j=1}^{i-1} A_jx_j^{k+1} + A_ix_i + \sum_{j=i+1}^{m} A_jx_j^k - b\|^2 \mid x_i \in \mathcal{X}_i\}, \\
\cdots\cdots \\
x_m^{k+1} = \text{argmin}\{\theta_m(x_m) - (\lambda^k)^T(A_mx_m) + \frac{\beta}{2}\| \sum_{j=1}^{m-1} A_jx_j^{k+1} + A_mx_m - b\|^2 \mid x_m \in \mathcal{X}_m\}, \\
\lambda^{k+1} = \lambda^k - \beta(\sum_{i=1}^{m} A_ix_i^{k+1} - b).
\end{array}
\right.
$$

- This direct extension of the ADMM has been widely used in the literature; and it does work very well for many applications!
- But for a very long time, neither affirmative convergence proof nor counter example showing its divergence was available.

- Recently we [2] found some examples showing the divergence of the direct extension of ADMM even when $m = 3$. So, the direct extension of ADMM for multi-block separable convex optimization model is not necessarily convergent!

[2] Chen/He/Ye/Y., The direct extension of ADMM for multi-block separable convex minimization models is not necessarily convergent, September 2013.

- Recently we [2] found some examples showing the divergence of the direct extension of ADMM even when $m = 3$. So, the direct extension of ADMM for multi-block separable convex optimization model is not necessarily convergent!
- That is, even to solve

$$\min\left\{\theta_1(x_1)+\theta_2(x_2)+\theta_3(x_3)\;\middle|\;A_1x_1+A_2x_2+A_3x_3=b,\; x_i\in\mathcal{X}_i,\; i=1,2,3\right\},$$

[2] Chen/He/Ye/Y., The direct extension of ADMM for multi-block separable convex minimization models is not necessarily convergent, September 2013.

- Recently we [2] found some examples showing the divergence of the direct extension of ADMM even when $m = 3$. So, the direct extension of ADMM for multi-block separable convex optimization model is not necessarily convergent!
- That is, even to solve

$$\min\left\{\theta_1(x_1)+\theta_2(x_2)+\theta_3(x_3)\;\middle|\; A_1x_1+A_2x_2+A_3x_3=b,\; x_i\in\mathcal{X}_i,\; i=1,2,3\right\},$$

the following scheme is not necessarily convergent:

$$\left\{\begin{array}{l} x_1^{k+1} = \operatorname{argmin}\{\theta_1(x_1) - (\lambda^k)T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\},\\ x_2^{k+1} = \operatorname{argmin}\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\},\\ x_3^{k+1} = \operatorname{argmin}\{\theta_3(x_3) - (\lambda^k)^T(A_3x_3) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\},\\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{array}\right.$$

[2] Chen/He/Ye/Y., The direct extension of ADMM for multi-block separable convex minimization models is not necessarily convergent, September 2013.

- Recently we [2] found some examples showing the divergence of the direct extension of ADMM even when $m = 3$. So, the direct extension of ADMM for multi-block separable convex optimization model is not necessarily convergent!
- That is, even to solve

$$\min \left\{ \theta_1(x_1) + \theta_2(x_2) + \theta_3(x_3) \;\middle|\; A_1x_1 + A_2x_2 + A_3x_3 = b,\; x_i \in \mathcal{X}_i,\; i = 1, 2, 3 \right\},$$

the following scheme is not necessarily convergent:

$$\left\{ \begin{array}{l} x_1^{k+1} = \mathrm{argmin}\{\theta_1(x_1) - (\lambda^k)T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \mathrm{argmin}\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \mathrm{argmin}\{\theta_3(x_3) - (\lambda^k)^T(A_3x_3) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{array} \right.$$

- Both Jacobian and Gauss-Seidel decompositions fail — too much loss of accuracy for $m \geq 3$!

[2] Chen/He/Ye/Y., The direct extension of ADMM for multi-block separable convex minimization models is not necessarily convergent, September 2013.

One Way of Applying the ADMM

- Conceptually, we can treat the multi-block model as a two-block model

$$\min \left\{ \theta_1(x_1) + \theta_2(x_2) + \theta_3(x_3) \;\middle|\; A_1x_1 + A_2x_2 + A_3x_3 = b,\ x_i \in \mathcal{X}_i,\ i = 1,2,3 \right\},$$

One Way of Applying the ADMM

- Conceptually, we can treat the multi-block model as a two-block model

$$\min\left\{\theta_1(x_1)+\theta_2(x_2)+\theta_3(x_3)\;\middle|\;A_1x_1+A_2x_2+A_3x_3=b,\; x_i\in\mathcal{X}_i,\; i=1,2,3\right\},$$

