ハイパフォーマンス
コンピューティング 62－22
（1996. 8. 29）

並列数値計算ライブラリの開発

清水 大志、†佐々木 誠、市原 潔、岸田 則生、鈴木 惣一朗、
佐藤 滋、田中 靖久、横川 三津夫、蕪木 英雄

日本原子力研究所 計算科学技術推進センター
shimizu@koma.jaeri.go.jp
†日本総合研究所

近年、演算回路素子等の速度向上が限界となりつつあるために、並列計算は大規模数値シミュレーションの分野において重要な手法となっており、効率及び移植性の良い並列ライブラリが必要とされている。そこで、我々は各種計算機に対応可能であるMPIまたはPVMを用いた分散メモリ型ベクトル並列計算機用数値計算ライブラリの開発を行っている。本論ではHouseholder変換による三重対角化及び2分法を用いた実密対称行列の固有値問題解法ルーチンの開発について報告する。行列のデータは列方向サイクリック方式により分割し、プロセッサ間のデータ転送量を減らすため対称行列の全ての成分を格納する。Householder変換について8プロセッサを使用した並列化による速度向上率は、2000×2000行列に対してParagonで6.0倍である。VPP300では4000×4000行列に対して4.2倍の値を得た。

Development of Parallel Libraries

Futoshi Shimizu, †Makoto Sasaki, Kiyoshi Ichihara, Norio Kishida, Soichiro Suzuki,
Shigeru Sato, Yasuhisa Tanaka, Mitsuo Yokokawa, Hideo Kaburaki

Center for Promotion of Computational Science and Engineering,
Japan Atomic Energy Research Institute
shimizu@koma.jaeri.go.jp
†The Japan Research Institute

In recent years, since the development of fast processor element has reached the upper limit, parallelism is one of the solutions for massive numerical simulations, and efficient and portable parallel libraries are needed. With MPI or PVM which is not limited to a certain architecture, we are developing parallel subroutine libraries specific for use on parallel vector processors. We report here the development of parallel subroutine library for eigenvalue problem of real symmetric matrix based on the Householder transformation and the bisection method. The matrix is partitioned into columns by the column-wise cyclic decomposition scheme, and all elements of the symmetric matrix are stored in order to reduce data exchanges among processors. For the Householder transformation using eight processors on Paragon, the speedup ratio of 6.0 has been achieved for a matrix of 2000 × 2000 elements. In the case of the matrix of 4000 × 4000 elements, the ratio is found to be 4.2 on VPP300.

1 はじめに

近年、演算回路素子等の大幅な速度向上は望めないことから、多数のプロセッサを並行して稼働させる分散メモリ型並列計算機が注目されている。しかし、並列計算機ではプロセッサ間のデータ転送の記述等、逐次処理計算機では見られなかったプログラム作成の難しさがあり、アプリケーション分野の研究者による利用を困難にしている。

逐次処理計算機の利用においては、固有値計算等プログラミングが複雑な数値計算に対して、データの受け渡し等のインタフェイスが定められた数値計算ライブラリは非常に重要な役割を果たしてきた。これらはサブルーチン単位でデータ記述が閉じているため、ライブラリへのデータ受け渡しが容易に出来る。しかし、分散メモリ型並列計算機においては、データの各プロセッサへのマッピングがサブルーチンに渡って記述されるため、数値計算ライブラリの作成も複雑である。

並列計算機に対しては、Dongarra 達のグループによる密行列やベクトルの四則演算の並列化ルーチンの整備がほぼ完了した [1]。また、科学技術計算においてよく現れる帯行列や疎行列に関するライブラリは開発が進行中である。これらはスカラ並列計算機を開発対象としている。一方、ベクトル並列計算機は差分法のような連続したデータを扱う計算に対して有効であり、ベクトル並列計算機向きのライブラリ開発も重要と考えられる。ベクトル計算機用のライブラリは、ある種の数値計算法に対してベンダ提供のものが一部存在するが、その他のライブラリと呼び出しインタフェイスやデータ構造が異なっているため、一般的でない。

そこで、我々は各種計算機に対応可能である標準化メッセージパッシングライブラリ MPI (Message Passing Interface)[2] または PVM (Parallel Virtual Machine)[3] を用いたベクトル並列計算機用数値計算ライブラリの構築を目指している。現在、大次元疎行列を係数行列に持つ連立一次方程式解法ルーチンや、実密対称行列の固有値問題解法ルーチン、高速フーリエ変換ルーチン、計算精度評価ライブラリ等の開発を行っている。

