ハイパフォーマンス
コンピューティング 72－6
（1998. 8. 6）

密行列に対する共役勾配法系算法の並列化

横山至治、重原孝臣、三島健稔
埼玉大学工学部情報システム工学科
338–8570 埼玉県浦和市下大久保 255

概要

共役勾配法系算法の一つである共役残差法を密行列を係数にもつ連立一次方程式に適用し、分散メモリ型並列計算機で実行することに関する研究を行った。実行時間に関する簡単なモデルにより、最適なプロセッサ数は問題サイズに比例することを示した。日立 SR2201 上での数値実験により、上記の予測がかなりの範囲で正しいことが分かった。

The conjugate gradient method and its variants for dense linear systems on distributed memory computers.

Muneharu Yokoyama, Takaomi Shigehara, and Taketoshi Mishima
Department of Information and Computer Sciences, Saitama University
Shimo-Okubo 255, Urawa, SAITAMA 338–8570, JAPAN

Abstract

The conjugate residual method, one of the iterative methods for solving linear systems, is applied to the problems with a dense coefficient matrix on distributed memory parallel computers. Based on an assumption on the computation and communication times of the proposed algorithm for parallel computers, it is shown that the optimum number of processing elements is proportional to the problem size N. The validity of the prediction is confirmed through numerical experiments on Hitachi SR2201.

1 はじめに

この論文では、連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を解くことを考える。ここで、A は $N \times N$ の正則な係数行列、$\boldsymbol{x}$ と $\boldsymbol{b}$ は長さ N の列ベクトルである。連立一次方程式の数値解法は、Gauss の消去法などの直接法と、SOR 法や共役勾配法などの反復法の二つに大きく分類できる。一般的な直接法では、密な係数行列を持つ問題に対しては演算量が $O(N^3)$ であり、問題サイズが大きくなると高速な計算機においても非常に長い実行時間が必要となる。一方、最終的な結果に対して必要となる精度は、高々2 ～ 3 桁程度で充分である場合が多い。従って、大規模な問題に対しては、ある程度精度を犠牲にする代わりに計算時間が少くてすむような解法が望ましいと考えられる。ここでは、反復法の一種である共役勾配法の系統がこの点において好ましい性質を持っていることを示し、さらに分散メモリ型並列計算機上での効率的な実装について研究した結果を示す。

分散メモリ型並列計算機を効率的に使うためには、計算と通信のそれぞれの負荷のバランスを考えることが重要である。計算アルゴリズムによっては、必ずしもプロセッサ数を可能な限り最大に取れば実行時間が最小になるとは限らない。この場合、問題サイズに応じて最適なプロセッサ数を設定する必要がある。共役勾配法系算法に対しては、小さな問題について予備的な計算を行なった結果を使って、より大きな問題に対する最適なプロセッサ数を予め知ることができることを示す。

2 共役勾配法および共役残差法

共役勾配法 (conjugate gradient method[1]) は、正定値対称行列を係数とする問題に適用できる。その計算手順は以下の通り。

初期ベクトル $\boldsymbol{x}_0$ をとる;
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$; $\boldsymbol{p}_0 := \boldsymbol{r}_0$;
for $k := 0, 1, \cdots$ **until** $||\boldsymbol{r}_k|| \leq \epsilon||\boldsymbol{b}||$ **do**
begin
  $\alpha_k := (\boldsymbol{r}_k, \boldsymbol{p}_k)/(\boldsymbol{p}_k, A\boldsymbol{p}_k)$;
  $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$;
  $\boldsymbol{r}_{k+1} := \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$;
  $\beta_k := -(\boldsymbol{r}_{k+1}, A\boldsymbol{p}_k)/(\boldsymbol{p}_k, A\boldsymbol{p}_k)$;
  $\boldsymbol{p}_{k+1} := \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$;
end

