# 高速な4倍精度演算を用いたクリロフ部分空間法の安定化

小武守　恒（JST／東京大学）
藤井 昭宏（工学院大学）
長谷川 秀彦（筑波大学）
西田 晃（中央大学・JST）

## はじめに

- 大規模疎行列に対する線型方程式Ax=bを解く
- クリロフ部分空間法は丸め誤差の影響で収束までに多くの反復が必要または停滞
- 収束の改善には高精度演算，例えば4倍精度演算が有効であるが，計算コスト大
- 反復解法ライブラリLisに高速な4倍精度演算を実装
- 倍精度演算と比較して収束が安定することを確かめる

## 高速な4倍精度演算の実装

- 厳密な4倍精度ではなく
  - 倍精度浮動小数点数を2個用いるdouble-double精度演算を採用　a = a.hi + a.lo, |a.hi|>|a.lo|
  - 仮数部がIEEE準拠より8ビット少ない
  - 有効桁数　double-double精度　約32桁
    IEEE準拠4倍精度　約33桁

double-double精度

| | 指数部 11ビット | 仮数部　52ビット | | 指数部 11ビット | 仮数部　52ビット |
|---|---|---|---|---|---|

IEEE準拠の4倍精度

| | 指数部 15ビット | 仮数部　112ビット |
|---|---|---|

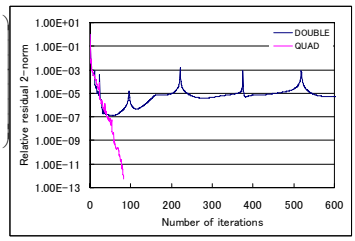
## Lisでの倍精度演算実装方針

- 入力（係数行列A，右辺ベクトル b，初期ベクトル $x_0$ ）は倍精度
- 解法中の解 x，補助ベクトル,スカラーは4倍精度
- 解の出力は倍精度，4倍精度でも可能
  - すべてが4倍精度よりもメモリの削減が可能
- ユーザインタフェースを変えない
- 前処理行列Mの生成部分は倍精度演算
  - 前処理行列Mは係数行列Aの近似
- 反復中のMu = v の求解は4倍精度

## 反復中のMu = v の求解 倍精度 vs 4倍精度

- Toeplitz行列（n=10,000）に対して ILU(0)前処理付BiCG法

$$\begin{pmatrix} 2 & 1 & & & & \\ 0 & 2 & 1 & & & \\ 1.4 & 0 & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & 1.4 & 0 & 2 & 1 \\ & & & 1.4 & 0 & 2 \end{pmatrix}$$




## 数値実験

- Lis4倍精度の性能
  - FORTRAN REAL*16, 倍精度とのスピード比較
  - FORTRAN REAL*16との収束性の比較
- 前処理付Lis 4倍精度反復解法の安定性
  - 倍精度との比較

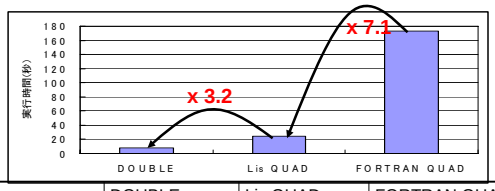
# 実験条件

- 反復解法： BiCG, GPBiCG
- 前処理： Jacobi, ILU(0), SSOR,
  Crout版 ILU(ILUC)
- 右辺ベクトルb
  - 解ベクトルx が x = (1,…,1)$^T$ となるように設定
- 初期ベクトル$x_0$ = (0,…,0)$^T$
- 収束判定基準 $\|r_{k+1}\|_2 / \|r_0\|_2 \leq 10^{-12}$
- 計算環境
  - CPU:Xeon 2.8GHz, OS: Linux 2.4.20smp 32Bit,
    Compiler: Intel C++ 7.0, Intel Fortran 9.0, -O3 -xW

# テスト行列

- 行列A1
  - 2次元ポアソン方程式を有限差分で離散化
  - n=1,000,000
- 行列A2
  - MatrixMarketの行列 rdb2048l n=2048,nnz=12032, cond=1.8E+3
  - 分野 Chemical engineering
- 行列A3
  - MatrixMarketの行列 olm1000 n=1000,nnz=3996, cond=3.0E+6
  - 分野 Hydrodynamics
- 行列A4
  - ホール効果やイオンスリップといった効果を含めた一般化
    されたオームの法則から生じる電場ポテンシャル問題
  - n=23,994,nnz=214,060, cond=2.7E+18
- 行列A5
  - UF Sparse Matrix Collectionの行列 cryg10000
    n=10000,nnz=49699, cond=6.9E+20
  - 分野 CRYSTAL GROWTH EIGENMODES