本論では実密対称行列の固有値問題解法ルーチンの開発について報告する。

2 実密対称行列の固有値問題解法ルーチン

2.1 解法

実対称行列の固有値と固有ベクトルを求めるための手法には Jacobi 法、QR 法、Lanczos 法、Householder 変換+2 分法などがある [4, 5, 6]。今回は並列プロセッサ間のデータ転送の形態の単純さから、Householder 変換による三重対角化を用いた方法を並列化することにした。

2.1.1 Householder 変換による三重対角化

n 次元実対称行列 $A(a_{ij})$ に対して $A^{(0)} = A$ として以下のような相似変換をおこなう。

$$A^{(k)} = P_k A^{(k-1)} P_k \qquad (k = 1, 2, ..., n-2) \tag{1}$$

ここで P_k は次のように定義される対称な直交行列である。

$$P_k = I - \frac{\boldsymbol{u}_k \boldsymbol{u}_k^T}{h_k} \tag{2}$$

$$u_{ik} = 0 \qquad (i = 1, ..., k)$$

$$u_{k+1,k} = a_{k+1,k}^{(k-1)} + \sigma$$

$$u_{ik} = a_{ik}^{(k-1)} (i = k+2, ..., n)$$

$$\sigma = sign(a_{k+1,k}^{(k-1)}) \sqrt{\sum_{i=k+1}^{n} (a_{ik}^{(k-1)})^2}$$

$$h_k = \|\boldsymbol{u}_k\|^2/2$$

この変換により三重対角行列 $T \equiv A^{(n-2)}$ を得る。

2.1.2 二分法による三重対角行列の固有値の計算

三重対角行列 T の固有値の計算は 2 分法により行う。T の成分を、

$$T_{ii} = \alpha_i, T_{i+1,i} = T_{i,i+1} = \beta_i$$

として、行列 $T - \lambda I$ の列 k、行 k までの部分行列に対する小行列式 $f_k(\lambda)$ を以下の漸化式により計算する。

$$f_0(\lambda) = 1$$

$$f_1(\lambda) = \alpha_1 - \lambda$$

$$f_k(\lambda) = (\alpha_k - \lambda)f_{k-1}(\lambda) - \beta_{k-1}^2 f_{k-2}(\lambda)$$

このとき、$f_0(\lambda), f_1(\lambda), ..., f_n(\lambda)$ の隣り合う数値の符号が一致した回数 $N(\lambda)$ が λ より大きい固有値の個数を与える。

大きい方から k 番目の固有値 λ_k は、

$$N(a) \geq k, N(b) < k$$

となる $a, b(a < b)$ の中間の値 $c = (a + b)/2$ について $N(c)$ を計算し、

$$N(c) \geq k \Longrightarrow a = c$$
$$N(c) < k \Longrightarrow b = c$$

という処理を繰り返すことにより必要な精度で λ_k を求められる。実際の計算では $f_k(\lambda)$ の代わりに

$$g_k(\lambda) = f_k(\lambda)/f_{k-1}(\lambda)$$

を求め、負でない $g_k(\lambda)$ の個数を $N(\lambda)$ とする。このとき、$g_k(\lambda)$ の満たす漸化式は次のようになる。

$$g_1(\lambda) = \alpha_1 - \lambda$$

$$g_k(\lambda) = \alpha_k - \lambda - \beta_{k-1}^2/g_{k-1}(\lambda)$$

2.1.3 逆反復法による三重対角行列の固有ベクトルの計算及び、もとの行列の固有ベクトルへの逆変換

対称な三重対角行列の固有ベクトルは逆反復法により求める。行列 T の固有値 λ に対し、任意の初期ベクトル $\boldsymbol{y}_0$ から始めて式

$$(T - \lambda I)\boldsymbol{y}_k = \boldsymbol{y}_{k-1} \qquad (3)$$

で計算される $\boldsymbol{y}_k$ は行列 T の固有ベクトルに収束する。ただし縮退している固有値の固有ベクトルに対しては、2 番目以降の縮退固有値について $\boldsymbol{y}_0$ の成分を乱数であたえる。

固有値問題

$$A\boldsymbol{x} = \lambda\boldsymbol{x}$$

の固有ベクトルへの逆変換は、$\boldsymbol{y}_{n-2} = \boldsymbol{y}$ からはじめて、

$$\boldsymbol{y}_{k-1} = P_k\boldsymbol{y}_k = \boldsymbol{y}_k - \boldsymbol{u}_k(\boldsymbol{u}_k^T\boldsymbol{y}_k)/h_k$$

