

領域分割法とその並列化

野田 敏達 小柳 義夫

東京大学大学院 理学系研究科 情報科学専攻

東京都文京区本郷 7-3-1

要旨

偏微分方程式を離散化し得られる大規模な連立一次方程式を解く方法の一つに領域分割法がある。領域をいくつかの小領域に分割して、各小領域とその境界の方程式を各々たてて解く解法である。2次元の場合、コレスキー分解や共役勾配法では、1辺のサイズが M の領域の計算量が $O(M^3)$ だが、領域を $M^{\frac{2}{3}}$ に分割し、小領域内をコレスキー分解、小領域間を前処理付共役勾配法を用いることにより、 $O(M^{\frac{4}{3}})$ で解けることが分かった。また、前処理として対角ブロック部分のコレスキー分解を用いる領域分割法は、良い収束性と高い並列性を持つ解法となる。本研究では、この解法を並列計算機上に実装し、高い並列化効率を得ることが出来た。

A Study of Domain Decomposition Method and it's Parallelization

Bintatsu Noda, Yoshio Oyanagi

Department of Information Science, Faculty of Science, University of Tokyo

7-3-1 Hongo, Bunkyo-ku, Tokyo 113

Abstract

To solve finite element problems which ensue from discretization of partial differential equations, a large system of linear equations must be solved. Domain Decomposition Method (DDM) is one of the methods to solve those problems by decomposing the domain into some subdomains. By partitioning the domain into subdomains, the dimension of the linear equations can be much less than that of the original problem. While the computational complexity of Cholesky Decomposition and Preconditioned Conjugate Gradient Method (PCG) to solve problems on two-dimensional domains is $O(M^3)$, this study shows the complexity of DDM is $O(M^{\frac{4}{3}})$. This method can achieve high parallelism by using block diagonal preconditioned CG method. It is verified by implementing it on parallel computer.

1 はじめに

自然現象は、ポアソン方程式などの偏微分方程式によって定式化されることが多い。これらの方程式を離散化して数値的に解く場合、非常に大きな連立一次方程式を解く必要がある。一般的に、これを解くのに用いられる前処理付共役勾配法や、コレスキー分解などでは、一辺のサイズが M の矩形領域を解くのに必要な計算量は $O(M^3)$ である。本研究では、1870 年 Schwarz によって考案された、領域分割法を用いた。これは領域を小さく分割することにより、問題サイズを小さくして解く方法である。近年では、Keyes らの並列化の研究 [4] などが行われている。本研究では、小領域内をコレスキー分解、小領域間を前処理付共役勾配法を用いることにより、 $O(M^{\frac{8}{3}})$ で解けることが分かった。

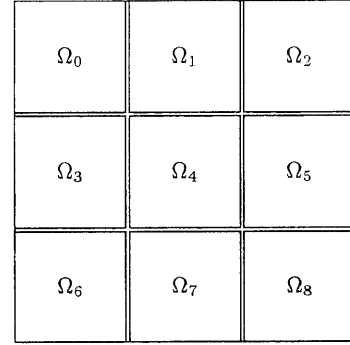


図 1 Domain Decomposition

2 問題のモデル

正方領域 $\Omega = (0, 1) \times (0, 1)$ で、ディレクレ境界条件をもつポアソン方程式

$$-k \nabla^2 u = f \quad \text{in } \Omega \quad (1)$$

ただし、

$$\nabla = \frac{\partial}{\partial x} + \frac{\partial}{\partial y}$$

を離散化した問題を考える。

3 領域分割法

3.1 領域分割法

式 (1) を 5 点中心差分で離散近似することによって

$$Au = f \quad (2)$$

という式が得られる。このとき、 A は正値対称行列であり、 u は求めるべき Ω 上の各格子点の値である。

今、図 1 のように、 Ω はいくつかの接しない小領域 ($\Omega_0, \Omega_1, \dots, \Omega_{R-1}$ 、図 1 では $R = 9$) と、それらの小領域に含まれない内部境界 (Ω_R) の集合と考える。つまり、 Ω を $R + 1$ 個の小領域に分割する。

各小領域 ($\Omega_0, \dots, \Omega_{R-1}$) 内の各点は、その小領域内の他の点と内部境界 Ω_R の点以外の影響を受けないので、先に内部境界 Ω_R を求めてしまえば、各小領域 (Ω_r) はそれぞれ、独立に解ける。

今、各小領域 (Ω_r) 内の点の集合を (u_r) とすると、

$$\begin{pmatrix} A_{00} & O & O & \cdots & A_{0R} \\ O & A_{11} & O & \cdots & A_{1R} \\ O & O & A_{22} & \cdots & A_{2R} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{R0} & A_{R1} & A_{R2} & \cdots & A_{RR} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_R \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_R \end{pmatrix} \quad (3)$$

