Where does operator splitting come from

Andersen Ang

Mathématique et recherche opérationnelle
UMONS, Belgium

manshun.ang@umons.ac.be Homepage: angms.science

First draft : November 21, 2018
Last update : November 21, 2018

Solving linear system

Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, solve for $\mathbf{x} \in \mathbb{R}^n$ in $\mathbf{A}\mathbf{x} = \mathbf{b}$.

How to solve it?

Linear Algebra : use inverse $\mathbf{x} = \mathbf{A}^\dagger \mathbf{b}$.

Let $n = m$ and $\mathbf{A}$ is invertible, the key step in solving for $\mathbf{x}$ is to compute :

$$\mathbf{A}^{-1}$$

Computing $\mathbf{A}^{-1}$ has many numerical issues :

- if m is big (and $\mathbf{A}$ has no specific structure), then inverting $\mathbf{A}$ is generally expensive
- if $\mathbf{A}$ is ill-conditioned (conditional number of $\mathbf{A}$ is big), then inverting $\mathbf{A}$ is prone to large numerical errors

What can we do?

Computing inverse of $\mathbf{A}$ by splitting

Such problem can be solved by the method of splitting[1] : decomposes the "problematic" matrix $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{B} + \mathbf{C},$$

where the inverses of $\mathbf{B}$ is easier to do, less error prone.
What about $\mathbf{C}$: treat it as a residue/error.

Now the $\mathbf{Ax} = \mathbf{b}$ becomes

put $\mathbf{A} = \mathbf{B} + \mathbf{C}$	$(\mathbf{B} + \mathbf{C})\mathbf{x} = \mathbf{b}$
in fixed-point equation form	$\mathbf{Bx} = \mathbf{b} - \mathbf{Cx}$
in fixed-point iteration form	$\mathbf{x}_{k+1} = \mathbf{B}^{-1}(\mathbf{b} - \mathbf{Cx}_k)$

What if $\mathbf{B}$ and $\mathbf{C}$ are also ill-conditioned?

[1] The other way to attach the problem is to use Kyrlov subspace.

Computing inverse of $\mathbf{A}$ by splitting with enhancement

If $\mathbf{B}$ and $\mathbf{C}$ are also ill-conditioned, we can enhance the condition of $\mathbf{B}$ and $\mathbf{C}$ by performing the following splitting of $\mathbf{A}$:

$$\mathbf{A} \quad = \quad \mathbf{B} + \alpha\mathbf{I} + \mathbf{C} - \alpha\mathbf{I}.$$

If α is large enough : the inverses of $(\mathbf{B} + \alpha\mathbf{I})$ is easier to do, less error prone and it is not ill-conditioned.

Now the $\mathbf{Ax} = \mathbf{b}$ becomes

put $\mathbf{A} = \mathbf{B} + \alpha\mathbf{I} + \mathbf{C} - \alpha\mathbf{I}$	$(\mathbf{B} + \alpha\mathbf{I} + \mathbf{C} - \alpha\mathbf{I})\mathbf{x} = \mathbf{b}$
in fixed-point equation form	$(\mathbf{B} + \alpha\mathbf{I})\mathbf{x}_{k+1} = \mathbf{b} - (\mathbf{C} - \alpha\mathbf{I})\mathbf{x}_k$

We know α has to be large, but how to large? how to determine α?

