多倍長疎行列計算の性能評価

幸谷　智紀 (静岡理工科大学)

2011 年 8 月 9 日

1　初めに

我々は MPFR[3]/GMP[1] を利用した多倍長数値計算ライブラリ BNCpack[2] を作成してきた。本講演ではこれを土台として新たに追加した多倍長疎行列・ベクトル積の性能を評価した結果と，Krylov 部分空間法に適用した結果について報告する。

2 進・整数演算ベースの任意精度浮動小数点演算ライブラリである MPFR/GMP は，複雑な構造体を用いて任意精度浮動小数点数を定義しており，この中で実際に格納されている仮数部の長さを記憶している (図 1)。これにより，必要な仮数部長だけを演算することができる。例えば $0 \times a = 0$ や $0/a = 0$ は殆どメモリアクセス程度の時間で結果を得ることが出来る演算省力化機構が働くようになっている。そのため，MPFR/GMP を用いた疎行列計算の機能を実装したとしても，記憶領域の削減以外の効果は見込めないと我々は考えていた。

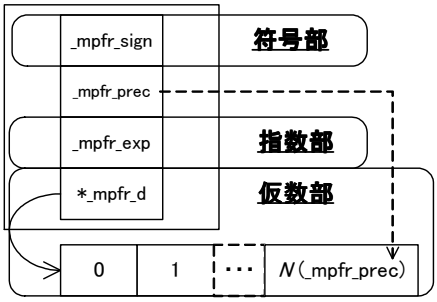

図 1　MPFR/GMP の構造体

今回，CRS(Compressed Row Strage) 形式のランダム疎行列形式を BNCpack で試験的にサポートし，実行列・実ベクトル積のベンチマークテストを行ったところ，確かに記憶領域の削減効果に比べて劣るものの，それなりに計算時間の削減にも繋がることが分かった。その結果，Krylov 部分空間法も同程度の削減効果が得られ，実用上も有用であることが確認できた。

2　BNCpack の疎行列形式

今回実装した CRS 形式の多倍長疎行列を定義した構造体は以下のようになる。

```
typedef struct {
 unsigned long prec; // 精度 (bit)
 mpf_t *element;     // 行列成分
 long int row_dim, col_dim;//次元数
 long int **nzero_index;
  // 非零要素の添字
 long int *nzero_col_dim;//非零数 (行)
 long int *nzero_row_dim //非零数 (列)
 long int nzero_total_num;//非零要素数
} mpfrsmatrix;
```

これを使うと，図 2 のような形式でランダム疎行列が格納できる。

次元数が十分大きければ，記憶領域は零要素の分だけ確実に減らせる。しかし，前述の MPFR/GMP の演算省力化機構の働きにより，計算時間がどの程度削減できるかはベンチマークテストを行ってみなければ分からない。

今回は多倍長実疎行列のデータ型の定義と，実疎行列・実ベクトル積のみ実装した。疎行列以外の定義は既存の BNCpack のものをそのまま利用している。まだデバッグが十分とは言えないが，コードの詳細は BNCpack 最新版 [2] を参照されたい。

3　性能評価

性能評価は次の環境で行った。並列計算は行っておらず，全てシングルスレッドでの逐次計算である。

H/W　Intel Core i7 920, 8GB RAM
S/W　CentOS 5.6 x86_64, gcc 4.1.2
MPFR/GMP　MPFR 3.0.1, GMP 5.0.1
BNCpack　BNCpack 0.6d

疎行列は, 仮数部の桁が全て埋まるよう, 次の Lotkin 行列

$$A = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1/2 & 1/3 & \cdots & 1/n \\ \vdots & \vdots & & \vdots \\ 1/n & 1/(n+1) & \cdots & 1/(2n-1) \end{bmatrix}$$

$$
\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array}
\begin{array}{c}
\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\
\left[\begin{array}{ccccc}
0 & 1 & 1/2 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/2 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{array}\right]
\end{array}
$$

```
element=[1 1/2 1/2 1/2 1/2 1 1]
nzero_total_num = 7
row_dim = 5,col_dim = 5
nzero_index[0] = [1 2]
nzero_index[1] = [0]
nzero_index[2] = [0]
nzero_index[3] = [2 4]
nzero_index[4] = [3]
nzero_col_dim[0] = 2
nzero_col_dim[1] = 1
nzero_col_dim[2] = 1
nzero_col_dim[3] = 2
nzero_col_dim[4] = 1
nzero_row_dim[0] = 2
nzero_row_dim[1] = 1
nzero_row_dim[2] = 2
nzero_row_dim[3] = 1
nzero_row_dim[4] = 1
```