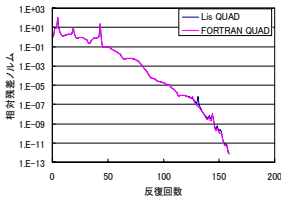
# 行列A1（n=1,000,000）に対する実行時間（BiCG法50回反復）



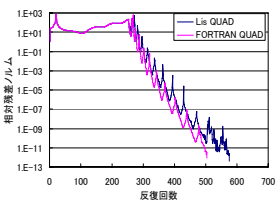

| | DOUBLE | Lis QUAD | FORTRAN QUAD |
|---|---|---|---|
| 行列A(CRS) | 4(n+nnz)+8nnz | 4(n+nnz)+8nnz | 4(n+nnz)+16nnz |
| ベクトルb | 8n | 8n | 16n |
| ベクトルx | 8n | 16n | 16n |
| 補助ベクトル | 6*8n | 6*16n | 6*16n |
| 合計 | 121.9MB | 175.8MB | 221.6MB |

# FORTRAN QUADとの差 前処理なしBiCG

行列A2
（n=2048,nnz=12032,cond=1.8E+3）




行列A3
（n=1000,nnz=3996,cond=3.0E+6）

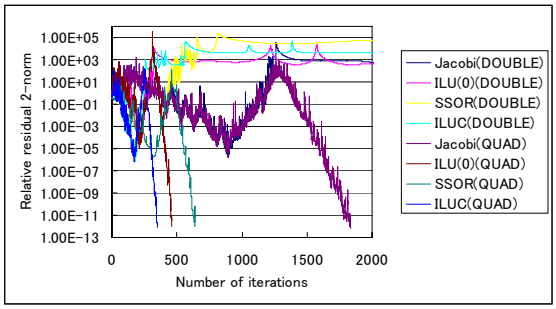



- 反復回数の差は10%程度以下

# 行列A4の結果

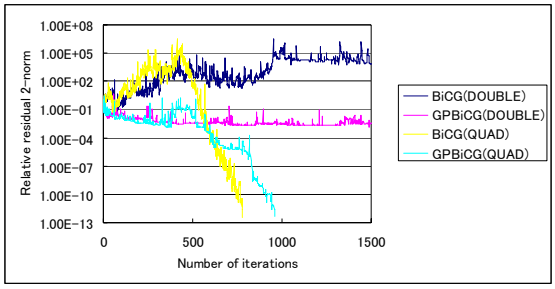
| 前処理 | 倍精度 | | | Lis 4倍精度 | | |
|---|---|---|---|---|---|---|
| | sec. | iter. | TRR | sec. | iter. | TRR |
| BiCG | | | | | | |
| Jacobi | | | | 26.58 | 1833 | 7.68E-15 |
| ILU(0) | | | | 20.41 | 460 | 1.25E-14 |
| SSOR | | | | 29.78 | 642 | 1.27E-14 |
| ILUC | | | | 17.78 | 350 | 1.13E-14 |
| GPBiCG | | | | | | |
| Jacobi | | | | 34.50 | 1403 | 6.89E-15 |
| ILU(0) | 2.99 | 407 | 1.91E-14 | 18.43 | 225 | 1.17E-14 |
| SSOR | | | | 42.53 | 500 | 1.02E-14 |
| ILUC | 11.71 | 364 | 1.67E-14 | 25.95 | 274 | 3.05E-15 |

# 行列A4（ BiCG ）の残差履歴



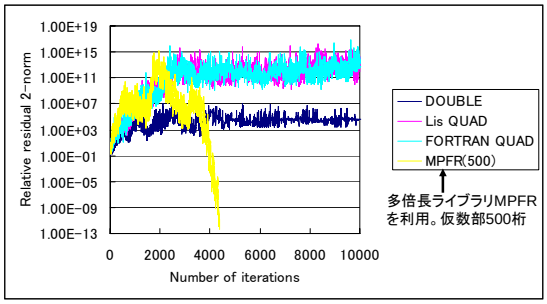

- Lis QUADで収束

## 行列A5(ILUC前処理)の残差履歴




- Lis QUADで収束

## 行列A5(Jacobi-BiCG)の残差履歴




- Jacobi前処理はQUADでは精度が足りない

## まとめ

- 高速な4倍精度演算を実装
  - FORTRAN4倍精度より7.1倍高速
  - 倍精度の3.2倍程度の実行時間
  - 反復回数はFORTRAN4倍精度と同じ～10%増
- Lis 4倍精度の利用
  - 収束する前処理と解法の組み合わせが増加
  - 条件数が非常に大きい場合に有効
  - 前処理の効果を補強
- それでも実行時間がかかる
  - 精度を反復途中で切り替える(倍→4倍)
  - 並列化