連立1次方程式の直接解法とソフトウェア

渡部, 善隆
九州大学情報基盤研究開発センター

https://doi.org/10.15017/1470152

出版情報：九州大学情報基盤研究開発センター全国共同利用システム広報. 1 (2), pp.31-49, 2007-12.
九州大学情報統括本部広報委員会
バージョン：
権利関係：

連立1次方程式の直接解法とソフトウェア

渡部 善隆*

本稿では，数値計算の重要な算法のひとつである連立1次方程式に対する直接法を用いた数値解法について，最近の話題をいくつか紹介します．

連立1次方程式の直接解法は，Gaussの消去法に代表される行列の分解にもとづく手法が現在の主流であり，多くの数値計算の教科書の始めの方に登場します．

以前は,「実用的な直接法の適用範囲は数千次元の密行列や帯行列に限定され，特に大規模疎行列については反復法でないと絶対に解けない！」とされてきました．しかし，近年のアルゴリズムおよび実装技法の急速な発展によって，状況は少し違ってきています．

1　連立1次方程式の直接解法

$n \times n$ の行列 A と n 次元ベクトル $\boldsymbol{x}$, $\boldsymbol{b}$ で表された**連立1次方程式**（simultaneous linear equations[1]）

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

を考えます．立てた数学モデルや離散化の方法などによって，行列 A の性質は異なります．

数値計算の分野では，連立1次方程式 (1) の解 $\boldsymbol{x}$ を求めるための様々な解法が存在します．これまでに提案され，現在も誰かが利用している手法だけでも100種類以上あるのは確実でしょう．

これらの解法の中で，**直接法**（direct method）は，もしも無限桁の数を有限桁で近似する時に生じる誤差（丸め誤差）がなければ，有限回の演算で数学的に厳密な解が得られる解法として分類されます．

$$A,\ \boldsymbol{b} \longrightarrow \boxed{\text{有限回の演算}} \longrightarrow \boldsymbol{x}$$

この定義を厳密に適用すると，共役勾配法と呼ばれる解法 [11] は直接法に分類されてしまいます．ただし，共役勾配法は（A が正定値対称実行列の場合）高々n 回の反復で厳密解に到達することが分かっているものの，n よりも少ない反復回数で収束を目指す解法として使われるため，通常は反復法に分類されます．

1.1　LU 分解

現在用いられている直接法のほとんどは，行列を三角行列（上または下対角成分がすべてゼロ）の積に分解することで解 $\boldsymbol{x}$ を求めます．一般の行列に対する代表的な過程は以下の通りです [12].

*九州大学情報基盤研究開発センター　E-mail: watanabe@cc.kyushu-u.ac.jp

[1] “system of linear equations” あるいは単に “linear equations” とも呼ばれます．

step 1. 行列の各行（列）の絶対値最大を 1 程度にそろえた方程式をあらためて $A\boldsymbol{x} = \boldsymbol{b}$ とする（正規化）

step 2. 必要に応じて方程式の行（列）の入れ替えを行ないながら，A を下三角行列 L と上三角行列 U との積に分解する（LU 分解）

step 3. 連立 1 次方程式 $L\boldsymbol{y} = \boldsymbol{b}$ を解く（前進代入）

step 4. 連立 1 次方程式 $U\boldsymbol{x} = \boldsymbol{y}$ を解く（後退代入）

step 1 の正規化は，行（列）の入れ替えに意味を持たせるための操作で，不要な場合は省略されることもあります．

一般の行列については，計算に要するコストのほとんどは step 2 の分解部分に集中します．具体的には，約 $2/3 \times n^3$ 回の四則演算が必要です．これに対し，その他のステップは多くても n^2 の数倍の演算数で済みます．ハイパフォーマンスコンピュータの性能コンテストで著名な “TOP500” (`http://www.netlib.org/benchmark/top500.html`) で使われる Linpack ベンチマークプログラムの主要な計算はこの LU 分解になります．

step 2 を数式で書くと以下の通りです[2]．

$$PA = LU, \qquad \text{または} \qquad PAP = LU. \tag{2}$$

P は行または列の入れ替えに対応する行列で，どの行にも，またどの列にも，ちょうど 1 個ずつ 1 があり，他はすべて 0 となる置換行列と呼ばれる特別な行列です．ただし，実際に P を作成する必要はなく，置換に対応する整数ベクトルを用意しておけば十分です．

A が正定値性対称の場合は $U = L^T$，対称行列の場合は $U = DL^T$ と分解できるため，メモリの節約ができます．ここで D は対角行列または 2×2 のブロックが対角に並ぶ行列です．

1.2 直接法の利点

一般の行列の場合，直接法に必要な演算回数は，方程式の入れ換え操作を無視すれば計算する前からわかります．したがって，行列の性質に関係なくほぼ同じ計算時間で答が出てきます．そのため，直接法は標準的な解法とされています．

また，前節で紹介した適当な行または列の入れ替えをともなう LU 分解の手順が成功することと A が正則であることは数学的には同値であるため，もし LU 分解が失敗した場合，作成した A は特異または特異に近い行列であるという知見が得られます．さらに，一度 LU 分解した行列を保存しておくことにより，異なる右辺に対する解を容易に計算することができます．

その他，反復法では，前処理技法の選択や初期値の設定が収束性に少なからぬ影響を与えるのに比べ，直接法ではその様な心配は不要です[3]．

[2] 上の二つは特別な場合で，一般に書くと $PAQ = LU$ (P, Q は置換行列) です．

[3] もちろん，浮動小数点演算の際に生じる丸め誤差には注意が必要です．

1.3 直接法の難点

成分のほとんどがゼロの行列を**疎行列**（sparse matrix）と呼びます．偏微分方程式を有限差分法や有限要素法などにより離散化して連立 1 次方程式に帰着させる場合，得られる行列の多くは疎行列になります．

疎行列に対して通常の直接法アルゴリズムを適用した場合，行列が疎行列であっても “ゼロが多い” という利点を生かしにくいことが知られています．

図 1 は，流体の安定性解析[4]の際に現れた 398×398 行列（左）と，それを部分ピボット選択と呼ばれる行の入れ替え手法 [12] を用いた LU 分解で下三角行列と上三角行列に分解したものを重ねた行列（右）の非ゼロ成分の分布です．白い部分がゼロ成分です．分解前はゼ