- Then, apply the original ADMM (for the two-block case)

$$\left\{\begin{array}{l} x_1^{k+1}=\arg\min\{\theta_1(x_1)-(\lambda^k)^T(A_1x_1)+\frac{\beta}{2}\|A_1x_1+A_2x_2^k+A_3x_3^k-b\|^2 \mid x_1\in\mathcal{X}_1\},\\ (x_2^{k+1},x_3^{k+1})=\arg\min\left\{\begin{array}{l}\theta_2(x_2)+\theta_3(x_3)-(\lambda^k)^T(A_2x_2+A_3x_3-b)\\ \quad+\frac{\beta}{2}\|A_1x_1^{k+1}+A_2x_2+A_3x_3-b\|^2 \mid x_2\in\mathcal{X}_2,\; x_3\in\mathcal{X}_3\end{array}\right\},\\ \lambda^{k+1}=\lambda^k-\alpha\beta(A_1x_1^{k+1}+A_2x_2^{k+1}+A_3x_3^{k+1}-b).\end{array}\right.$$

One Way of Applying the ADMM

- Conceptually, we can treat the multi-block model as a two-block model

$$\min\left\{\theta_1(x_1)+\theta_2(x_2)+\theta_3(x_3)\;\middle|\; A_1x_1+A_2x_2+A_3x_3=b,\; x_i\in\mathcal{X}_i,\; i=1,2,3\right\},$$

- Then, apply the original ADMM (for the two-block case)

$$\left\{\begin{array}{l} x_1^{k+1}=\arg\min\{\theta_1(x_1)-(\lambda^k)^T(A_1x_1)+\frac{\beta}{2}\|A_1x_1+A_2x_2^k+A_3x_3^k-b\|^2 \mid x_1\in\mathcal{X}_1\},\\ (x_2^{k+1},x_3^{k+1})=\arg\min\left\{\begin{array}{l}\theta_2(x_2)+\theta_3(x_3)-(\lambda^k)^T(A_2x_2+A_3x_3-b)\\ \quad+\frac{\beta}{2}\|A_1x_1^{k+1}+A_2x_2+A_3x_3-b\|^2 \mid x_2\in\mathcal{X}_2,\; x_3\in\mathcal{X}_3\end{array}\right\},\\ \lambda^{k+1}=\lambda^k-\alpha\beta(A_1x_1^{k+1}+A_2x_2^{k+1}+A_3x_3^{k+1}-b).\end{array}\right.$$

- It is accurate (recall ADMM's convergence).

One Way of Applying the ADMM

- Conceptually, we can treat the multi-block model as a two-block model

$$\min\left\{\theta_1(x_1)+\theta_2(x_2)+\theta_3(x_3)\;\middle|\;A_1x_1+A_2x_2+A_3x_3=b,\;x_i\in\mathcal{X}_i,\;i=1,2,3\right\},$$

- Then, apply the original ADMM (for the two-block case)

$$\left\{\begin{array}{l} x_1^{k+1}=\arg\min\{\theta_1(x_1)-(\lambda^k)^T(A_1x_1)+\frac{\beta}{2}\|A_1x_1+A_2x_2^k+A_3x_3^k-b\|^2\mid x_1\in\mathcal{X}_1\},\\ (x_2^{k+1},x_3^{k+1})=\arg\min\left\{\begin{array}{l}\theta_2(x_2)+\theta_3(x_3)-(\lambda^k)^T(A_2x_2+A_3x_3-b)\\ \quad+\frac{\beta}{2}\|A_1x_1^{k+1}+A_2x_2+A_3x_3-b\|^2\mid x_2\in\mathcal{X}_2,\;x_3\in\mathcal{X}_3\end{array}\right\},\\ \lambda^{k+1}=\lambda^k-\alpha\beta(A_1x_1^{k+1}+A_2x_2^{k+1}+A_3x_3^{k+1}-b).\end{array}\right.$$

- It is accurate (recall ADMM's convergence).
- But it is not implementable (hard to solve the (x_2,x_3)-subproblem).

ADMM with Further Splitting

- Split the (x_2, x_3)-subproblem in parallel

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^k + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^k + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

ADMM with Further Splitting

- Split the (x_2, x_3)-subproblem in parallel

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^k + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^k + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

- Split the (x_2, x_3)-subproblem sequentially

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^{k+1} + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

ADMM with Further Splitting

- Split the (x_2, x_3)-subproblem in parallel

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^k + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^k + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

- Split the (x_2, x_3)-subproblem sequentially

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^{k+1} + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

- Both are implementable,

ADMM with Further Splitting

- Split the (x_2, x_3)-subproblem in parallel

$$
\begin{cases}
x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\
x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\
x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^k + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^k + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\
\lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b).
\end{cases}
$$

- Split the (x_2, x_3)-subproblem sequentially

$$
\begin{cases}
x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\
x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\
x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^{k+1} + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\
\lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b).
\end{cases}
$$

- Both are implementable, but how about the accuracy?

