並列化ICCGソルバとその公開について

岩下 武史, (京都大学大型計算機センター)

平成13年1月26日

1 はじめに

近年、高性能コンピューティングの分野では、並列計算機の利用が一般的になっている。京大大型計算機センターでも、VPP-800、GP-7000といった並列計算機を計算サーバとして運用している。しかし、一般のユーザーにとって、並列計算機上で高い並列台数効果を得ることは容易なことではない。これは、高い台数効果を得るためには、VPP-Fortran等の並列型言語や、MPIライブラリなどの通信ライブラリを使用した並列プログラミングが必要となるからである。そこで、逐次型プログラムを変更することなく並列計算機上で高い台数効果を得たいというユーザーが少なからずいると考えられる。このような要請に対しては、本来自動並列化コンパイラによって答えるべきであろうが、現状の自動並列化コンパイラでは、ユーザーが扱っている問題の性質やコーディングによっては十分な台数効果を得ることは難しい。自動並列化コンパイラの性能向上を待つ一方で、別のアプローチも行われている。それらの一つに並列化線形ライブラリの利用がある。これは、数値計算で頻繁に行われる行列計算に対して並列化したライブラリをユーザーに提供し、ユーザーが簡単に並列化された行列演算を行えるようにしたものである。有名な並列化線形ライブラリにはScaLapackがあげられるが、京大センターでは、これをVPP用にチューニングしたものをセンターホームページ (http://www.kudpc.kyoto-u.ac.jp/HPC-WG/) において公開している [1]。本稿では、このようなユーザーに負担をかけない並列化手法を支援するものの一つとして、並列化ICCGソルバを公開する。

ICCGソルバは、正値対称な係数行列を持つ連立一次方程式の反復法ソルバとして最も一般的なソルバで、有限要素解析などに広く用いられる。本稿では、バンド行列を係数行列とする連立一次方程式を対象とした並列化ICCGソルバを提供する。ICCGソルバは、共役勾配 (CG) ソルバに不完全コレスキー分解 (Incomplete Cholesky) 前処理を施したものである。一般に、CGソルバはその並列化が容易であるが、IC分解と前処理に伴う前進・後退代入計算はその並列化が困難である。本稿では、ICCGソルバの並列化手法として、ブロック化による手法 [3] と著者らが提案しているリナンバリング処理による手法 [4] を用いる。

2 提供するソルバの利用法と注意点

2.1 全般的な注意事項

今回ソルバを公開するにあたって留意しているのは、ユーザーが既存の逐次型ソルバを本並列化ソルバに変更するのみで並列台数効果が得られるという点である。従って、本ソルバはメモリ分散に対応しておらず、また、係数行列の格納方式には、疎行列の格納方式として最も一般的なCRS形式を用いる。プログラムはFORTRAN+MPIライブラリで書かれており、なるべく特別なチューニングをせず、一般に知られたICCGアルゴリズムに沿って書かれている。従って、本センター外の並列計算機やPCクラスタでも利用することができ、計算環境に応じたチューニングが行うことができる。公開は当面、e-mailによる問い合わせによるものとし、VPP-800向けのチューニング版やメモリ分散版への要請もあわせて受け付ける。ひととおり、ユーザーの意見を聞いた上で、WEB(http://www.kudpc.kyoto-u.ac.jp/HPC-WG/) 上への公開を考えたい。

本稿では、以下の二種類のソルバをサブルーチン

として提供する。

1. BICCG: ブロック ICCG 法 [3]
2. PICCGRP: リナンバリング処理付き並列化 ICCG 法 [4]

これらの二つのソルバは、係数行列がバンド行列であることを仮定しているため、バンド幅が連立一次方程式の次元数に比べて十分に小さいことが必要である（10 分の 1 程度以下）。ソルバ 1、2 を比較した場合、2 の方が性能面で上回る場合が多いと考えられるが、ソルバ 1 の方が適用範囲は広い。実際、ソルバ 1 はバンド行列でなくとも動作する（収束性を考えなければ）。このため、ユーザーにはまずソルバ 1 を使用し、ソルバ 1 が動作したらソルバ 2 を試すという方法をお勧めする。また、使用するプロセッサ数についても、問題のサイズに合わした値を用いる必要がある。2、4 台から初めて、次第に増やしてテストするのがよいと思われる。本センターの実行では、ジョブクラス g の最大プロセッサ数である 10CPU が一つの目安となる。それ以上のプロセッサ数を使用する場合は、計算時間、反復回数などに注意が必要である。おおまかな見積もりとしては、最大使用プロセッサ数は (次元数／バンド幅) で与えられる。

2.2 CRS 形式での係数行列の格納

本節では、本ソルバが係数行列の格納方式として用いている CRS 形式について述べる。ユーザーは解きたい連立一次方程式を CRS 形式で格納しなければならない。CRS 形式では、行列の非ゼロ要素のみを格納し、格納用配列として、3 つの一次元配列を用いる。そのうちの 2 つは整数型で、1 つは (倍精度) 浮動小数点型である。例えば、以下のようになる。

$$A = \begin{pmatrix} 10.5 & 0 & 0 & 2 \\ -1 & 2 & 4 & 0 \\ 2 & 0 & 0 & 3 \\ 0 & 0 & -4 & -5 \end{pmatrix} \qquad (1)$$

