クリロフ部分空間法の左前処理と右前処理の比較

井上, 拓樹 / INOUE, Hiroki

(出版者 / Publisher)
法政大学大学院理工学・工学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 理工学・工学研究科編 / 法政大学大学院紀要. 理工学・工学研究科編
(巻 / Volume)
55
(開始ページ / Start Page)
1
(終了ページ / End Page)
4
(発行年 / Year)
2014-03-24
(URL)
https://doi.org/10.15002/00010459

クリロフ部分空間法の左前処理と右前処理の比較

THE COMPARISON OF RIGHT PRECONDITIONED AND LEFT PRECONDITIONED IN KRYLOV SUBSPACE METHODS

井上 拓樹
Hiroki INOUE
指導教員　堀端　康善

法政大学大学院工学研究科情報電子工学専攻修士課程

To the problem of large-scale systems of linear equations, we describe the comparison of left preconditioned and right preconditioned. We used the preconditioned method using the left and right for each methods of Krylov subspace methods, and inspect the difference of left preconditioned and right preconditioned.

KeyWords : *Krylov subspace method, left precondtioned method, right precondtioned method*

1.　はじめに

複雑な形状の計算領域で有限差分法を使って楕円型偏微分方程式を解く場合、差分法を使って方程式を離散化しようとすると境界の処理が困難となることから、一般座標変換を用いて簡単な長方形の領域に写像し、その上で有限差分法を適用するのが効果的である。この場合、離散化して得られた連立一次方程式

$$Ax = b \tag{1}$$

の係数行列 A は非対称行列で、直交座標で離散化したときに比べ非零要素の対角線の本数が増える。このような方程式を解く方法としてクリロフ部分空間法を使用するケースが非常に多い。さらにそれの高速化として前処理を施すことが一般的であるであるが、この前処理というのは複数の種類がある。そこで本研究では、左前処理と右前処理をとりあげる。クリロフ部分空間法に実装し、両者の比較・検討を行う。

2.　計算領域と境界条件

本論文で実際に解く境界値問題としては、扇型の計算領域を図 1 のように一般座標変換し、ラプラス方程式

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = 0 \tag{2}$$

を、図 2, 3, 4 のように $r_x = 1.0$、2.0、3.0 と計算領域を歪ませていく 3 ケースについてディリクレ型境界条件の元で解く。なお、以下のようなディリクレ型境界条件を使用する。

$$onWX, \quad \phi = 0, \tag{3}$$
$$onXY, \quad \phi = \frac{\sin\theta}{r_{XY}}, \tag{4}$$
$$onYZ, \quad \phi = \frac{1}{r_{YZ}}, \tag{5}$$
$$onZW, \quad \phi = \frac{\sin\theta}{r_{WZ}} \tag{6}$$

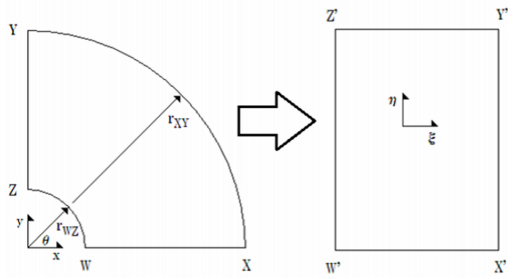

図 1: 計算領域と座標変換

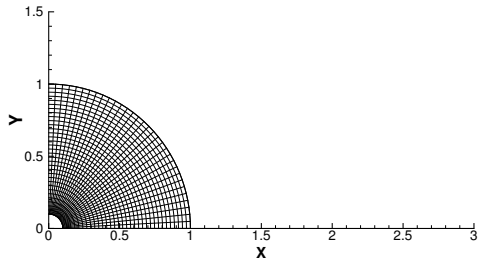

図 2: r_x=1.0 としたときの格子

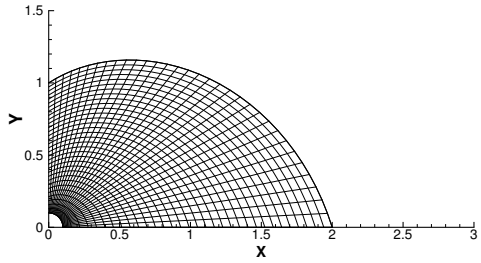

図 3: r_x=2.0 としたときの格子

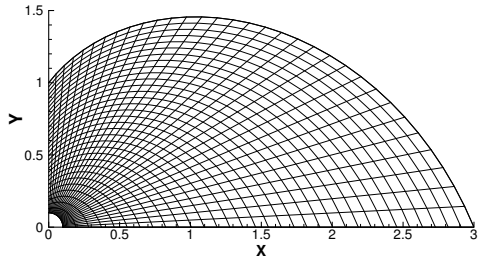

図 4: r_x=3.0 としたときの格子

3. 左前処理と右前処理

前処理とは連立方程式 $Ax = b$ に、A に近い前処理行列 C を用意しそれを作用さえる事により、数値解放の高速化をすることである。前処理にはいくつか種類があり、本論文では左前処理と右前処理を対象とした。

左前処理とは前処理行列 C を $C^{-1}Ax = C^{-1}b$ のように左から作用させることである。同様に右前処理は右から作用させ、$AC^{-1}Cx = b$ となり、わかりやすさの為 $Cx = u$ と置く。最終的に $A^{-1}u = b$ となる。

左右前処理の一番の違いは残差ベクトルにあり、前処理をしないオリジナルの残差ベクトルは、$r = b - Ax$ であるが、左前処理では $r = C^{-1}(b - Ax)$ となる。右前処理では $r = b - AC^{-1}u$ で、u を戻すと $r = b - Ax$ となり、オリジナルと同じ残差ベクトルになる。この点が左右前処理の一番の違いである。