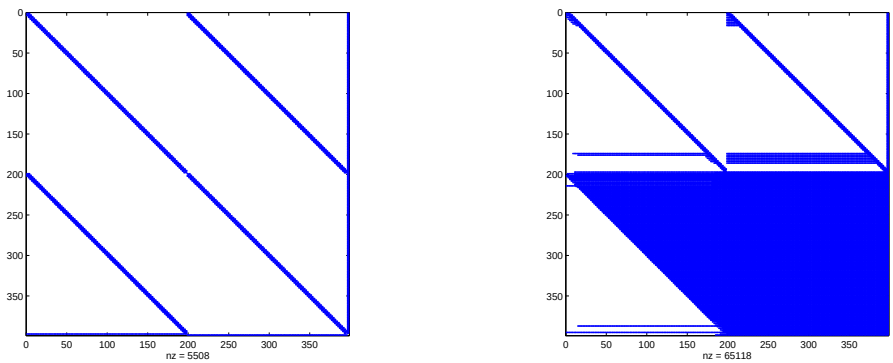

図 1: 行列の非ゼロ成分の分布．LU 分解前（左）と LU 分解後（右）

ロの割合が 96.5%だったものが，LU 分解後 58.8%に減ってしまいました．

行列の分解によるゼロ成分の損失を fill-in と呼びます．直接法では，fill-in の生じる成分の値も必要になるため，次数が大きくなるにしたがって，確保できるメモリ量が大きな問題となります．

直接法のもうひとつの難点に，n が大きくなった場合の演算数の増加があります．一般の問題に対する直接法で必要な演算回数は，ほぼ次数 n の 3 乗に比例します．そのため，次数が大きくなればなるほど，計算機の性能をうまく引き出したプログラムを書いたとしても，“有限回で終了する” といいながらいつまで経っても計算が終らない状況になります．

1.4 帯行列の場合

帯行列（band matrix）とは，ゼロでない成分が行列の対角成分の比較的近くに分布している行列のことです．ゼロでない成分が対角線から遠くに分布している場合でも，帯構造をしていれば帯行列と呼ぶこともあります．

LU 分解の性質として，分解後の行列の非ゼロ成分が帯構造の外に出てくることはありません．図 2 は，ある反応拡散方程式の有限要素計算を行なった際に現れる 841×841 行列（左）と，それを LU 分解で下三角行列と上三角行列に分解したものを重ねた行列（右）の分布です．

[4] とりあえず行列の由来だけ紹介します．「非自己共役複素固有値問題として知られる Orr-Sommerfeld 方程式を実部と虚部で分離し実数表示させ区分的 3 次 Hermite 多項式で離散近似し適当な近似固有値から Newton-Raphson 法で反復改良する際に導かれる非対称行列」です．

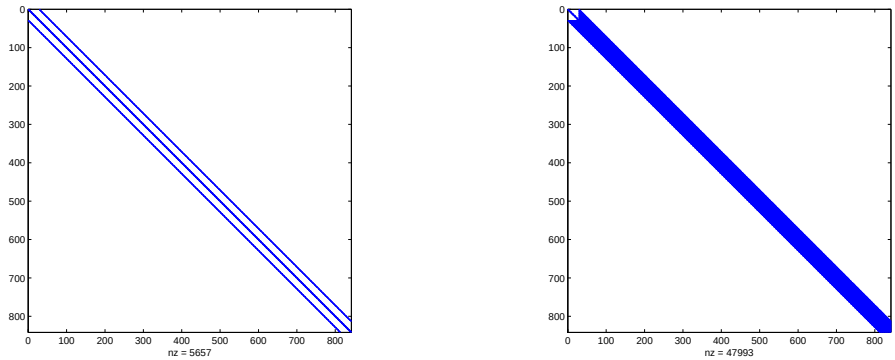

図 2: 帯行列の分布．*LU* 分解前（左）と *LU* 分解後（右）

この行列の場合，対角成分付近だけを配列として記憶しておけばよく，メモリ量および計算量を大幅に削減できます．数値計算プログラムライブラリの多くで，帯行列に対応した直接法を利用することができます．

1.5 直接法と反復法の組合せ

浮動小数点演算において発生する丸め誤差を考えれば，直接法を用いたとしても厳密解が得られるわけではありません．出てくる数値結果からできるだけ誤差をなくすために，直接法で得られた結果を反復法で改良する方法もあります．また，反復法の過程で，方程式の一部を直接法で解く方法もあります．

その他，得られた近似解をもとに，ある集合の中に数学的に厳密な解が存在することを誤差限界とともに示す精度保証付き数値計算法も提案されています．

2 一般行列に対する直接解法ソフトウェア

Mathematica や MATLAB などの統合数値計算環境では，

```
x = A\b                    ← MATLAB
x = LinearSolve(A,b)       ← Mathematica
```

のように，簡単に連立 1 次方程式を解くことができます．

このような実装は C や Fortran などのプログラム言語でも可能です[5]．しかし，規格として組込み関数化されている訳ではありませんので，プログラム言語を用いて連立 1 次方程式を解くには何らかの形でソフトウェアを入手する必要があります．

2.1 プログラムパッケージを使う前に

アルゴリズムを確認する意味でも，まず自分で連立 1 次方程式を解くプログラムを組んでみることをお勧めします．特に，連立 1 次方程式は数値シミュレーションの根幹をなすこと

[5]たとえば `http://www.cc.kyushu-u.ac.jp/RD/watanabe/` では Mathematica もどきのインターフェイスを公開しています．

が多く，この部分の計算を理解していることはとても大切だからです．

次に，きちんと解法の意味を理解した上で，性能向上のためにプログラムパッケージを積極的に利用することも検討すべきです．なぜなら，自作のプログラムに対して，数倍から数十倍も性能差が出ることが珍しくないからです．また，プログラムパッケージは，これまで多くの人から利用されたことによる知見を生かした精度改善，エラーへの対処，細かいオプションの指定などの機能が組み込まれており，信頼性も高いと言えます．

2.2 プログラムパッケージ

表 1 に，主な有償プログラムパッケージをあげます．コンパイラやオペレーティングシステムに依存したり，コンパイラに付属している場合もありますので，選択する際は注意してください．URL は会社のトップページのみ紹介します．リンク先は頻繁に変りますので，各トップページからサイト内検索をかけることをお勧めします．多くのプログラムパッケージでは，連立 1 次方程式の直接解法だけでなく，固有値計算，高速 Fourier 変換，数値微積分などの汎用的な数値計算に対応しています．

名称	提供元	URL
NAG 数値計算ライブラリ	NAG	`http://www.nag-j.co.jp/`
IMSL 数値計算ライブラリ	Visual Numerics	`http://www.vnij.com/`
Math Kernel Library	Intel	`http://www.intel.co.jp/`
ACML	AMD	`http://developer.amd.com/`
MSL2, MATRIX/MPP	HITACHI	`http://www.hitachi.co.jp/`
SSL II	FUJITSU	`http://jp.fujitsu.com/`
ASL	NEC	`http://www.nec.co.jp/`
ESSL	IBM	`http://www.ibm.com/jp/`

表 1: おもな有償プログラムパッケージ

最初の 2 つ: NAG, IMSL は世界的に利用されている数値計算ライブラリで，多くのプラットフォームに対応しています． Math Kernel Library，ACML は，プロセッサを出荷している会社の製品です．最後の 4 つは，計算機ベンダ製です．