ここでは $\epsilon > 0$ を適当に定めて、k 回目の反復で得られる近似解 $\boldsymbol{x}_k$ に対する残差 $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$ に関して $||\boldsymbol{r}_k|| \leq \epsilon||\boldsymbol{b}||$ を収束判定条件とする。共役勾配法では $\boldsymbol{x}_k$ が反復毎に真の解 $\boldsymbol{x}$ に近付くことが知られているが、残差 $\boldsymbol{r}_k$ は激しく振動する [2]。

一方、共役残差法 (conjugate residual method、文献によっては Orthomin(2) などと呼ばれることもある [2]) は共役勾配法に似た反復法であるが、非対称な係数行列を持つ問題に対して適用できるという特長を持つ。共役残差法の計算手順は以下の通り。

初期ベクトル $\boldsymbol{x}_0$ をとる;
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$; $\boldsymbol{p}_0 := \boldsymbol{r}_0$; $\boldsymbol{q} := A\boldsymbol{p}_0$;
for $k := 0, 1, \cdots$ **until** $||\boldsymbol{r}_k|| \leq \epsilon||\boldsymbol{b}||$ **do**
begin
  $\mu := (\boldsymbol{q}, \boldsymbol{q})$;
  $\alpha_k := (\boldsymbol{r}_k, \boldsymbol{q})/\mu$;
  $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$;
  $\boldsymbol{r}_{k+1} := \boldsymbol{r}_k - \alpha_k \boldsymbol{q}$;
  $\boldsymbol{a} := A\boldsymbol{r}_{k+1}$;
  $\beta_k := -(\boldsymbol{a}, \boldsymbol{q})/\mu$;
  $\boldsymbol{p}_{k+1} := \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$;
  $\boldsymbol{q} := \boldsymbol{a} + \beta_k \boldsymbol{q}$;
end

係数行列の対称部分 $M = (A + A^T)/2$ が定値であれば、共役残差法における残差 $\boldsymbol{r}_k$ は単調に減少する。この利点から、本研究では共役残差法を採用することにする。なお、各反復における主要な演算は両解法ともに行列・ベクトルの積が 1 回であり、反復 1 回あたりの演算量はほぼ同じである。

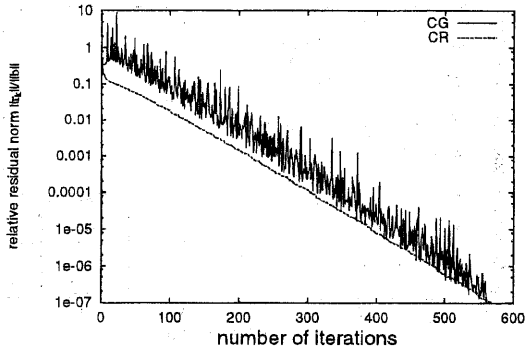

図 1: 共役勾配法 (CG) と共役残差法 (CR) の収束特性

共役勾配法と共役残差法の収束特性を図 1 に示す。係数行列には Frank 行列

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 1 \\ 1 & 2 & 2 & \cdots & 2 & 2 & 2 \\ 1 & 2 & 3 & \cdots & 3 & 3 & 3 \\ \vdots & \vdots & \vdots & & & & \\ 1 & 2 & 3 & \cdots & N-2 & N-2 & N-2 \\ 1 & 2 & 3 & \cdots & N-2 & N-1 & N-1 \\ 1 & 2 & 3 & \cdots & N-2 & N-1 & N \end{pmatrix}$$

