計算数理工学論文集 Vol. 8 (2008 年 11 月), 論文 No. 09-081128 JASCOME

数値的に不安定な線型問題のための 多倍長精度・並列直接解法ライブラリの開発

MULTIPLE-PRECISION PARALLEL DIRECT SOLVERS FOR UNSTABLE LINEAR PROBLEMS

藤原 宏志 [1)]

Hiroshi FUJIWARA

1) 京都大学大学院 情報学研究科 (〒 606-8501 京都市左京区吉田本町, E-mail: fujiwara@acs.i.kyoto-u.ac.jp)

We discuss the multiple-precision parallel LU decomposition for numerical solution of inverse and ill-posed problems. Multiple-precision arithmetic is required in reliable numerical treatments of unstable processes, but it takes a lot of computational time. In the presentation, we propose a multiple-precision parallel LU solver for C++ or FORTRAN90, which are based on MPI and exflib[(3)]. The proposed software realizes superlinear speed-up in comparison with a C++ template serial solver.

Key Words: Multiple-Precision Arithmetic, Parallel Computation, Computational Mechanics

1. 緒言

逆問題や非適切問題 (ill-posed problem) の計算機上での扱いで現れる数値的に不安定な問題に対し，多倍長計算による数値解法の有効性が示されつつある．しかし，多倍長計算は一般にソフトウエアで実現されており，計算力学などに現れる大規模問題への適用においては，計算時間およびメモリ量が問題となる．これに対して本研究では，並列計算 (parallel computation) により問題の解決を図る．特に微分方程式の高精度離散化や，積分方程式の離散化で現れる密行列による連立一次方程式を念頭におき，直接解法である LU 分解をとりあげる．並列計算環境として，近年広く利用される PC クラスタなどの分散メモリ計算機を対象とした設計と実装をおこない，多倍長精度計算における並列化効率と有効性について論じる．

工学，医学，地球物理学などに現れる計算機断層撮影法 (Computed Tomography) は，社会の安全性と密接に関連する重要な問題として精力的に研究が進められている．そのうち典型的なものの幾つかは，解析函数核の積分作用素により，適当な Hilbert 空間上での第一種積分方程式で記述される．典型的な逆問題においては，現れる積分作用素はコンパクトであり，特に L^2 空間や Sobolev 空間などの函数空間上で考えると，その逆作用素は有界とはなり得ない．これは第一種積分方程式の直接的な離散化で得られる線型方程式の係数行列の条件数が極めて大きくなることを意味する．したがって，従来手法に基いて分割数を増大させて大規模計算をおこなっても精度の向上は達成され得ず，新たな数値的手法の提案が望まれている．

これに対して近年，多倍長計算とスペクトル法の併用による数値計算の可能性が示された [(4)]．数値計算においては種々の計算誤差が混入するが，今井らは，スペクトル法をもちいて離散化誤差を小さくすると同時に，多倍長計算で丸め誤差を小さくすることで，第一種積分方程式に対して高精度な数値解の構成に成功した．

大規模問題に対する多倍長計算はメモリと計算時間の点から負荷分散による並列計算が効果的と考えられる．本論文では，積分方程式や微分方程式に対するスペクトル法の数値計算で現れる密行列による連立一次方程式の高速解法を目的として，多倍長計算には exflib[(1, 2, 3)] を利用し，並列計算ライブラリには分散メモリ環境での利用を目的として MPI (Message Passing Interface) をもちい，多倍長精度での LU 分解並列ソルバの設計と実装をおこなった．次節で，並列計算におけるメッセージ通信において必要となる多倍長計算のデータ構造について論じ，提案するソルバのアルゴリズムとインターフェースを 3 節で述べる．4 節では，その並列化効率について論じる．

2. 多倍長精度浮動小数点型 exfloat

本研究で利用する多倍長数の型 exfloat のデータ構造を Fig. 1 に示す．exfloat は 64 ビット数の配列からなっており，s は符号部，e_b はバイアス付きの指数部，$f_1, \cdots, f_n$ は仮数

2008 年 9 月 17 日受付，2008 年 10 月 30 日受理

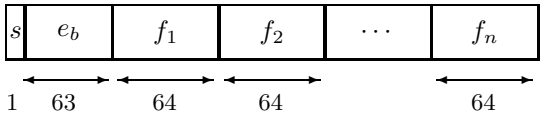

Fig. 1　多倍長精度・浮動小数点型 exfloat のデータ構造

部を示し，Fig. 1 によって

$$(-1)^s \times 2^e \times \left(1 + \sum_{i=1}^{n} f_i \times 2^{-64i}\right)$$

を表す．ここで $e = e_b - (2^{62} - 1)$ である．MPI 並列計算における exfloat 型の交換は，Fig. 1 に示す $n+1$ の 64 ビット数の配列の交換によって実現される．

3. 設計とインターフェース

近年の計算機システムでは階層化された並列化機構が提供される．多倍長計算ライブラリ exflib は四則演算にスーパースカラなど命令レベル並列をおこない，本論文で提案する LU 分解ソルバは，マルチコアやマルチプロセサ，およびクラスタを分散メモリ環境と捉え，プロセスレベル並列をおこなう．プロセスレベルの並列化には MPI (Message Passing Interface) をもちい，その代表的な実装である MPICH2[(5)], OpenMPI[(6)] などで動作を確認している．