上で紹介したプログラムパッケージの中で，一般の行列に対する連立 1 次方程式解法はすべて部分ピボット選択付き *LU* 分解法に基づくアルゴリズムを採用しています．ただし，関数・サブルーチンの名前や引数の指定方法は，残念ながらまちまちなことが多く，他のプログラムパッケージから乗り換える場合には，プログラムの書き換えが（ものすごい手間ではないものの）必要になります．

たとえば，後に紹介する LAPACK の中のひとつである密実行列に対する連立 1 次方程式の解法 `DGESV` を Fortran プログラムから呼び出すには

```
CALL DGESV(N,NRHS,A,LDA,IPIV,B,LDB,INFO)
```

と記述します．`N` は方程式の次数，`NRHS` は方程式の本数（一度 *LU* 分解した場合，異なる右辺で何度も方程式が解けるため），`A` は行列，`LDA` は `A` の実際の宣言寸法，`IPIV` は部

分ピボット選択の情報を格納するベクトル，B は右辺，LDB は右辺ベクトルの実際の宣言寸法，INFO は計算に関する情報を引数として与えます．

このサブルーチンを，たとえば富士通株式会社提供の Fortran サブルーチンライブラリ SSL II に差し換えると，次のようになります．

```
CALL DM_VLAX(A,LDA,N,B,EPSZ,ISW,IS,IPIV,INFO)
```

DGESV で用いた変数以外のものがいくつか現れます．具体的に，EPSZ はピボット選択の閾値，ISW は方程式の本数，IS は行列式のための情報です．また，引数の順番も変ってきます．

特に，富士通，日立，NEC の計算機ベンダでは，最高の性能を引き出すソルバーは自社製のインターフェイスとなっています．

2.3 可搬性を維持するには

性能を引き出すためとはいえ，システムに応じてプログラムを変更するのは，面倒な上に書き間違えによるバグの混入の恐れがあります．

プログラムの書き換えを行なわず，それなりの性能を引き出す方法のひとつとして，線形代数に関するパッケージ LAPACK を利用する方法があります．フリーの数学ソフトウェアパッケージを集めた世界的に著名なサイトに netlib (http://www.netlib.org/) があります．ここから，LAPACK およびその演算の核となる BLAS がダウンロードできます．

LAPACK は，機能ごとに完結したソースコードをダウンロードすることができますので，必要な部分だけ自分のプログラムに組み込んで利用することができます．また，より性能を重視する場合には，使う計算機用に BLAS を最適化する機能を備えた ATLAS をあわせて利用することもできます．

実は，表 1 で紹介したプログラムパッケージのいくつかはチューニングされた LAPACK をサポートしています．たとえば，Math Kernel Library, ACML, NAG 数値計算ライブラリは LAPACK の機能を包含しています．また，富士通の Fortran/C/C++コンパイラを購入すると LAPACK が標準で付属されています．

インターフェイスは同じですのでソースの書き換えは必要ありませんし，中身は最適化が行なわれています．したがって，LAPACK の環境がない場合にはダウンロードしたソースを使い，サポートされている場合にはソース版と差し換えることで，良い性能が引き出せることが期待されます．

この方法は，一般の連立 1 次方程式の直接解法に限らず，LAPACK がサポートする他の解法（固有値問題，特異値分解など）にも応用ができます．

2.4 実際の性能は？

ここでは，九州大学情報基盤センターの IBM eServer p5 モデル 595 の 16 コアまでを用いた性能を紹介します．具体的な環境は表 2 の通りです．

“LAPACK”は，netlib の LAPACK 3.0 版のサブルーチン DGESV をダウンロードしたもの，“ESSL” は，IBM 提供の科学技術計算用ライブラリ ESSL（Engineering and Scientific Subroutine Library）のサブルーチン DGESV を用いたものです．それぞれ，サブルーチン名

計算機名	IBM eServer p5 モデル 595
プロセッサ	POWER5 1.90GHz(Turbo) 64 コア
メモリ	256GB
L2 キャッシュ	1.9MB/2 コア
L3 キャッシュ	36MB/2 コア
OS	AIX 5L V5.3
コンパイラ	IBM XL Fortran Enterprise Edition V9.1

表 2: 計算環境

と引数はまったく同じですので，`DGESV` を呼び出すソースプログラムの修正は必要ありません．netlib からダウンロードしたソースプログラムは，自動並列化オプション付きの

`-O3 -qsmp=auto -qtune=pwr5 -qsmp=auto` を用いて翻訳しました．

行列の次数は 10,000，成分は $A_{i,j} = \sqrt{\frac{2}{n+1}} \sin(\frac{ij\pi}{n+1})$，右辺は A の各列和で生成しました．

eServer p5 は共有メモリ型並列計算機です．したがって，実行時にスレッド並列の環境変数を指定することで並列実行を行なうことができます．

連立 1 次方程式の求解に要する経過時間を時間計測関数により測定しました．図 3 は，連立 1 次方程式の解を求めるまでの時間（秒）を各スレッド数ごとに棒グラフで表したものです．まったく同じインターフェイスをソースから ESSL に差し換えるだけで格段の性能向上が得られています．

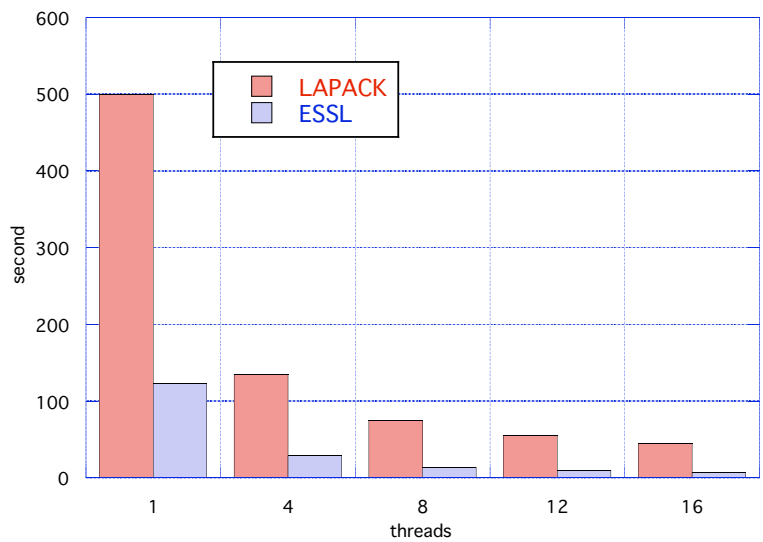

図 3: 連立 1 次方程式の解を求めるまでの時間

計算機の実数演算の性能の指標として使われる値に FLOPS というものがあります．これは「1 秒間に何回の演算を行なったか」を表します．eServer p5 はクロック周波数 1.9GHz の計算機で，この周波数の各テンポで 4 回の実数計算（浮動小数点演算）を実行することができます．したがって，逐次処理における理論ピーク性能は 1.9×4 で得られる 7.6GFLOPS になり，これにスレッド数をかけたものが「これ以上の性能は出ない」理論ピーク性能になります．