と書ける。

つまり、小領域 Ω_r の方程式は、

$$A_{rr}u_r + A_{rR}u_R = f_r \quad (4)$$

と書け、内部境界 Ω_R の方程式は、

$$\sum_{r=0}^{R-1} A_{Rr}u_r + A_{RR}u_R = f_R \quad (5)$$

と書ける。この方程式 (5) から、 u_r を消去すると、 u_R だけの方程式ができ、

$$Cu_R = s_R \quad (6)$$

ただし、 C は capacitance 行列で、 C と s_R は以下のよう表せる。

$$C = A_{RR} - \sum_{r=0}^{R-1} A_{Rr}A_{rr}^{-1}A_{rR} \quad (7)$$

$$s_R = f_R - \sum_{r=0}^{R-1} A_{Rr}A_{rr}^{-1}f_r \quad (8)$$

これを capacitance system [2] と呼ぶ。

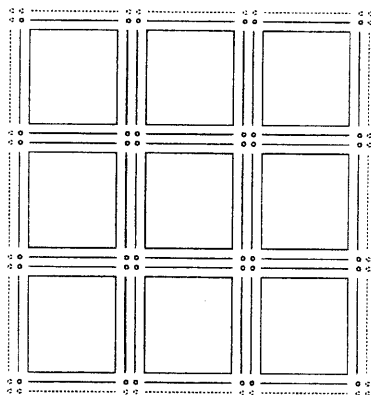


図2 Two Lines Boundaries Between Subdomains

3.2 内部境界と capacitance 行列

capacitance 行列は小領域間の内部境界に依存するが、ここでは、図2のように境界を二列にとる方法を用いた。

内部境界を二列をとると、一列のときと比較して、capacitance 行列の大きさは、2倍になる。しかし、行列内部の構造は、図3のように、単純になる。

3.3 capacitance system の作成法

式(7)、式(8)を見れば分かるように、capacitance system を作成するには、各小領域(Ω_r)の係数行列(A_{rr})の逆行列を使わなければならない。これは、capacitance system を解いた後にも使われるので、コレスキー分解を用いた。

3.4 capacitance system の解法

領域分割法で解く場合、最も重要な問題は、いかにこの capacitance system を解くか、である。 $N \times N$ の領域を $M \times M$ に分割する場合、capacitance system のサイズは $4 \times M \times N$ と、元の行列($N \times N$)より非常に小さくなっているが、非ゼロ要素の数は変わらない。本研究では、並列化をしやすいするため、capacitance 行列の対角ブロックの逆行列を前処理として用いる共役勾配法を用いた。

4 並列化

4.1 capacitance 行列の作成

1領域に1台のプロセッサを割りあてる。capacitance 行列は、内点消去により作成されるので、各々独立に通信無

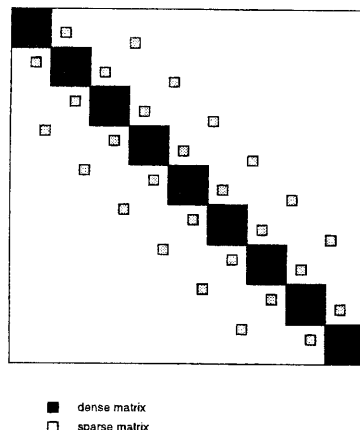


図3 The Structure of Capacitance Matrix of Two Lines Boundaries

して、作ることができる。全ての小領域の大きさが同じなら、この内点消去ではロードバランスがとれている。

4.2 capacitance system を解く

前処理付共役勾配法で解く。1回のループで値を更新するのに必要なデータは、接している領域の境界の値だけなので、通信は近接で行うことができ、またその量は問題サイズが $N \times N$ 、分割が $M \times M$ ならば、 $\frac{4 \times N}{M}$ である。また、1回の演算量 $O((\frac{N}{M})^2)$ である。この通信の隠蔽が並列化効率に大きく関係する。