を用いた。Frank 行列は、その逆行列や固有値が解析的に分かっており、連立一次方程式や固有値問題のプログラムのベンチマークによく使われる。問題の大きさは $N = 1024$ とした。相対的残差ノルム $||\boldsymbol{r}_k||/||\boldsymbol{b}||$ が 10^{-7} に達するまでの反復回数は、ほぼ同じく約 560 回であり、どちらの場合でも残差ノルムは概ね指数関数的に減少する。ただし、共役勾配法では激しく振動しながら収束に向かうのに対し、共役残差法では残差が単調に減少しながら収束している。共役残差法の収束特性に注目すると、相対的残差ノルム $||\boldsymbol{r}_k||/||\boldsymbol{b}||$ が 10^{-3} に到達するまでに必要な反復回数が $N/3$ 以下であることが分かる。$N/3$ 回の反復で共役残差法の演算量はほぼ $2N^3/3+O(N^2)$ となり、これは LU 分解を用いた直接解法の演算量に相当する。

N	1024	2048	4096	8192	16384
反復回数	558	1068	2073	4077	7970

表 1: 共役残差法の収束までの反復回数

共役残差法を用い、上記と全く同じ条件で問題の大きさだけを変えた時の相対残差ノルムが一定値 $\epsilon = 10^{-7}$ までの反復回数を表 1 に示した。どの場合においても $N/2$ 回程度の反復で収束しており、Frank 行列に対しては N が大きい場合でも上記の演算量に関する議論が成り立つことが分かる。

3 並列化アルゴリズムおよび性能評価

ここでは、共役残差法の並列化アルゴリズムを提示し、分散メモリ型並列計算機上での数値実験の結果を示す。

問題サイズ N が充分に大きい場合、行列・ベクトル積が演算量の大部分を占める。従って、並列化はこの部分だけに対して行い、ベクトル同士の和や内積などの他の部分については全てのプロセッサにおいて同一の計算を行なう。行列 A の内容は計算全体を通じて不変なので、等しい行数 N/P に分割して各プロセッサに分割する (簡単のため、ここでは N はプロセッサ数 P で割り切れるものとする)。A の分割の様子を図 2 に示した。各プロセッサは N/P 行 N 列の行列を持ち、行列・ベクトル積が必要な時は、ベクトル (N 行) を各々の行列に掛け、答のベクトル (N/P 行) を得る。これらのベクトルは 1 つのプロセッサに収集 (Gather) され、N 行のベクトルとして各プロセッサに放送 (Broadcast) される。数値実験では通信ライブラリとして MPI[3] を用いているが、この場合上記の通信は関数 `MPI_Allgatherv()`[1]への

[1]ここでは、N が P で割り切れない場合にも対応できるように、各プロセッサが持つベクトルの長さが異なるタイプのライブラリ関数を用いた。

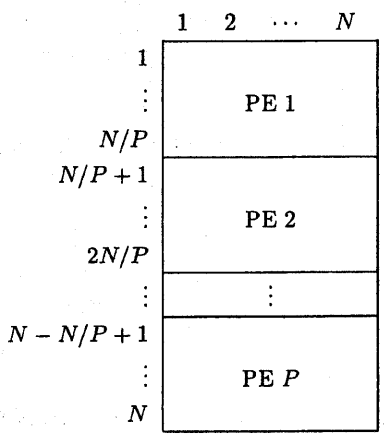

図 2: 行列 A の行ブロック分割

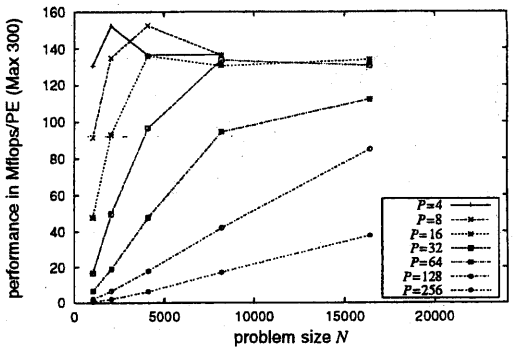

図 3: 問題サイズを変化させた場合のプロセッサ 1 台あたりの性能

呼び出し 1 回で実現される。各プロセッサ上での演算にはレベル 1 およびレベル 2 の BLAS[4, 5] のサブルーチン群を用いている。このことにより、プログラムの移植性を高めると同時に、特定のアーキテクチャに対して最適化されたライブラリの使用を可能にしている。