図 4 は，図 3 の結果を FLOPS 値で表わしたものです．特に実数計算をたくさん行なうプ

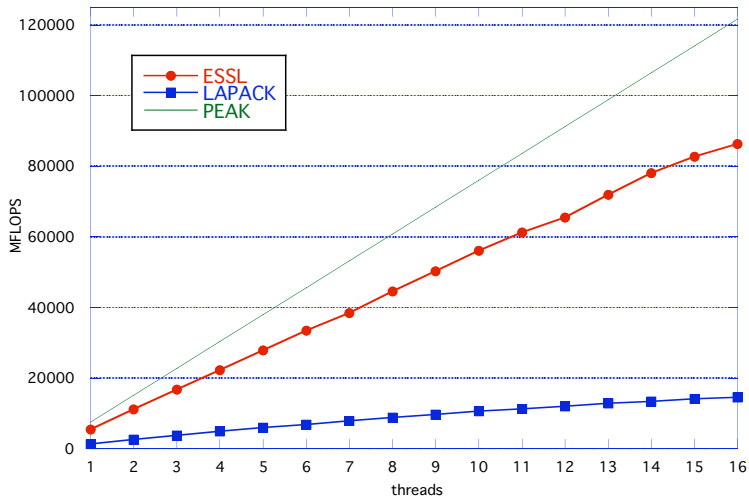

図 4: ピーク性能比

ログラムにおいては，FLOPS 値がピーク性能（PEAK）に近づくのが理想です．しかし，実際にそのような性能を出すコードはなかなか作成できません．その意味で，ESSL の性能は，計算機の性能を十分に引き出していると言ってよいでしょう．

3　疎行列に対する直接法

前節で，密行列に対する連立 1 次方程式の直接解法の性能を紹介しました．現在の並列計算機を使えば，$10,000 \times 10,000$ 程度の行列でしたら 10 秒以下で解が得られます．また，対称行列や帯行列でしたら，その特性を生かしたサブルーチン・関数を用いてより速く答えが得られることが期待されます．

ただし，より大きな次数の方程式を直接法で解く場合には，1.3 節で述べたように計算時間とメモリ量が大きな問題となります．そこで，応用分野で現れる行列が疎行列であるということを生かした工夫を施す必要が出てきます．この節では，疎行列に対する直接法の概要を紹介します．

疎行列の構造の特殊性を利用した直接解法としては，以前からバンドマトリックス法，スカイライン（skyline）法，ウェーブフロント法（wavefront）法などが用いられてきました [13, 15]．1990 年代の後半から，これまでの成果をふまえ，並列計算機への実装技術の深化とともに，多くの疎行列に対する直接法ソフトウェアパッケージが公開され，各種のアプリケーションソフトウェアへの組込みを含め，急速に普及しつつあります [3]．

3.1　基本的な手順

疎行列に対する連立 1 次方程式の直接解法と言っても，基本は 2 節で紹介した *LU* 分解の過程を経て解を求めます．ただし，

- fill-in をできるだけ減らすような行または列の入れ替えを行ない
- *LU* 分解後の行列構造をあらかじめ求めることによって，不要な計算を省く

工夫を行ないます．これらの工夫はグラフ理論と深く関係する非数値的・組み合わせ論的な考察に基づいています [11, 5, 7].

解を求めるまでの段階を大きく分けると，

1. Ordering
2. Symbolic factorization
3. Numerical factorization
4. 前進・後退代入
5. 反復改良

となります．なお，正規化の過程は省略します．日本語の用語として定着していないと思われるものは，英語表記にしました．以下，各段階について簡単に説明します.

3.2 Ordering

この段階では，行または列の並べ替えを行ないます．並べ替えの主目的は fill-in の数を減らすことによって，メモリと演算量を（できれば劇的に）削減することです．この手順は fill reducing ordering とも呼ばれます．この過程で，並列計算における負荷分散も考慮した並べ替えをあわせて行うこともあります.

ただし，fill-in を最小にする最適な置換を探すという問題は，NP 完全であることが証明されています [16]．したがって，いかに十分満足できる ordering を導くかが fill reducing ordering アルゴリズムの目的になります.

fill reducing ordering には，現在のところ決定版はなく，多くのソフトウェアパッケージは複数のアルゴリズムを採用しています．表 3 に，現在よく使われているアルゴリズムを示します.

アルゴリズム名	提案者, 年	略記
Minimum Degree	Tinney and Walker, 1967 [14]	MD
Multiple Minimum Degree	Liu, 1985 [9]	MMD
Approximate Minimum Degree	Amestoy, Davis and Duff, 1996 [1]	AMD
Nested Dissection	George, 1973 [6]	ND
Multisection	Ashcraft and Liu, 1998 [2]	MS
Minimum Fill	Tinney and Walker, 1967 [14]	MF

表 3: fill reducing ordering アルゴリズム

アメリカ・ミネソタ大学で開発されたグラフ分割プログラム METIS [8] には，Nested Dissection アルゴリズムにもとづく fill reducing ordering プログラム `METIS_NodeND` が含まれています．疎行列の直接解法プログラムパッケージには，`METIS_NodeND` のインターフェイスをサポートしているものがあります.

図 1 で紹介した 398×398 行列に，MATLAB R2006a の Approximate Minimum Degree アルゴリズムを実装した関数 `colamd` を用いて並び替えた行列（左）と，それを LU 分解したものを示します．ordering をしない場合には 58.8% だった零要素の割合が，83.2%と向上しています.

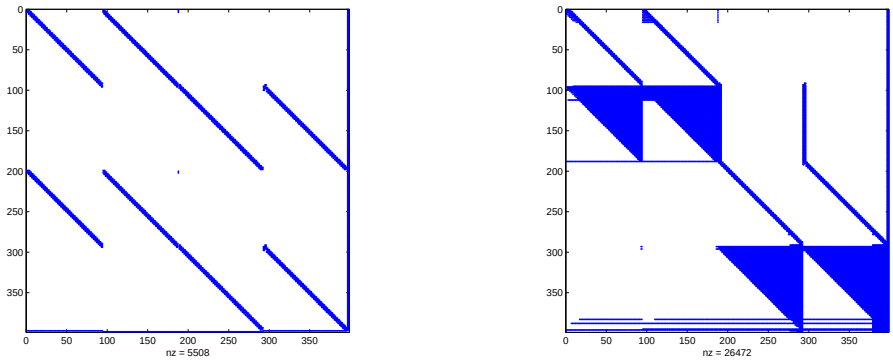

図 5: 行列の非ゼロ要素の分布．LU 分解前（左）と LU 分解後（右）

実際の計算では，ordering によって行列の成分を本当に入れ替えることはせず，置換のための表を整数型配列として作成します．したがって，ほとんどのプログラムパッケージで他の方法で作成した置換表を入力することが出来ますし，ordering の結果だけを出力し，他の目的（たとえば反復法の前処理）に用いることもできます．