ADMM with Further Splitting

- Split the (x_2, x_3)-subproblem in parallel

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^k + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^k + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

- Split the (x_2, x_3)-subproblem sequentially

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^{k+1} + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

- Both are implementable, but how about the accuracy?
- Both are not necessarily convergent (Liu/Lu/Y., in pending)

ADMM with Further Splitting

- Split the (x_2, x_3)-subproblem in parallel

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^k + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^k + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

- Split the (x_2, x_3)-subproblem sequentially

$$\begin{cases} x_1^{k+1} = \arg\min\{\theta_1(x_1) - (\lambda^k)^T(A_1x_1) + \frac{\beta}{2}\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\|^2 \mid x_1 \in \mathcal{X}_1\}, \\ x_2^{k+1} = \arg\min\{\theta_2(x_2) - (\lambda^k)^T(A_2x_2 + A_3x_3^k - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\|^2 \mid x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\theta_3(x_3) - (\lambda^k)^T(A_2x_2^{k+1} + A_3x_3 - b) + \frac{\beta}{2}\|A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3 - b\|^2 \mid x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+1} = \lambda^k - \alpha\beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). \end{cases}$$

- Both are implementable, but how about the accuracy?
- Both are not necessarily convergent (Liu/Lu/Y., in pending)
- Implementable but not accurate!

Convergence-guarantee

How to guarantee the convergence while remain the implementability?

Convergence-guarantee

How to guarantee the convergence while remain the implementability?

- Correct the output of the decomposed subproblems, see our work in 2011-2013.

Convergence-guarantee

How to guarantee the convergence while remain the implementability?

- Correct the output of the decomposed subproblems, see our work in 2011-2013.
- Proximally regularized the decomposed subproblems (this works even when the ALM subproblem is decomposed in parallel), see He/Xu/Y., Deng/Lai/Pang/Yin, Wang/Hong/Ma/Luo, etc.

Accuracy Improvement

Accuracy Improvement

For the convergence-guaranteed and implementability-preserved algorithms,

Accuracy Improvement

For the convergence-guaranteed and implementability-preserved algorithms,

- How to design inexact criteria for the subproblems for the general setting? (Y., ongoing)

Accuracy Improvement

For the convergence-guaranteed and implementability-preserved algorithms,

- How to design inexact criteria for the subproblems for the general setting? (Y., ongoing)
- Do we really need to decompose m times?

Accuracy Improvement

For the convergence-guaranteed and implementability-preserved algorithms,

- How to design inexact criteria for the subproblems for the general setting? (Y., ongoing)
- Do we really need to decompose m times? — How about decomposing less blocks thus preserve more accuracy of the subproblems?

Accuracy Improvement

For the convergence-guaranteed and implementability-preserved algorithms,

- How to design inexact criteria for the subproblems for the general setting? (Y., ongoing)
- Do we really need to decompose m times? — How about decomposing less blocks thus preserve more accuracy of the subproblems?
- We can regroup m block as t blocks with $t \ll m$, apply existing methods for the t-block reformulated model to gain the accuracy (i.e., the proved convergence) and further decompose each subproblem to gain the implementability

Accuracy Improvement

For the convergence-guaranteed and implementability-preserved algorithms,

- How to design inexact criteria for the subproblems for the general setting? (Y., ongoing)
- Do we really need to decompose m times? — How about decomposing less blocks thus preserve more accuracy of the subproblems?
- We can regroup m block as t blocks with $t \ll m$, apply existing methods for the t-block reformulated model to gain the accuracy (i.e., the proved convergence) and further decompose each subproblem to gain the implementability —-(He/Y. and Fu/He/Wang/Y.'s work in August 2014)

Outline

Conclusions

- Accuracy and implementability are two common yet usually conflicted objectives in algorithmic design.

Conclusions

- Accuracy and implementability are two common yet usually conflicted objectives in algorithmic design.
- We show by some convex optimization models with strong application backgrounds (imaging, learning, cloud computing, big data, etc.) how to consider these two objectives.

Conclusions

- Accuracy and implementability are two common yet usually conflicted objectives in algorithmic design.
- We show by some convex optimization models with strong application backgrounds (imaging, learning, cloud computing, big data, etc.) how to consider these two objectives.
- Interesting theoretical questions arise, such as the convergence rate analysis (introducing some new analytic tools like variational analysis).

Conclusions

- Accuracy and implementability are two common yet usually conflicted objectives in algorithmic design.
- We show by some convex optimization models with strong application backgrounds (imaging, learning, cloud computing, big data, etc.) how to consider these two objectives.
- Interesting theoretical questions arise, such as the convergence rate analysis (introducing some new analytic tools like variational analysis).
- Extendable to more areas (e.g., PDE or PDE-constrained optimization (control) problems).

Conclusions

- Accuracy and implementability are two common yet usually conflicted objectives in algorithmic design.
- We show by some convex optimization models with strong application backgrounds (imaging, learning, cloud computing, big data, etc.) how to consider these two objectives.
- Interesting theoretical questions arise, such as the convergence rate analysis (introducing some new analytic tools like variational analysis).
- Extendable to more areas (e.g., PDE or PDE-constrained optimization (control) problems).
- Application-driven optimization makes sense!

Thank you!

xmyuan@hkbu.edu.hk