まず、浮動少数点型の配列 val には係数行列Aの非ゼロ要素を行方向に順々に見ていく形で格納する。次に、整数型の配列 colind には、配列 val に対応するように、その非ゼロ要素の列番号を格納する。さらに、整数型の配列 rowptr(i) には配列 val 及び colind の i 行の開始位置を格納する。便宜的に rowptr(n + 1) = $nnonzero + 1$ とする。ここで、n は係数行列の次元数で、$nnonzero$ は係数行列中の非ゼロ要素の数である。このように格納することで、係数行列の非ゼロ要素は以下のように取り出すことができる。

val	10.5	2	-1	2	4	2	3	-4	-5
colind	1		4	1	2	3	1	4	4	5

rowptr	1	3	6	8	10

```
do i=1,n
  do j=rowptr(i),rowptr(i+1)-1
   i 行目 colind(j) 列は非ゼロで、値は val(j)
  enddo
enddo
```

CRS 形式の係数行列格納についての詳細は文献 [2] を参照されたい。

2.3 ソルバの呼び出し方法

図 1 にソルバの呼び出し方法を示す。図 1 において、配列 rowptr、colind、val は前述のとおりで、配列 b は右辺ベクトル、配列 solx は解ベクトル、n は連立一次方程式の次元数である。err は打ち切り誤差で、本ソルバでは、右辺ベクトルと残差ベクトルの 2 ノルムの比を用いる。配列宣言用の na、$nnonzero$ は param.h の中で与えているが、それぞれ、次元数、非ゼロ要素数である。これらの値は実際よりも大きくて構わないが、rowptr(n+1) の値は正確に非ゼロ要素数+1 になっていなければならない。ユーザーはまず param.h を修正して、na、$nnonzero$ の値を与える。次に、プログラムの中で CRS 形式で係数行列を作成する。[1] 次に、図 1 のようにソルバのサブルーチンを呼び出す。サブルーチン内で ICCG 法が並列に実行される。本ソルバでは、収束の過程を標準出力に出力する。収束することが分かった場

[1] ICCG ソルバは対称な係数行列に対して有効であるので、通常は係数行列の上三角部分ないし下三角部分を格納すればよいが、本ソルバでは、係数行列全体を格納する。これは、非対称線形システム用ソルバも同様の呼び出し形式で用いることができれば便利であると考えたためである。将来的にはこれらのソルバの公開も考えたい。上または下三角部分の格納のみでソルバ利用したい場合は別途、御連絡を頂きたい。

```
program main
include 'param.h'
integer*4 rowptr(na+1),icol(nnonzero)
real*8 val(nnonzero),b(na),solx(na)
real*8 err

ユーザーにより係数行列を作成する
Make rowptr, icol, val

打ち切り誤差の設定
Set err

右辺ベクトルの設定
Set b

call paraiccg(rowptr,icol,val,b,n,solx,err)

解ベクトルが solx に返却される

end
```

param.h の中身

```
parameter(na=???,nnonzero=???)
```

図 1: ソルバの呼び出し方法

合には、ソルバ内のソースコードを変更して収束過程の出力をコメントアウトしてもよい。

2.4 京大センター VPP-800 での実行

ユーザーの作成したプログラムを user.f とし、ソルバを parasolver.f とする。コンパイルは、mpifrt user.f parasolver.f とすることで行える。会話型の実行では、jobexec -vp 4 ./a.out とすればよい（4CPU 実行時）。-vp の後の数を変更すれば、使用プロセッサ数を変更することができる。但し、運用上会話型では最大 4CPU までで、それ以上の CPU を使用したい場合は、次のようなスクリプトを NQS に投入し、バッチジョブで計算を行う。

```
# @$-lP 16
VPP_MBX_SIZE=268435328; export VPP_MBX_SIZE
cd $QSUB_WORKDIR
./a.out
code=$?
```

詳細はセンターホームページのスーパーコンピュータの項目を参照してもらいたい。

2.5 問い合わせとダウンロード

前述のようにソルバの公開方法は、当面 e-mail によるものとする。take@kudpc.kyoto-u.ac.jp (岩下宛) までメールしてください。1ヶ月程度ユーザーの意見などを聞いた上で、H13 年 4 月頃に、http://www.kudpc.kyoto-u.ac.jp/HPC-WG/にアップロードする予定である。

参考文献

[1] 浅岡 香枝, 平野 彰雄, 稲荷 淳, 岡部 寿男, 金澤 正憲, 「データ並列言語 VPP Fortran による線形計算ライブラリ ScaLAPACK の実現」, 情報処理学会論文誌, Vol. 41, No.5, (2000), pp. 1549-1557.

[2] 長谷川 里見, 長谷川 秀彦, 藤野 清次 訳, 「反復法 Templates」朝倉書店, (1996), pp. 77-78.

[3] 小国 力 訳, 「コンピュータによる連立一次方程式の解法」, 丸善, (1993).

[4] Takeshi Iwashita and Masaaki Shimasaki, "Parallel Processing of 3-D Eddy Current Analysis with Moving Conductor Using Parallelized ICCG Solver with Renumbering Process", IEEE Transaction of Magnetics, Vol. 36-4, (2000), pp. 1504-1509.