3.3 Symbolic factorization

この段階では，行列の分解の過程を表す有向グラフ（directed graph）を作成し，L と U の構造を計算します．ただし，計算は非数値的（浮動小数点演算を使わず）に行なわれます．この計算によって，分解後の構造とともに，実際に必要になるメモリや計算量が見積もられます．

依存関係のグラフの作り方は，プログラムパッケージによって異なります．詳細は，それぞれのソフトウェアに示された参考文献を参照してください．

非線形問題の数値解法では Newton-Raphson 法のような繰り返し計算をよく用います．その時，連立 1 次方程式の行列の値は反復の度に変っていくものの，非ゼロ構造は同じ場合がほとんどです．

symbolic factorization は非数値的な計算ですので，ここまでの過程を保存しておくことにより，前回の計算と同じ構造をしていて値が異なる行列に対しては，これ以降のステップのみで計算可能です．

3.4 Numerical factorization

この段階では，実際に行列を分解します．多くのプログラムパッケージでは，並列性やメモリの有効利用の観点から，マルチフロンタル法（multifrontal mathod）[4, 10]，あるいはマルチフロンタル法を拡張したアルゴリズムを採用しています．

他にも right-looking, left-looking, left-right looking アルゴリズムが用いられることもあります．

3.5 前進・後退代入

分解された三角行列 L, U を使って解を求めるいつもの計算部分です．右辺 $\boldsymbol{b}$ が複数ある場合には，Numerical factorization までの過程を保存しておくことで，解が効率よく得られます．

3.6 反復改良

ソフトウェアパッケージによっては，得られた近似解を初期値 $\boldsymbol{x}^0$ として，

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + A^{-1}(\boldsymbol{b} - A\boldsymbol{x}^k) \qquad k \geq 1$$

で解の精度を改良することができます．

$A^{-1}(\boldsymbol{b} - A\boldsymbol{x}^k)$ の計算は，A の逆行列を求めるのではなく，分解された L, U を用いた前進・後退代入で求まります．

また，ある閾値を設けた上で Numerical factorization を「ルーズ」に行なって得られた近似解（ということは直接法の近似解とは言えません）を初期ベクトルとして反復法を適用するという手法も提案されています．

3.7 プログラムパッケージ

表 4 に，疎行列に対するおもな直接法プログラムパッケージを示します．

コード名	fill reducing ordering	入手先
BCSLIB-EXT	MMD, METIS	www.scs.co.jp/BCSLIB/
MA57	MD, AMD, METIS	www.cse.clrc.ac.uk/nag/hsl/
MUMPS	MD, AMD, METIS	graal.ens-lyon.fr/MUMPS/
PARDISO	MMD, METIS	www.computational.unibas.ch/cs/scicomp/software/pardiso/
SPOOLES	MMD, ND, MS	http://netlib.bell-labs.com/netlib/linalg/spooles/spooles.2.2.html
SuperLU	MMD	http://crd.lbl.gov/ xiaoye/SuperLU/
SMS-MF	ND	http://www.vinas.com/jp/seihin/sms/
TAUCS	MD, AMD, MMD, METIS	http://www.tau.ac.il/~stoledo/taucs/
UMFPACK	AMD	http://www.cise.ufl.edu/research/sparse/umfpack/
WSMP	ND, MF	http://www-users.cs.umn.edu/~agupta/wsmp.html

表 4: 疎行列に対する直接法ソフトウェアパッケージ

4 WSMP の性能評価

この節では，疎行列に対する直接法プログラムパッケージのひとつ WSMP を用いた実測値を紹介します．

4.1 WSMP の概要

WSMP（Watson Sparse Matrix Package）は，IBM T.J. Watson Research Center の Anshul Gupta さんが中心となって作成した疎行列に対する連立 1 次方程式の直接解法ソフトウェアパッケージです．行列は実対称正定値，実対称不定値，実一般，複素 Hermite，複素対称，複素数一般に対応し，Fortran, C から利用可能です．精度は倍精度，圧縮格納形式は CSR, CSC, CSR-UT, CSC-LT, MSR, MSC に対応します．内部で Level 3 BLAS を呼び出します．逐次版は無料でダウンロード可能です．有償版はスレッド並列，MPI 並列に対応します．ソースコードは非公開です．

以降の測定は，2 節と同じ IBM eServer p5 の 16 スレッド並列を用いて行ないました．WSMP のバージョンは 5.7.1 のスレッド並列版です．オプションはすべて標準値を採用しています．

4.2 テスト問題

行列は，

- Matrix Market:
  `http://math.nist.gov/MatrixMarket/`
- フロリダ大学疎行列コレクション:
  `http://www.cise.ufl.edu/research/sparse/matrices/`
- CCLRC 疎行列データベース:
  `ftp://ftp.numerical.rl.ac.uk/pub/matrices/symmetrix/`

で公開されている圧縮列格納を用いました．右辺は，真の解が $\boldsymbol{x} = \mathbf{1}$ となるように A の各行和として設定しました．なお，加算により発生する丸め誤差は無視しています．$\boldsymbol{x} = \mathbf{1}$ と近似解との精度は最大値ノルムで測り，解を求めるまでの経過時間をサブルーチンの前後に時間計測関数を挿入し測定しました．

4.3 性能評価（正定値対称行列）

表 5 に，テスト行列を示します．比較の解法として，IBM ESSL V4.2 の

- DSRIS^{-10}: 不完全 Cholesky 分解付き CG 法，初期値 $\boldsymbol{x}^0 = \mathbf{0}$,
  反復停止基準 $\|\boldsymbol{b} - A\boldsymbol{x}^n\|_2/\|\boldsymbol{x}^n\|_2 < 10^{-10}$.
- DSRIS^{-6}: 不完全 Cholesky 分解付き CG 法，初期値 $\boldsymbol{x}^0 = \mathbf{0}$,
  反復停止基準: $\|\boldsymbol{b} - A\boldsymbol{x}^n\|_2/\|\boldsymbol{x}^n\|_2 < 10^{-6}$.
- DGESV: 部分ピボット選択付き LU 分解， 一般行列用直接法.