提案するソルバは，正則な正方行列の LU 分解と前進代入および後退代入により，連立一次方程式の求解をおこなう．LU 分解には外積形式ガウス消去 (right looking)[(7)] を採用し，列サイクリック分割によって静的な均等負荷分散をおこなっている．軸選択における最大要素の検索や総和などの集約演算には，二項木 (binomial tree) を利用している．

本ライブラリは，Linux, Solaris10 などの UNIX オペレーティング・システムにおいて FORTRAN90，C++言語で利用可能である．FORTRAN90 ではモジュール，C++言語ではクラスとして提供している．これらの言語における典型的な利用例を Fig. 2, Fig. 3 に示す．例は，宣言と初期化，行列成分の設定，LU 分解の実行，右辺値ベクトルの設定，求解，解の出力から構成される．

```
PROGRAM lu

  USE exflib
  USE exflib_la
  ...
  TYPE(exflib_lu_mpi) :: system

  CALL MPI_INIT(ierr)
  CALL MPI_COMM_RANK(MPI_COMM_WORLD, &
    myproc, ierr)

  CALL init(system, N)

  DO i = 1, N
    DO j = 1, N
      IF (has(system,i,j)) THEN
        CALL set_entry(system,i,j, ...)
      END IF
    END DO
  END DO

  CALL decomp(system)

  DO i = 1, N
    IF (has_rhs(system, i)) THEN
      CALL set_rhs(system, i, ...)
    END IF
  END DO

  CALL solve(system)

  IF (myproc == 0) THEN
    DO i = 1, N
      write(*,*) exflib_format('f.6', &
        get_sol(system, i))
    END DO
  END IF

  CALL fin(system)
  CALL MPI_FINALIZE(ierr)

END PROGRAM lu
```

Fig. 2　FORTRAN90 での並列 LU 分解の利用例

4. LU 分解の並列化効率

本研究で実装した LU 分解では，外積形式ガウス消去アルゴリズム [(7)] を採用し，列サイクリック分割によって静的な負荷分散をおこなっている．

計算精度や行列サイズに対する並列化の効果を調べるため，Dual-Core Opteron 2214 (2.2GHz) を 2 個搭載する Sun Fire X2200 M2 で構成されるクラスタをもちいて数値計算をおこなった．オペレーティングシステムは Solaris10 を，MPI ライブラリには mpich2-1.0.6p1[(5)] を利用した．計算時間の比較のため，1 プロセスでの LU 分解には, exflib で利用可能な JAMA 1.2.5/Template Numerical Toolkit 1.2.6[(8)] をもちいた．

10 進法で d 桁の精度は，exfloat 型の仮数部において $1 + \lceil \frac{d}{64} \log_2 10 \rceil$ 要素の配列で表現され，これはおよそ $8 + 0.415d$ バイトに相当する．したがって次元 N の問題を p プロセスで処理する場合，行列を保持するのに要するおよそのメモリ量 (ギガバイト) は，1 プロセスあたり

$$\frac{dN^2}{p} \times \frac{0.415}{1024^3} \approx \frac{dN^2}{p} \times 0.396 \times 10^{-9} \qquad (1)$$

と見積られる．消費メモリ量の理論値と，上記の環境における実測値を Table 1 に示す．

数値計算に要した計算時間等を Table 2 および Fig. 4 に示す．ここで，プロセス数が p 個の場合にプログラムの開始から終了に要する時間，すなわち計算リソースの確保と解放，係数の計算，LU 分解および求解等に要した計算時間を $T(p)$ とすると，速度向上率 (speed up ratio) は

$$S(p) = \frac{T(1)}{T(p)}$$

Table 1　24 プロセスでの LU 分解に必要な合計メモリ量 (単位：giga byte)

decimal digits	100	100	500	500	1000	1000
`exfloat` size (byte)	56	56	216	216	424	424
matrix size	4000	8000	4000	8000	4000	6000
theoretical	0.83	3.34	3.22	12.9	6.32	14.2
MPICH2　24 proc	0.91	3.43	3.34	13.1	6.51	14.5

Table 2　数値計算に要した計算時間 (上段，単位:秒)，速度向上率 (中段) および並列化効率 (下段)