4.3 小領域を解く

capacitance system を解くことにより、境界の値が求まれば、すでにコレスキー分解が行われているので、小領域の値は簡単に求まる。

5 数値実験

コレスキー分解、CG法、ICCG(1,1)法、領域分割法($2 \times 2, 4 \times 4, 8 \times 8$)の5種類について、メッシュサイズおよび拡散係数の値を変えて、それぞれ比較実験した。また、領域分割法の並列化効率を調べた。なお、コレスキー分解は、データ配置として、帯行列ではなく、メッシュサイズが $n \times n$ なら $O(n^3)$ の計算量ですむ nested dissection を用いている。

source term f の値を、単純なもの(図4)と、複雑なもの(図5)の2つについて実験した。それぞれ問題1、問題2

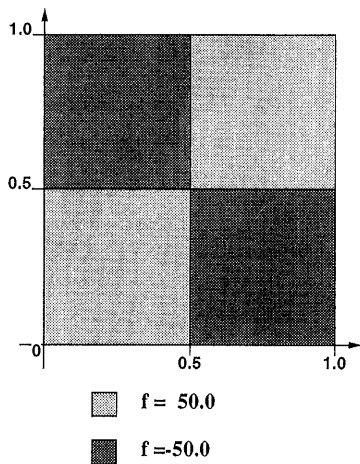


図 4 source term

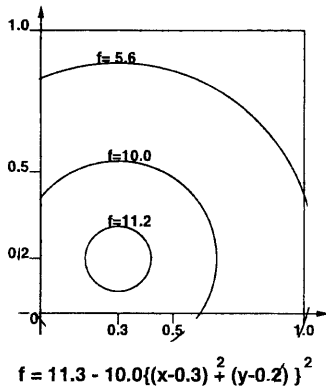


図 5 source term2

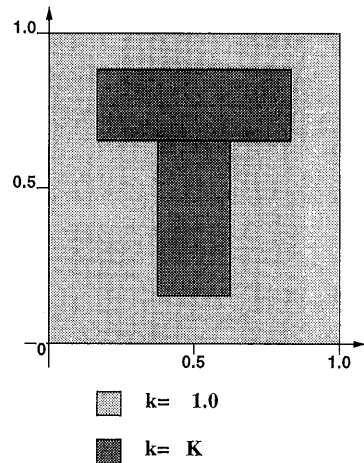


図 6 拡散係数

とする。

64×64 のメッシュサイズにおいて、図 6 の拡散係数 k を 1.0、1.1、10.0、100.0、1000.0 と変えて、実験した。

ディレクレ境界条件の値は全て 0 とする。

実験は SPARCstation20、AP1000+ 上で行った。

5.1 メッシュサイズによる比較

拡散係数 $k = 1.0$ および $k = 100$ において、メッシュサイズを 32×32、64×64、128×128 と値を変えて実験した。

表 1 は問題 1 の拡散係数が $k = 1.0$ である問題の、一定の反復回数および、浮動小数点演算数を表わしている。また表 2～表 4 はそれぞれ、問題 1 で $k = 100$ 、問題 2 で $k = 1.0$ 、問題 2 で $k = 100$ という問題の反復回数を表わしている。

メッシュサイズを $n \times n$ 、分割した小領域数を $m \times m$ とすると、実験結果から、メッシュサイズにより、全計算量が小さくなる分割数が異なり、メッシュサイズを大きくすると、効率のよい分割数も大きくなる事が分かる。また、反復回数は $\sqrt{mn}$ にほぼ比例していて、メッシュサイズが小さいと、反復回数が $O(n)$ である他の方法と計算量はあまり変わらないが、大きくすると、計算量は少なくて済むことが分かる。

これから以下のことが分かる。

1. コレスキー分解、CG 法、ICCG(1,1) 法の浮動小数点演算回数は $O(n^3)$ である。

表 1 メッシュサイズによる反復回数と浮動小数点演算数

		Cholesky	DDM			CG	ICCG (1,1)
prob size			2 × 2	4 × 4	8 × 8		
32 × 32	LU(MFlop)	1.15	0.40	0.03	0.00	–	0.01
	CG(iter.)	–	11	24	32	54	36
	total(MFlop)	1.23	1.13	1.33	1.38	1.11	1.19
64 × 64	LU(MFlop)	11.3	5.3	0.4	0.0	–	0.0
	CG(iter.)	–	12	32	45	107	65
	total(MFlop)	11.7	8.7	8.3	9.6	8.8	8.3
128 × 128	LU(MFlop)	103	59	5.8	0.8	–	0.1
	CG(iter.)	–	12	43	59	391	132
	total(MFlop)	106	74	50	57	128	67

表 2 メッシュサイズによる反復回数 - 問題 1、拡散係数 $k=100$

		Cholesky	DDM			CG	ICCG (1,1)
prob size			2 × 2	4 × 4	8 × 8		
32 × 32	CG(iter.)	–	41	63	92	837	55
64 × 64	CG(iter.)	–	57	87	131	1822	104