を用いました．表 6 に計算時間を，表 7 に誤差を示します．“—” は何らかの原因で実行が打ち切られたことを意味します．表中の太字はもっとも優秀なもの，斜体は解が求まったものの実用的に「？」と思われるものです．

ファイル名	次元数	非零要素数	分野
bodyy6.RSA	19366	77057	NASA matrix
bcsstk36.RSA	23052	583096	automobile shock absorber
msc23052.RSA	23052	588933	MSC.NASTRAN
THREAD.rsa	29736	2249892	Threaded connector
GRIDGENA.rsa	48962	329485	Grid generation optimization
nasasrb.RSA	54870	1366097	Shuttle rocket booster
crankseg_2.rsa	63838	7106348	Linear static analysis
OILPAN.rsa	73752	1835470	Car olipan
finan512.RSA	74752	335872	Portfolio optimization
s3dkq4m2.rsa	90449	2455670	Cylindrical Shell
M_T1.rsa	97578	4925574	Tubular joint
X104.rsa	108384	5138004	Beam joint
cfd2.RSA	123440	1605669	CFD pressure matrix
bmwcra_1.rsa	148770	5396386	Automotive crankshaft model
SHIPSEC5.rsa	179860	5146478	Ship section
hood.rsa	220542	5494489	Car hood
pwtk.RSA	217918	5926171	pressurized wind tunnel
inline_1.rsa	503712	18660027	Inline skater
audikw_1.rsa	943695	39297771	Automotive crankshaft model
ldoor.rsa	952203	23737339	Large door

表 5: テスト行列（正定値対称）

名称	次元数	非零要素数	WSMP	DSRIS^{-10}	DSRIS^{-6}	DGESV
bodyy6.RSA	19366	77057	0.178	0.205	**0.130**	63.162
bcsstk36.RSA	23052	583096	**0.589**	—	66.587	97.816
msc23052.RSA	23052	588933	**0.581**	—	89.271	97.329
THREAD.rsa	29736	2249892	**3.454**	—	—	204.202
GRIDGENA.rsa	48962	329485	—	—	—	**951.616**
nasasrb.RSA	54870	1366097	**1.340**	109.20	42.414	1370.299
crankseg_2.rsa	63838	7106348	**6.573**	43.241	31.185	2064.227
OILPAN.rsa	73752	1835470	**1.644**	80.049	47.560	3044.869
finan512.RSA	74752	335872	0.7107	0.164	**0.129**	3456.823
s3dkq4m2.rsa	90449	2455670	**2.227**	172.90	147.08	—
M_T1.rsa	97578	4925574	**4.299**	4997.9	1956.1	—
X104.rsa	108384	5138004	**5.476**	9495.7	2443.5	—
cfd2.RSA	123440	1605669	**2.902**	22.767	14.233	—
bmwcra_1.rsa	148770	5396386	**6.024**	202.61	174.55	—
SHIPSEC5.rsa	179860	5146478	**5.629**	140.18	89.481	—
hood.rsa	220542	5494489	**4.448**	60.943	31.676	—
pwtk.RSA	217918	5926171	**5.138**	2496.6	582.79	—
inline_1.rsa	503712	18660027	**19.138**	4365.5	1316.5	—
audikw_1.rsa	943695	39297771	**121.63**	2658.5	1980.4	—
ldoor.rsa	952203	23737339	**22.910**	625.83	370.67	—

表 6: 正定値対称行列の計算時間

名称	次元数	非零要素数	WSMP	DSRIS^{-10}	DSRIS^{-6}	DGESV
bodyy6.RSA	19366	77057	**0.975E-12**	0.66E-08	0.16E-03	0.182E-11
bcsstk36.RSA	23052	583096	**0.146E-06**	—	*0.47E+00*	0.848E-03
msc23052.RSA	23052	588933	**0.125E-06**	—	*0.54E+00*	0.239E-03
THREAD.rsa	29736	2249892	**0.195E-07**	—	—	0.595E-07
GRIDGENA.rsa	48962	329485	—	—	—	**0.406E-11**
nasasrb.RSA	54870	1366097	**0.471E-09**	0.60E-06	0.26E-01	0.175E-08
crankseg_2.rsa	63838	7106348	**0.342E-11**	0.12E-07	0.20E-03	0.675E-10
OILPAN.rsa	73752	1835470	**0.640E-09**	0.10E-06	*0.11E+00*	0.199E-08
finan512.RSA	74752	335872	**0.133E-14**	0.71E-09	0.32E-04	0.877E-14
s3dkq4m2.rsa	90449	2455670	**0.171E-06**	0.45E-06	0.25E-03	—
M_T1.rsa	97578	4925574	**0.233E-06**	0.13E-04	0.81E-01	—
X104.rsa	108384	5138004	**0.256E-05**	0.52E-05	*0.17E+00*	—
cfd2.RSA	123440	1605669	**0.142E-11**	0.30E-06	0.35E-02	—
bmwcra_1.rsa	148770	5396386	**0.364E-10**	0.34E-06	*0.33E+00*	—
SHIPSEC5.rsa	179860	5146478	**0.256E-08**	0.47E-06	0.15E-02	—
pwtk.RSA	217918	5926171	**0.108E-06**	0.71E-03	*0.26E+01*	—
hood.rsa	220542	5494489	**0.136E-10**	0.28E-06	0.76E-02	—
inline_1.rsa	503712	18660027	**0.245E-09**	0.37E-06	*0.13E+01*	—
audikw_1.rsa	943695	39297771	**0.169E-10**	0.18E-06	0.15E-01	—
ldoor.rsa	952203	23737339	**0.891E-11**	0.45E-05	0.97E-02	—

表 7: 正定値対称行列の計算精度

4.3.1 各ステップの割合

表5の行列 `ldoor.rsa`，`finan512.RSA`，`audikw_1.rsa` に対し，3.1 節で紹介した各ステップが全体の時間に占める割合（%）をスレッド数を変えながら測定した結果を表8に示します．反復改良は省略値では1回のみです．また，step0 は，パラメータの設定などの初期化部分です．

4.3.2 並列化効果

図6に，`ldoor.rsa`，`finan512.RSA`，`audikw_1.rsa` に対しスレッド数を変化させた並列化効果を示します．

“ideal” が，理想の並列化効果です．`ldoor.rsa`，`finan512.RSA` は全体の実行時間が短いこともあり，4スレッド並列あたりで性能が止っています．これに対し，`audikw_1.rsa` は，比較的良好な並列性能を出しています．

4.4 性能評価（不定値対称行列）

表9に，テスト行列を示します．比較の解法として IBM ESSL V4.2 の

- DSRIS
  対角スケーリング付き TFQMR 法， 疎行列用反復法 $\boldsymbol{x}^0 = \mathbf{0}$,
  反復停止基準: $\|\boldsymbol{b} - A\boldsymbol{x}^n\|_2/\|\boldsymbol{x}^n\|_2 < 10^{-10}$
- DGESV: 部分ピボット選択付き LU 分解， 一般行列用直接法.