Determine α

For $\mathbf{A} = \mathbf{B} + \alpha\mathbf{I} + \mathbf{C} - \alpha\mathbf{I}$, $\mathbf{Ax} = \mathbf{b}$ becomes
$((\mathbf{B} + \alpha\mathbf{I}) + (\mathbf{C} - \alpha\mathbf{I}))\mathbf{x} = \mathbf{b}$ or $((\mathbf{C} + \alpha\mathbf{I}) + (\mathbf{B} - \alpha\mathbf{I}))\mathbf{x} = \mathbf{b}$.
We get two fixed-point equations:

$$\begin{aligned} \mathbf{B} \text{ as subject} \quad & (\mathbf{B} + \alpha\mathbf{I})\mathbf{x}_{k+1} = \mathbf{b} - (\mathbf{C} - \alpha\mathbf{I})\mathbf{x}_k \\ \mathbf{C} \text{ as subject} \quad & (\mathbf{C} + \alpha\mathbf{I})\mathbf{x}_{k+1} = \mathbf{b} - (\mathbf{B} - \alpha\mathbf{I})\mathbf{x}_k \end{aligned}$$

The line of thought :

- We split $\mathbf{A}$ as a sum of $\mathbf{B}$ and $\mathbf{C}$ for the computation of the inverse of the problematic matrix $\mathbf{A}$.
- To prevent $\mathbf{B}$ and $\mathbf{C}$ also being "problematic", we use the term $\alpha\mathbf{I}$ to make $\mathbf{B} + \alpha\mathbf{I}$ and $\mathbf{C} + \alpha\mathbf{I}$ numerically invertible (not ill-conditioned).
- To make $\mathbf{B} + \alpha\mathbf{I}$ **and** $\mathbf{C} + \alpha\mathbf{I}$ both well-conditioned, we can set

  $$\alpha = \max\{\alpha_B, \alpha_C\},$$

  where the conditional number (the number which measures how "problematic" a matrix is) of $\mathbf{B} + \alpha_B\mathbf{I}$ and $\mathbf{C} + \alpha_C\mathbf{I}$ are both smaller than a desired value

Computing inverse of $\mathbf{A}$ by splitting

Recall our original problem :
given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, solve for $\mathbf{x} \in \mathbb{R}^n$ in $\mathbf{Ax} = \mathbf{b}$.
After splitting, we now have

$$(\mathbf{B} + \alpha\mathbf{I})\mathbf{x}_{k+1} = \mathbf{b} - (\mathbf{C} - \alpha\mathbf{I})\mathbf{y}_k$$
$$(\mathbf{C} + \alpha\mathbf{I})\mathbf{y}_{k+1} = \mathbf{b} - (\mathbf{B} - \alpha\mathbf{I})\mathbf{x}_k$$

or

$$\mathbf{x}_{k+1} = (\mathbf{B} + \alpha\mathbf{I})^{-1}(\mathbf{b} + (\alpha\mathbf{I} - \mathbf{C})\mathbf{y}_k)$$
$$\mathbf{y}_{k+1} = (\mathbf{C} + \alpha\mathbf{I})^{-1}(\mathbf{b} + (\alpha\mathbf{I} - \mathbf{B})\mathbf{x}_k)$$

That is, we run an iterative scheme of above, the sequence $\{\mathbf{x}_k\}_{k\in\mathbb{N}}$ (or $\{\mathbf{y}_k\}_{k\in\mathbb{N}}$) will converge to the $\mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b}$.

Splitting in optimization

Sub-gradient characterization of first order optimality condition
In unconstrained optimization, the first order optimality condition for differentiable f

$$f(\mathbf{x}^*) = \inf_{\mathbf{x}} f(\mathbf{x}) \iff 0 = \nabla f(\mathbf{x}^*).$$

Generalize to non-differentiable f, we use sub-gradient characterization

$$f(\mathbf{x}^*) = \inf_{\mathbf{x}} f(\mathbf{x}) \iff 0 \in \partial f(\mathbf{x}^*)$$

where ∂f is the sub-differential.
For unconstrained optimization in the form $\min f(\mathbf{x}) + g(\mathbf{x})$, the first order optimality condition using sub-gradient characterization is

$$0 \in (\partial f + \partial g)(\mathbf{x}^*)$$

Recall that $\mathbf{A}^{-1}$ can be computed via $\mathbf{A} = \mathbf{B} + \mathbf{C}$. Here we see splitting $\partial f + \partial g$ in optimization !

End of document