	decimal digits	100	100	500	500	1000	1000
	size of matrix	4000	8000	4000	8000	4000	6000
total memory (theoretical, GB)		0.84	3.3	3.2	12.9	6.3	14.2
JAMA/TNT	1 proc	4124	32952	31825	278340	94757	330733
proposed	4 proc	1162	9224	7538	60690	20641	70960
(MPICH2)		3.55	3.57	4.22	4.59	4.59	4.66
		0.89	0.89	1.06	1.15	1.15	1.17
	8 proc	595	4575	3865	30420	10383	34879
		6.93	7.20	8.23	9.15	9.13	9.48
		0.87	0.90	1.03	1.14	1.14	1.19
	16 proc	319	2347	2011	15516	5294	17644
		12.9	14.0	15.8	17.9	17.9	18.7
		0.81	0.87	0.98	1.11	1.10	1.17
	24 proc	218	1585	1365	10493	3566	11829
		18.9	20.8	23.3	26.5	26.6	28.0
		0.79	0.87	0.97	1.11	1.11	1.17

```
...
#include "exflib_la.h"

int main(int argc, char *argv[])
{
  ....
  MPI::COMM_WORLD.Init(argc, argv);

  exflib_lu_mpi system(N);

  for(i = 0; i < N; i++)
    for(j = 0; j < N; j++)
      if(system.has(i,j))
        system.entry(i,j) = ...;

  system.decomp();

  for(i = 0; i < N; i++)
    if(system.has_rhs(i))
      system.rhs(i) = ...;

  system.solve();

  if(MPI::COMM_WORLD.Get_rank() == 0)
    for(i = 0; i < N; i++)
      cout << system.sol(i) << endl;

  MPI::Finalize();
  ...
}
```

Fig. 3　C++ 言語での並列 LU 分解の利用例

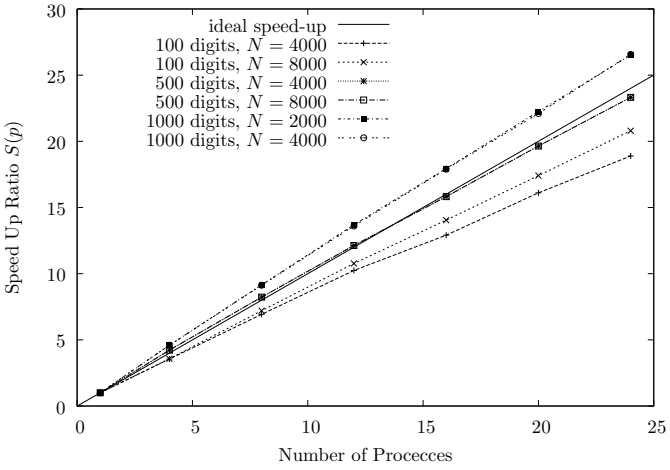

Fig. 4　多倍長精度 LU 分解の速度向上率

Table 3　三重ループ内の積和演算を省略した計算時間 (秒)

decimal digits	100	100	500	1000
matrix size	4000	8000	4000	4000
4 proc	9.6	36	34	79
8 proc	16	60	59	126
16 proc	38	83	53	99
24 proc	30	81	59	82

で定義される．これから，並列化効率 (parallel efficiency)

$$E(p) = \frac{S(p)}{p}$$

を求めると，この範囲のプロセス数では高い並列化効率が実現されており，利用するメモリが多い場合には，$E(p) > 1$ のスーパーリニアな速度向上が達成され，提案する並列計算が有効であることがわかる．ベンチマークに用いた計算機は NUMA (Non-Uniform Memory Access) アーキテクチャであり，また，JAMA はキャッシュ・メモリによるメモリアクセスの局所化のために内積形式ガウス消去法を利用しているため，上述の指標での直接的な比較には注意を要する．

数値計算における演算の主要部は，LU 分解の三重ループに現れる積和演算 (multiply-accumulate) である．三重ループ内の積和演算を省略した場合の計算時間を Table 3 に示す．Table 3 はリソースの確保と解放，軸選択やデータ交換などに要する時間を表す．Table 2 と比較すると，これら積和演算以外の処理が占める計算時間の割合は小さく，計算の高速化には積和演算の高速化が重要であることがわかる．

近年，計算力学など大規模数値計算では分散メモリのクラスタが広く利用されている．今後，マルチコアなどの普及に伴い，多倍長計算の計算力学への適用は一層重要になると考えられる．そこでは，並列計算の階層化はより複雑になると考えられ，共有キャッシュなどを有効に利用する実装とともに，インターフェースの整備も重要となる．

謝辞　本研究の遂行にあたり，日本学術振興会科学研究費 (課題番号 19340022, 20040057) の助成を頂きました．

参考文献

(1) 藤原宏志：高速な多倍長計算環境の PC・WS 上での実現，計算数理工学レビュー No.2005-1, pp33–40 (2005).

(2) 藤原宏志：科学技術計算に適した多倍長数値計算環境の構築と数値的に不安定なスキームの直接計算の実現，情報処理学会論文誌：コンピューティングシステム，Vol. 48, No. SIG 8 (ACS 18) pp.22–30 (2007).

(3) http://www-an.acs.i.kyoto-u.ac.jp/~fujiwara/exflib/index.html

(4) Imai, H. and Takeuchi, T.: Some advanced applications of the spectral collocation method, *GAKUTO Internat. Ser. Math. Sci. Appl.*, Vol. 17, pp. 323–335 (2002).

(5) MPICH2, http://www.mcs.anl.gov/research/projects/mpich2.

(6) OpenMPI, http://www.open-mpi.org/

(7) 島崎眞昭：スーパーコンピュータとプログラミング，共立出版 (1989).

(8) Template Numerical Toolkit, http://math.nist.gov/tnt.