を $k = n-2, ..., 2, 1$ に対して計算し、$\boldsymbol{x} = \boldsymbol{y}_0$ とする。

2.2 並列化の方法

2.2.1 プロセッサへのデータ分割

密対称行列のデータは図 1に示すような列方向サイクリック方式により、プロセッサに分割する。この方法の利点は、

1. 単純な分割法であるために並列処理を行い易い。
2. 処理を行う範囲が行列上の限られた部分になっても休眠状態になるプロセッサの数が少ない。

等である。後者は Householder 変換で処理が行われる範囲が 1 行/1 列ずつ狭まっていくことを考えると、負荷分散の観点から重要な性質である。

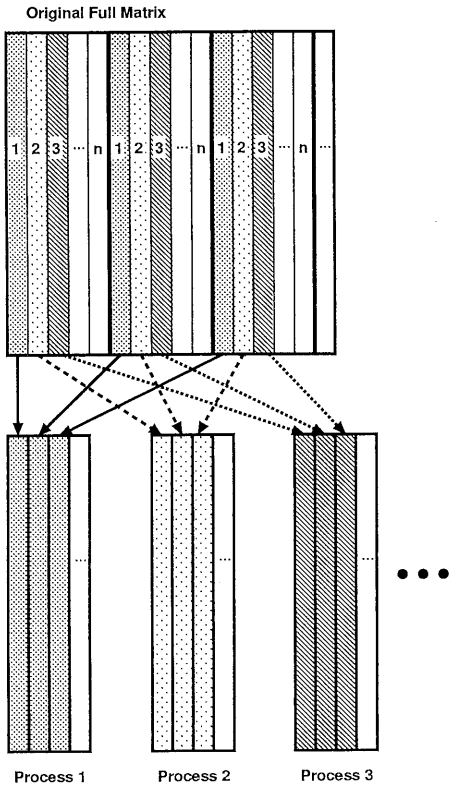

図 1: 列方向サイクリック方式による
データ分割

また、Householder 変換による三重対角化処理の並列化においてプロセッサ間のデータ転送量を減らすため、対称行列のすべての成分を格納する。

2.2.2 三重対角化処理の並列化

Householder 変換の各段の処理は前段の結果に依存するため、複数の段の処理を並列に行うことが出来ない。このため並列処理は各段で各々のプロセッサに分散された列ベクトルに対して行う。

手続き

$$\boldsymbol{x} \equiv (0, \ldots, 0, a_{k+1,k}^{(k-1)}, a_{k+2,k}^{(k-1)}, \ldots, a_{n,k}^{(k-1)})^T$$

$$\sigma = sign(a_{k+1,k}^{(k-1)})\sqrt{\|\boldsymbol{x}\|^2}$$

$$\boldsymbol{u}_k = \boldsymbol{x} + \sigma \boldsymbol{e}_{k+1}$$

$$h_k = \sigma(a_{k+1,k}^{(k-1)} + \sigma) = \sigma u_{k+1,k}$$

$$A^{(k)} = P_k A^{(k-1)} P_k, \qquad P_k = I - \frac{\boldsymbol{u}_k \boldsymbol{u}_k^T}{h_k}$$

の並列処理は以下のような手順で行った。

0. k 列を持つプロセッサが σ を計算し $\boldsymbol{u}_k$ を全プロセッサに送信する。
1. $\boxed{\boldsymbol{p} = A^{(k-1)}\boldsymbol{u}_k/h_k}$ (並列計算)
2. $\boldsymbol{p}$ をひとつのプロセッサに集めてから全プロセッサに送信。
3. $s = \boldsymbol{u}_k^T \boldsymbol{p}$
4. $\boldsymbol{q} = \boldsymbol{p} - \frac{s}{2h_k}\boldsymbol{u}_k$
5. $\boxed{A^{(k)} = A^{(k-1)} - (\boldsymbol{u}_k \boldsymbol{q}^T + \boldsymbol{q}\boldsymbol{u}_k^T)}$ (並列計算)

手順 0 及び 2 でプロセッサ間のデータの転送が発生し、手順 1 及び 5 で異なるデータに対する処理が並列に行われる。手順 3 及び 4 は各プロセッサで同じものを計算する。