東大大型計算機センターの日立 SR2201 上で数値実験を行った。SR2201 のプロセッサ単体の理論ピーク性能は 300Mflops である。図 3 に問題サイズ N を変化させた時のプロセッサ 1 台当たりの性能を示した。プロセッサ数 P を様々な値に固定した結果が示してある。ここでも、Frank 行列を係数にとった。また、性能の算出には行列・ベクトル積の部分のみを計上し、その他の演算は無視している。いずれのプロセッサ数においても、問題サイズが大きくなるにつれ性能が向上し、一定値 (約 140Mflops/PE) に近付く傾向にある事が分かる。飽和性能の 140Mflops/PE は、ピーク性能の 300Mflops/PE の約 1/2 にあたる。これはベクトル長を十分に大きく取った時のレベル 2 BLAS の性能がそのまま出ているものである。

ここで、現アルゴリズムにおいて実行時間を最小とするプロセッサ数について考える。そのために、共役残差法の反復 1 回当たりの実行時間 T が、以下のように問題サイズ N とプロセッサ数 P で表わせると仮定する。

$$T = \alpha_1 \frac{N^2}{P} + \alpha_2 P + const. \qquad (1)$$

式 (1) の第 1 項は各プロセッサで行列・ベクトル積を計算するために必要な時間である。前述のように $N \times N$ の行列 A が P 個のプロセッサに均等に配分されるので、この時間は N^2 に比例し、P に反比例する。第 2 項は通信時間の内 P に依存する部分で、ここでは P に比例するものと仮定した。最後の項 $const$ は実行時間の内で P に依存しない部分である。α_1 および α_2 は N および P とは独立で、計算機の性能および通信ライブラリの実装により決まる定数である。

以上の仮定の下で、一定の N に対して T を最小とする P は、

$$P_0 = \sqrt{\frac{\alpha_1}{\alpha_2}} N \qquad (2)$$

と書ける。つまり、最適なプロセッサ数は問題サイズに比例する。反復 1 回あたりの実行時間 T は計算を通じて一定であり、かつ係数行列 A およびベクトル $\boldsymbol{b}$ の値に依存しない。従って、P_0 はある決まったサイズの問題に対して実行時間を最小とする最適なプロセッサ数であると言うことができる。ある新し

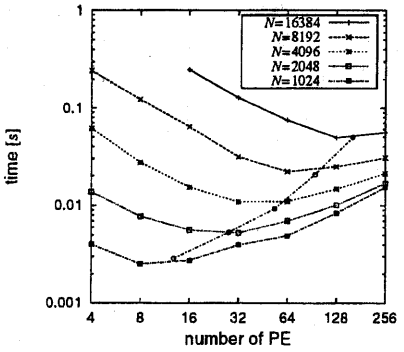

図 4: PE 台数を変化させた場合の反復 1 回当たりの実行時間。鎖線は実行時間を最小となる点を結んだもの

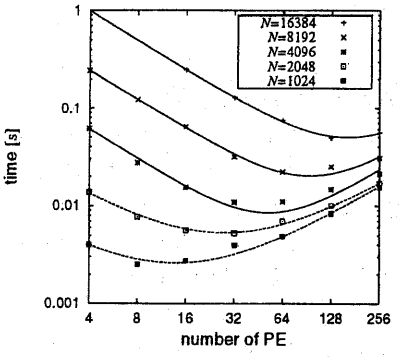

図 5: PE 台数を変化させた場合の反復 1 回当たりの実行時間。各曲線については本文参照

い並列計算機に共役残差法を実装する場合、小さな N に対する計算をいくつか実行して P_0 を決定すれば、式 (2) の比例係数を知ることができる。その後は、計算機や通信ライブラリに関する詳細を知らなくても、任意の大きさの問題に対して最適なプロセッサ数 P_0 を予め知ることができる。