図 2　BNCpack の疎行列形式

の各要素を精度 p(bit) で生成し，ランダムに非対角要素を 0 に置き換えて疎性 s %(100% で零行列，0% で密行列を意味する) の疎行列に変換した $A^{(s)}$ を使用している。実ベクトルは

$$\mathbf{b} = A^{(s,p)}[1\ 2\ \cdots n]^T$$

とし，$A^{(s,p)}\mathbf{b}$ の計算時間 (秒) を計測した。比較のため，疎行列 $A^{(s)}$ を密行列形式で格納したものを $A^{(s)}_{Dense}$ とし，その計算時間 (sec) との比 (密行列/疎行列) を速度向上率 (Speedup Ratio) とする。この結果を図 3 に示す。比較のため，倍精度の結果も載せてある。

倍精度疎行列の実装の効果が現れるのは 2000 次元の時で，ほぼ零要素の比率だけ計算時間は削減されている。それに対して，多倍長疎行列の計算時間削減効果は低く，約 1.3 倍〜3.2 倍程度に留まっている。MPFR/GMP の演算省力化機構が働いているため，零要素との積は非常に短い時間で計算できることがこの低性能向上比の原因であろう。実際，精

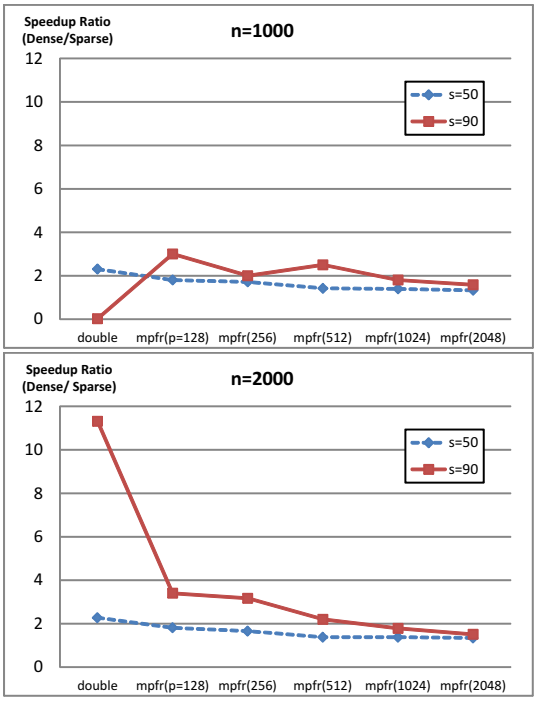

図 3　疎行列・ベクトル積の速度向上比 (密行列

度が大きくなると性能向上比は低くなることから，これが裏付けられる。

以上の結果から，s と p が大きい時程計算時間は短縮され，大規模行列で比較的精度が低い場合は効果が大きいことが判明した。同様のベンチマーク結果は，この疎行列・ベクトル積を適用した Krylov 部分空間法を用いた場合も得られている。詳細は講演時に示すことにする。

4　結論と今後の課題

事前にあまり期待していなかった多倍長疎行列の実装であるが，記憶領域の削減効果に加えて，ある程度の計算時間削減効果があることも判明した。特に比較的精度が低く，大規模な場合は効果が大きくなることも分かった。

今後の課題としては，並列計算の機能を実装し，逐次計算との性能向上比の比較を行うこと，より大規模な科学技術シミュレーションにこの疎行列の機能を適用することが挙げられる。

参考文献

[1] Swox AB. The GNU Multiple Precision arithmetic library. http://gmplib.org/.

[2] Tomonori Kouya. BNCpack. http://na-inet.jp/na/bnc/.

[3] MPFR Project. The MPFR library. http://www.mpfr.org/.