手順 1 でのベクトル $\boldsymbol{p}$ の計算は、対称性を利用して $A^{(k-1)}$ の行ベクトルと列ベクトル $\boldsymbol{u}_k$ の内積計算を各プロセッサが持つ列ベクトルと $\boldsymbol{u}_k$ の内積計算に置きかえて実行する。

$$\sum_{j=k+1}^{n} a_{ij}^{(k-1)} u_j = \sum_{j=k+1}^{n} a_{ji}^{(k-1)} u_j$$

ただし、各段の Householder 変換後に $A^{(k)}$ が全要素 (正確には $k+1$ 行、$k+1$ 列以降) で対称になっていることを保証するために、手順 5 の行列の更新処理を $i \geq j$ だけでなく $i < j$ に対しても行う必要がある。

2.2.3 三重対角行列の固有値、固有ベクトルの計算及び、もとの行列の固有ベクトルへの逆変換の並列化

計算すべき固有値/固有ベクトルを各プロセッサに振り分け、各々のプロセッサでは2分法+逆反復法により計算を行う。また、逆変換についても、

$$\boldsymbol{x}_m = P_1 P_2 \ldots P_{n-2} \boldsymbol{y}_m$$

を各プロセッサに分散して並列に計算する。

2.3 性能評価

並列処理性能評価のために、1000×1000、2000×2000 及び 4000×4000 の実対称行列に対する Householder 変換を、PVM が実装されている Paragon 及び VPP300 において実行した。表 1に変換の計算時間及び速度向上率 (1プロセッサの場合との経過時間の比の逆数) を示す。

Paragonによる実行において、1000×1000の行列に対して8プロセッサを使用した場合の速度向上率はおよそ3倍である。2000×2000の行列に対しては1プロセッサのメモリにデータが納まらず、パフォーマンスが著しく低下しているため、2プロセッサの場合を基準に速度向上率を求めた。このとき、8プロセッサにおける速度向上率は6.0倍であり、大規模な問題ほど並列化の効果が現れている。一方、VPP300のスカラモードではプロセッサ間のデータ転送によるオーバーヘッドがほとんど見られない。

VPP300 のベクトルモードによる実行では、スカラモードと比べて 1/12 ないし 1/45 の計算時間となっている。2000×2000 の行列に対しては並列化の効果が十分には得られていないが、4000×4000 の行列に対する速度向上率は 8 プロセッサで 4.2 倍である。

表 1: Householder 変換の計算時間 (秒) と速度向上率

Paragon

プロセッサ数	1000×1000	2000×2000
1	203 (1.0)	
2	139 (1.5)	1070 (2.0)
4	81 (2.5)	598 (3.6)
8	61 (3.3)	355 (6.0)

VPP300(スカラモード)

プロセッサ数	2000×2000	4000×4000
1	505.67 (1.0)	4850.82 (1.0)
2	256.14 (2.0)	2441.65 (2.0)
4	125.96 (4.0)	1244.50 (3.9)
8	66.40 (7.6)	622.50 (7.8)

VPP300(ベクトルモード)

プロセッサ数	2000×2000	4000×4000
1	11.17 (1.0)	115.97 (1.0)
2	8.54 (1.3)	74.07 (1.6)
4	6.42 (1.7)	39.95 (2.9)
8	5.46 (2.1)	27.57 (4.2)

3 まとめ

大規模な計算科学の諸問題を現在の並列計算機で効率的に解決する上で、数値計算ライブラリの構築は並列計算法の普及に重要な役割を果すと考えられる。今後、さらに幅広い計算に対応出来るようライブラリの充実を図っていく予定である。

本ライブラリの開発にあたり大変有益なアドバイスを頂戴した並列計算ライブラリ開発専門部会の方々に深く感謝の意を表します。

参考文献

[1] J.Choi, J.Demmel, I.Dhillon, J.Dongarra et al.: *ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers – Design Issues and Performance*, LAPACK Working Note #95, Department of Computer Science, University of Tennesee, Knoxville, Tennesee

[2] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, University of Tenessee, Knoxville, Tennessee,1994

[3] A.Geist, A.Beguelin, J.Dongarra et al.: *PVM3 User's Guide and Reference Manual*, ORNL/TM-12187,1994

[4] 森正武, 名取亮, 鳥居達生: 「岩波講座 情報科学-18 数値計算」, 岩波書店,1982

[5] 村田達郎, 小国力, 唐木幸比古: 「スーパーコンピューター – 科学技術計算への適用」, 丸善,1989

[6] 渡部力, 名取亮, 小国力: 「Fortran77 による数値計算ソフトウェア」, 丸善,1989