を用いました．表10に計算時間を，表11に誤差を示します．

行列名	並列度	step0	step1	step2	step3	step4	step5
ldoor.rsa	1	0.01	61.22	3.45	31.28	1.91	2.13
	2	0.01	59.58	7.11	29.28	1.83	2.19
	4	0.01	62.05	10.56	23.97	1.44	1.97
	8	0.02	65.43	13.66	18.14	1.06	1.69
	12	0.02	66.84	14.01	16.96	0.74	1.42
	16	0.03	67.28	14.06	16.43	0.74	1.46
finan512.RSA	1	0.18	77.58	4.25	12.32	2.80	2.84
	2	0.27	71.59	9.11	13.99	2.49	2.52
	4	0.40	62.41	12.05	20.75	2.09	2.26
	8	0.52	54.76	13.66	27.64	1.49	1.89
	12	0.60	52.64	13.23	30.81	1.26	1.41
	16	0.62	52.96	13.41	30.00	1.26	1.71
audikw_1.rsa	1	0.00	9.92	0.49	88.34	0.61	0.63
	2	0.00	8.91	1.08	88.78	0.59	0.63
	4	0.00	10.33	1.85	86.65	0.55	0.62
	8	0.00	14.03	3.20	81.18	0.74	0.85
	12	0.00	22.77	5.18	70.69	0.58	0.78
	16	0.00	25.85	5.92	66.65	0.69	0.89

表 8: 各ステップの占める割合（%）

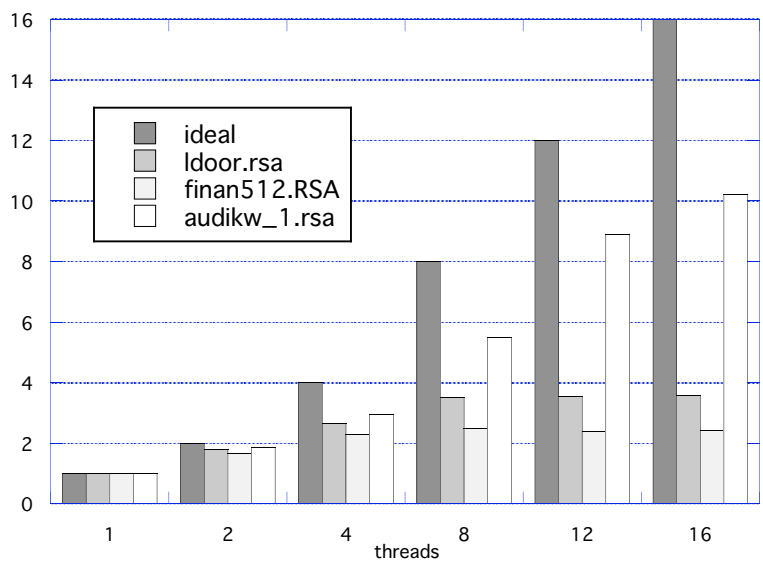

図 6: 並列化効果

名称	次元数	非零要素数	分野
SPMSRTLS.rsa	29995	129971	Sparse matrix square root
HELM3D01.rsa	32226	230335	Helmholtz problem
bcsstk39.RSA	46772	1068033	shuttle rocket booster
SPARSINE.rsa	50000	799494	Structural optimization
copter2.rsa	55476	407714	Helicopter rota blade
DIXMAANL.rsa	60000	179999	Dixon-Maany optimization
c-71.RSA	76638	468096	Optimization model
A0NSDSIL.rsa	80016	200021	Linear Complementarity problem
c-72.RSA	84064	395811	Optimization model
bmw3_2.rsa	227362	5757996	Linear static analysis
HELM2D03.rsa	392257	1567096	Helmholtz problem

表 9: テスト行列（不定値対称）

名称	次元数	非零要素数	WSMP	DSRIS	DGESV
SPMSRTLS.rsa	29995	129971	**0.215**	—	233.297
HELM3D01.rsa	32226	230335	**0.648**	—	285.757
bcsstk39.RSA	46772	1068033	**0.896**	—	891.506
SPARSINE.rsa	50000	799494	**41.513**	—	1003.262
copter2.rsa	55476	407714	**0.969**	—	1501.076
DIXMAANL.rsa	60000	179999	0.412	**0.046**	1760.508
c-71.RSA	76638	468096	4.827	**4.085**	3924.913
A0NSDSIL.rsa	80016	200021	—	—	**4183.087**
c-72.RSA	84064	395811	**7.215**	7.510	—
bmw3_2.rsa	227362	5757996	**5.457**	—	—
HELM2D03.rsa	392257	1567096	**3.683**	—	—

表 10: 不定値対称行列の計算時間

名称	次元数	非零要素数	WSMP	DSRIS	DGESV
SPMSRTLS.rsa	29995	129971	**0.103E-11**	—	0.104E-11
HELM3D01.rsa	32226	230335	**0.104E-11**	—	0.255E-10
bcsstk39.RSA	46772	1068033	**0.150E-10**	—	0.273E-09
SPARSINE.rsa	50000	799494	**0.403E-10**	—	0.124E-07
copter2.rsa	55476	407714	**0.259E-12**	—	0.121E-09
DIXMAANL.rsa	60000	179999	**0.156E-12**	0.13E-07	0.128E-10
c-71.RSA	76638	468096	**0.132E-11**	0.25E-05	0.119E-09
A0NSDSIL.rsa	80016	200021	—	—	**0.194E-09**
c-72.RSA	84064	395811	**0.160E-11**	0.28E-05	—
bmw3_2.rsa	227362	5757996	**0.804E-06**	—	—
HELM2D03.rsa	392257	1567096	**0.232E-12**	—	—

表 11: 不定値対称行列の計算精度

4.5 性能評価（非対称行列）

表 12 に，テスト行列を示します.

名称	次元数	非零要素数	分野
west2021	2021	7353	Chemical engineering
fidap011	16614	1096948	Fluid dynamics
e40r0100	17281	553562	Fluid dynamics
memplus	17758	126150	Electronic circuit design
raefsky3	21200	1488768	Fluid dynamics
af23560	23560	484256	Fluid dynamics
wang3	26064	177168	Fluid dynamics
lhr34c	35152	764014	Chemical engineering
onetone1	36057	341088	Nonlinear circuits
onetone2	36057	227628	Nonlinear circuits
bbmat	38744	1771722	Fluid dynamics
av41092	41092	1683902	Finite Element Analysis
bayer01	57735	277774	Chemistry
venkat50	62424	1717792	Fluid dynamics
epb3	84617	463625	Thermodynamics
twotone	120750	1224224	Nonlinear circuits
torso3	259156	4429042	Fluid dynamics
languag	399130	1216334	Natural-language processing
pre2	659033	5959282	Nonlinear circuits
hamrle3	1447360	5514242	Electronic circuit design

表 12: テスト行列（非対称）

比較の解法として IBM ESSL V4.2 の

- DGSF(S): 修正 Markowitz 法による LU 分解，疎行列用直接法
- DSRIS: 不完全 LU 分解付き BiCGStab 法， 疎行列用反復法，$\boldsymbol{x}^0 = \mathbf{0}$,
  反復停止基準: $\|\boldsymbol{b} - A\boldsymbol{x}^n\|_2/\|\boldsymbol{x}^n\|_2 < 10^{-10}$.
- DGESV: 部分ピボット選択付き LU 分解， 一般行列用直接法.