4. 数値実験

離散化して得られた大規模な連立一次方程式を左右前処理を施したクリロフ部分空間法で解く。計算領域を rx=1.0、rx=2.0、rx=3.0 としたときについて、それぞれ格子数を 257 × 257、513 × 513、769 × 769 にした場合の数値実験を行った。

また、左前処理において残差ベクトルを $r = C^{-1}(b - Ax)$ でなく、オリジナルと同じ残差ベクトルを計算しなおして使用している。これは左右前処理を比較するうえで条件を等しくするためである。

クリロフ部分空間法として、GMRES(k) 法、そして CG 系統の解法から BCG 法、CGS 法、BCGSTAB 法、BCGSTAB2 法、GPBCG 法、BCGSafe 法、CR 系統の解放からは BCR 法、CRS 法、BCRSTAB 法、BCRSTAB2 法、GPBCR 法、BCRSafe 法を用いた。各種解法に左前処理を施したものには LP を、右前処理を施したものには RP を頭につけることで差別化をする。

各解法で左右前処理による CPU 時間について比較したものは表 2、反復回数について比較したものは表 3 に示す。格子数 257 × 257、513 × 513 は割愛し、未知数が最も多い格子数 769 × 769 のときの結果のみを示す。

また、数値実験に用いた計算機の環境は、表 1 の通りである。

表 1: 実験に用いた計算機環境

計算機	HP xw8400 Workstation
CPU	Dual-core Intel Xeon 3.0GHz
OS	Red hat Linux 4
Compiler	Intel Fortran Compiler 9.1
メモリ	2GB
精度	倍精度
収束判定条件	$\|\|\mathbf{b} - A\mathbf{x}_k\|\|/\|\|\mathbf{b}\|\| < 10^{-12}$

表 2: 左右前処理による各解法の CPU 時間 [s]

解法	rx の値	左前処理	右前処理
BCG	rx = 1	89.5	83.5
	rx = 2	76.9	68.7
	rx = 3	65.8	58.3
BCR	rx = 1	126	112
	rx = 2	109	98.7
	rx = 3	96.2	87.5
CGS	rx = 1	64.3	59.8
	rx = 2	57.4	51.9
	rx = 3	52.3	45.7
CRS	rx = 1	69.1	64.1
	rx = 2	59.3	55.9
	rx = 3	51.3	46.9
BCGSTAB	rx = 1	69.7	62.3
	rx = 2	58.5	51.0
	rx = 3	51.8	45.4
BCRSTAB	rx = 1	78.0	72.2
	rx = 2	59.8	55.9
	rx = 3	56.4	49.2
GPBCG	rx = 1	89.3	101
	rx = 2	73.6	77.6
	rx = 3	67.2	71.3
GPBCR	rx = 1	110	130
	rx = 2	79.6	82.3
	rx = 3	76.0	70.5
BCGSTAB2	rx = 1	98.4	103
	rx = 2	80.6	85.9
	rx = 3	69.6	73.7
BCRSTAB2	rx = 1	98.0	105
	rx = 2	83.1	89.6
	rx = 3	72.2	73.0
BCGSafe	rx = 1	87.3	89.7
	rx = 2	76.5	72.2
	rx = 3	68.4	66.8
BCRSafe	rx = 1	120	131
	rx = 2	97.1	94.3
	rx = 3	84.0	83.1
GMRES	rx = 1	1334	1291
	rx = 2	891	850
	rx = 3	814	768

表 3: 左右前処理による各解法の反復回数

解法	rx の値	左前処理	右前処理
BCG	rx = 1	611	583
	rx = 2	527	494
	rx = 3	452	412
BCR	rx = 1	602	571
	rx = 2	533	494
	rx = 3	452	401
CGS	rx = 1	447	394
	rx = 2	373	318
	rx = 3	309	264
CRS	rx = 1	418	387
	rx = 2	356	334
	rx = 3	319	287
BCGSTAB	rx = 1	431	383
	rx = 2	358	313
	rx = 3	311	288
BCRSTAB	rx = 1	427	414
	rx = 2	381	356
	rx = 3	329	291
GPBCG	rx = 1	410	389
	rx = 2	337	291
	rx = 3	298	267
GPBCR	rx = 1	410	409
	rx = 2	337	304
	rx = 3	308	273
BCGSTAB2	rx = 1	445	407
	rx = 2	370	338
	rx = 3	321	285
BCRSTAB2	rx = 1	465	413
	rx = 2	373	354
	rx = 3	322	299
BCGSafe	rx = 1	397	352
	rx = 2	341	315
	rx = 3	307	266
BCRSafe	rx = 1	443	430
	rx = 2	361	340
	rx = 3	313	272
GMRES	rx = 1	3420	3300
	rx = 2	2460	2280
	rx = 3	1860	1720

5. まとめ

左右前処理の結果を比較すると、大きな違いは見られなかったものの右前処理のほうが全体的に CPU 時間が減少の傾向にあり、反復回数も減っていた。

一部解法において、右前処理のほうが CPU 時間が長いが反復回数が減っているという現象が起きている。これはアルゴリズムの中で方程式を解く回数が増えてしまったためであるが、反復回数は減っているので効果があったと言える。

右前処理は、残差もオリジナルのものと同じものが使われているのが大きな魅力であり、実験結果も良く出ているため、非常に有効な前処理だと分かった。

参考文献

[1] Pieter Wesseling : An Introduction Multigrid Methods, 2004

[2] Yousef Saad : Iterative Methods for Sparse Linear Systems SECOND EDITION, 2003