表 3 メッシュサイズによる反復回数 - 問題 2、拡散係数 $k=1.0$

		Cholesky	DDM			CG	ICCG (1,1)
prob size			2 × 2	4 × 4	8 × 8		
32 × 32	CG(iter.)	–	27	39	55	104	41
64 × 64	CG(iter.)	–	39	56	79	209	78

表 4 メッシュサイズによる反復回数 - 問題 2、拡散係数 $k=100$

		Cholesky	DDM			CG	ICCG (1,1)
prob size			2 × 2	4 × 4	8 × 8		
32 × 32	CG(iter.)	–	42	62	89	815	53
64 × 64	CG(iter.)	–	59	87	127	1812	102

表5 拡散係数の値による反復回数 - 問題1

k	1.0	1.1	10.0	100.0	1000.0
DDM(2×2)	12	43	53	57	61
DDM(4×4)	32	72	81	87	92
DDM(8×8)	45	101	120	131	138
CG	107	267	683	1810	4328
ICCG(1,1)	65	81	94	100	110

表6 拡散係数の値による反復回数 - 問題2

k	1.0	1.1	10.0	100.0	1000.0
DDM(2×2)	39	49	55	59	62
DDM(4×4)	56	72	81	87	91
DDM(8×8)	79	102	119	128	135
CG	209	283	672	1812	4285
ICCG(1,1)	78	83	94	102	107

2. コレスキー分解を用いて capacitance system をつくるのに、浮動小数点演算回数は $O(m^2(\frac{n}{m})^3)$ 必要。

3. capacitance system をブロック対角前処理付共役勾配法を用いて解くのに、反復回数は、 $O(\sqrt{mn})$ 、1 回あたりの浮動小数点演算は $O(n^2)$ 必要。

これは、2 は、各小領域の大きさと小領域数に、3 は capacitance system の大きさに依存しているからである。

以上から、計算量は、 $m = n^{\frac{1}{3}}$ とすることにより、最適の値 $O(n^{\frac{8}{3}})$ が期待でき、これは、コレスキー分解や、CG 法、ICCG 法を単純に用いるよりも少なくてすむ。

5.2 拡散係数の値による比較

表5、表6 はそれぞれ、問題1、問題2 の拡散係数 k の値による、反復回数を示している。

これから、 k の値を10 倍するごとに、ほぼ定数回増えるが、DDM はCG 法ほど拡散係数の影響を受けず、ICCG 法程度であることが分かる。

問題1 において、拡散係数 k の値が1.0 における反復回数が異常に小さいのは、境界の値が各小領域について対称であること、また、4 領域に分割された場合の境界の値がほぼ0 になるからである。

5.3 並列化効率

問題1 で、拡散係数 $k = 1.0$ 、メッシュサイズが 64×64 の問題を、 4×4 の領域に分割して16 台のプロセッサで解いたとき、並列化効率が0.95 を超え、この高い並列性が

あることが分かった。通信が必要ところが、capacitance system を前処理付共役勾配法で解くときのみだが、前処理にはブロック対角部分のコレスキー分解を用いているので、前処理にも通信がいらず、通信は $O(n)$ である。そのときの計算が $O(n^2)$ のため、計算の方が支配的となり、また通信そのものを隠蔽することにより影響を小さくしているからである。

6 まとめ

矩形領域のポアソン方程式を、メッシュサイズ、領域分割数、拡散係数の値を変えて実際に解いて、コレスキー分解、CG 法、ICCG(1,1) 法と比較してみた。その結果、メッシュサイズが $n \times n$ なら他の方法は $O(n^3)$ の計算量を必要とするのに対して、領域分割法は、領域分割数を $n^{\frac{1}{3}} \times n^{\frac{1}{3}}$ とすることにより、計算量は $O(n^{\frac{8}{3}})$ で済むことが分かった。

また、高い並列化効率も得られることが分かった。

今後の課題として、どういう種類の問題が領域分割法に向いていて、どういう種類が向いていないのかを調べたい。またより収束性の良い前処理の発見も課題の一つである。

参考文献

- [1] P.E.Bjorstad and O.B.Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures *SIAM Journal on Numerical Analysis*, 23:1097-1120,1986
- [2] Eric F. Van de Velde. Concurrent Scientific Computing *Springer-Verlag*
- [3] Alan George. Nested Dissection of a Regular Finite Element Mesh *SIAM Journal on Numerical Analysis*, 10:345-363,1973
- [4] D.E.Keyes, Y.Saad, and D.G.Truhlar. Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering. *SIAM, Philadelphia, PA, 1995*
- [5] Yousef Saad Iterative Methods for Sparse Liniar Systems *PWS Publishing Company*