図 4 にプロセッサ台数を変化させた場合の反復 1 回あたりの実行時間を問題サイズ毎に表示してある。実験を行なった範囲では $P_0 = N/80 \sim 100$ が成り立ち、上記のモデルの正当性を裏付けている。

しかし、図 4 の鎖線を注意深く見ると、$N = 4096$ から傾きが大きくなっていることが分かる。実際、各々の N の実行時間に対して式 (1) を仮定した最小 2 乗近似を行うと、$N = 4096$ 以上では N が大きくなるにつれ α_2 が直線的に増大していることが分かった。そこで、$N = 16384$ に対して

$$T = \alpha_1 \frac{N^2}{P} + (\alpha_2 + \alpha_3 N)P + const. \qquad (3)$$

を仮定した最小 2 乗近似を行い、得られた係数 $\alpha_1, \alpha_2, \alpha_3$ を使って $N = 4096, 8192, 16384$ に対する式 (3) をプロットしてみると図 5 のようになった。$N = 1024, 2048$ については、$N = 2048$ に対して式 (1) を仮定した最小 2 乗近似を行い、得られた係数 α_1, α_2 を使って式 (1) をプロットしてある。各曲線は、実験値をよく再現している。つまり、実行時間の N および P への依存性は、$N = 2048$ と $N = 4096$ の間を境に式 (1) から式 (3) へと変化していると結論することができる。従って、小さい問題サイズにおいて数値実験を行なっただけで大きい問題サイズでの最適なプロセッサ数を決めることができるという主張は厳密には成り立たない。しかし、現実の環境では一つのジョブに割り当て可能なプロセッサ数は 2^n(n は整数) となっているのが通常なので、与えられた問題サイズに対して実行時間が最小となる PE 数 を求めるためには、粗い計算で充分である。実際、今回実験を行なった範囲では、$N = 1024$ に対する計算で得られた結果を使って、$N = 16384$ までの全ての問題サイズに対して最適なプロセッサ数を予測できており、式 (1) の仮定を大きな問題サイズに適用しても実用的には問題ないと言える。

4 まとめ

連立一次方程式に対する共役勾配法系算法の一つである共役残差法を、大規模な密行列を係数に持つ方程式に適用し、分散メモリ型並列計算機上で実行することに関する研究を行った。共役残差法は共役勾配法と比較して収束特性が良く、適用範囲も広い。高い精度の解を必要としない場合には、共役残差法は一般的な直接法よりも短時間で実行することができる。

共役残差法の並列化アルゴリズムを提示し、その実行時間について簡単なモデルを仮定した結果、実行時間を最小とするプロセッサ数は、問題サイズに比例することが分かった。SR2201上での数値実験により、このモデルの正当性を示した。数値実験の結果を詳細に見ると、問題サイズが大きい場合にこのモデルが必ずしも成り立たないことが判明したが、実用的な範囲では小さな問題サイズの数値実験の結果により、任意の問題サイズで実行時間を最小にするプロセッサ数を推定することができる。

参考文献

[1] M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *Journal of Research of the National Bureau of Standards,* Vol. 49 (1952), pp. 409–436.

[2] A. Greenbaum, "Iterative Methods for Solving Linear Systems", pp.33–37, SIAM, Philadelphia, 1997.

[3] Message Passing Interface Forum, "MPI: A message-passing interface standard", Special issue on MPI, International Journal of Supercomputer Applications, vol.8, no.3/4, 1994.

[4] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, "Basic Linear Algebra Subprograms for Fortran Usage", ACM Trans. Math. Soft., vol.5, pp.308–325, 1979.

[5] J.J. Dongarra, J. DuCroz, S. Hammarling, and R. Hanson, "An Extended Set of Fortran Basic Linear Algebra Subprograms", ACM Trans. Math. Soft., vol.1, pp.1–32, 1988.