を用いました．表 13 に計算時間を，表 14 に誤差を示します.

4.6 課題とまとめ

限定されたテスト行列と解法による比較ながら，疎行列の直接解法の有効性が確認できました.

個人的な感触としては，現在のところ WSMP は

- 8 コア程度の並列数で
- 精度の要求される
- 数万〜数十万次元の問題

に特に有効ではないかと思われます.

また，行列の性質を非数値的に解析するという特徴を活かして，特異値・固有値解析への応用が期待されます．今後，新システムに導入される直接解法ソフトウェアの性能評価も含めて報告できればと考えています.

名称	次元数	非零要素数	WSMP	DGSF(S)	DSRIS	DGESV
west2021	2021	7353	0.03	**0.003**	—	0.12
fidap011	16614	1096948	**1.33**	625.88	—	37.71
e40r0100	17281	553562	**0.54**	51.55	—	39.30
memplus	17758	126150	0.38	**0.37**	1.25	44.75
raefsky3	21200	1488768	**1.38**	69.33	5.38	70.34
af23560	23560	484256	**1.28**	3044	—	102.27
wang3	26064	177168	1.41	603.93	**1.38**	129.75
lhr34c	35152	764014	**1.51**	17.32	—	317.53
onetone1	36057	341088	**1.89**	2.88	—	381.07
onetone2	36057	227628	0.85	**0.78**	—	386.70
bbmat	38744	1771722	**4.57**	—	—	441.22
av41092	41092	1683902	14.09	**13.41**	—	561.83
bayer01	57735	277774	1.09	**0.18**	—	1583.33
venkat50	62424	1717792	**1.51**	270.69	8.56	1859.18
epb3	84617	463625	1.77	20.70	**1.68**	—
twotone	120750	1224224	**4.22**	12.84	—	—
torso3	259156	4429042	35.45	344.66	**4.09**	—
languag	399130	1216334	1728	**0.68**	0.77	—
pre2	659033	5959282	**33.47**	—	—	—
hamrle3	1447360	5514242	**1200**	—	—	—

表 13: 非対称行列の計算時間

名称	次元数	非零要素数	WSMP	DGSF(S)	DSRIS	DGESV
west2021	2021	7353	**0.212E-08**	0.265E-07	—	0.448E-07
fidap011	16614	1096948	**0.139E-05**	*0.279E-01*	—	0.571E-04
e40r0100	17281	553562	**0.562E-11**	0.895E-06	—	0.338E-08
memplus	17758	126150	**0.133E-11**	0.103E-07	0.771E-05	0.207E-10
raefsky3	21200	1488768	**0.154E-12**	*0.286E-00*	0.542E-08	0.316E-08
af23560	23560	484256	**0.890E-13**	0.161E-04	—	0.119E-12
wang3	26064	177168	**0.448E-13**	0.416E-05	0.905E-08	0.253E-12
lhr34c	35152	764014	**0.186E-01**	*0.289E+03*	—	*0.105E+01*
onetone1	36057	341088	**0.117E-10**	0.749E-09	—	0.751E-09
onetone2	36057	227628	**0.744E-11**	0.980E-10	—	0.137E-09
bbmat	38744	1771722	**0.107E-10**	—	—	0.379E-08
av41092	41092	1683902	**0.264E-13**	0.717E-07	—	0.864E-12
bayer01	57735	277774	0.790E-06	*0.211E+03*	—	**0.757E-06**
venkat50	62424	1717792	**0.252E-12**	0.624E-05	0.248E-07	0.204E-11
epb3	84617	463625	**0.279E-13**	0.640E-08	0.381E-07	—
twotone	120750	1224224	**0.878E-10**	0.414E-08	—	—
torso3	259156	4429042	**0.444E-14**	0.561E-06	0.240E-07	—
languag	399130	1216334	**0.244E-14**	0.596E-11	0.179E-07	—
pre2	659033	5959282	**0.222E-04**	—	—	—
hamrle3	1447360	5514242	*0.200E-03*	—	—	—

表 14: 非対称行列の計算時間

参考文献

[1] P.R. Amestoy, T.A. Davis and I.S. Duff: An approximate minimum degree ordering algorithm, *SIAM J. Matrix Analysis and Applications*, **17**, 886–905 (1996).

[2] C. Ashcraft and J.W.H. Liu: Robust ordering of sparse matrices using multisection, *SIAM J. Matrix Analysis and Applications*, **19**, 816–832 (1998).

[3] T.A. Davis: *Direct Methods for Sparse Linear Systems*, Society for Industrial & Applied, 2006.

[4] I.S. Duff and J.K. Reid: The multifrontal solution of unsymmetric sets of linear equations, *SIAM J. Scientific and Statistical Computing*, **5**, 633–641 (1984).

[5] I.S. Duff, A.M. Erisman and J.K. Reid: *Discrete Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.

[6] A. George: Nested dissection of a regular finite-element mesh, *SIAM J. Numerical Analysis*, **10**, 345–363 (1973).

[7] A. George and J.W.H. Liu: *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[8] G. Karypis and V. Kumar: A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM J. Scientific Computing*, **20**, 359–392 (1999).
`http://glaros.dtc.umn.edu/gkhome/views/metis`

[9] J.W.H. Liu: Modification of the Minimum-Degree algorithm by multiple elimination, *ACM Transactions on Mathematical Software*, **11**, 141–153 (1985).

[10] J.W.H. Liu: The multifrontal method for sparse matrix solution: Theory and practice, *SIAM Review*, **34**, 82–109 (1992).

[11] 森 正武，杉原 正顯，室田 一雄: 線形計算, 岩波講座 応用数学, 岩波書店, 1994.

[12] 長谷川 秀彦，櫻井 鉄也，桧山 澄子，周 偉東，花田 孝郎，北川 高嗣: 数値計算法, オーム社, 1998.

[13] 小国 力，村田 健郎，三好 俊郎，J.J. ドンガラ，長谷川 秀彦: 行列計算ソフトウェア—WS, スーパーコン, 並列計算機—, 丸善, 1991.

[14] W.F. Tinney and J.W. Walker: Direct solutions of sparse network equations by optimally ordered triangular factorization, *Proc. IEEE*, **55**, 1801–1809 (1967).

[15] 戸川 隼人: マトリクスの数値計算, オーム社, 1971.

[16] M. Yannakakis, Computing the minimum fill-in is NP-complete, *SIAM J. Matrix Analysis and Applications*, **2**, 77-79 (1981).