社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－ARC－162 （18）
2005－HPC－101 （18）
2005／3／8

行列言語コンパイラ CMC の JDS 形式への対応と Matrix Market を用いた評価

湯之上　康一†　川端　英之†　北村　俊明†

MATLAB は数値計算コードを簡潔に記述できる言語および実行環境であり，広く利用されている．我々は，MATLAB コードの高速な実行環境の提供のため，MATLAB コードを静的解析により Fortran 90 記述に変換するコンパイラ CMC を開発している．CMC は疎行列計算にも対応しているが，これまでの実装では疎行列のデータ構造は CCS 形式や CRS 形式などしか扱えなかった．本稿では，CMC の JDS 形式への対応について述べる．Matrix Market を用いた行列ベクトル積の実測では，JDS 形式は他 CRS，CCS 形式と比べ，スカラ計算機 HPC2500 で最大 2.4 倍，ベクトル計算機 SX5 では最大 28 倍の高速化を確かめられた．

Support for JDS Format by the Matrix Language Compiler CMC and Its Evaluation using Matrix Market

Koichi Yunoue,† Hideyuki Kawabata† and Kitamura Toshiaki†

MATLAB is a language and an execution environment for matrix computations, which has been widely used. We have been developing CMC, a compiler for matrix computations, which translates MATLAB-based scripts into Fortran 90 programs by static analyses. CMC did not have the functionality to utilize sparse matrix schemes other than simple formats like CCS and CRS until the extension we show in this article was implemented. Newly supported data structure is called JDS format, which has been reported to be effective especially on vector machines. Experimental results show that the support of JDS format by CMC enables the translated codes of matrix-vector product to run 2.4 times and 28 times faster than the codes with CRS format, on HPC2500 and SX5, respectively.

1. はじめに

MATLAB は数値計算コードを簡潔に記述できる言語および実行環境である [1]．MATLAB は，行列演算をプリミティブとして持つこと，変数の型宣言が不要なこと，インタプリタ実行に基づき記憶管理が動的になされること，などの特徴を持っており，プロトタイピング言語として有効で，広く利用されている．行列積などの基本的な行列演算はチューニングされたライブラリによって処理されるので，簡単な行列計算コードであればインタプリタ環境での実行であっても比較的高速に実行でき，実用性も高い．

様々な MATLAB 記述が高速に実行できればよいのだが，MATLAB 実行環境は動的な型やデータ構造の変更を行なうので実行時のオーバヘッドは無視できない．これに対し，変数の型などを静的解析により決定して MATLAB 記述を Fortran などのコンパイル言語記述に変換し高速化を図る試みがあるが [2]，大規模数値計算に用いるためには不可欠な，疎行列データ構造への対応が行なわれたものは見られなかった．

これに対し我々は疎行列データ構造を扱える行列言語コンパイラ CMC （Compiler for Matrix Computations）を開発している [3),4]．CMC は，三角行列や対角行列などの行列の形状の詳細情報をプログラム記述から自動抽出し，それに応じた出力コードを生成できる [3]．また，複数種類の疎行列データ構造（CCS, CRS, および MD 形式）への対応により，扱うデータの非零要素配置やターゲットマシンのアーキテクチャに合わせた疎行列データ構造の選択によるチューニングを可能にした [4]．CMC を用いれば，繁雑になりがちな疎行列計算コードの開発を，汎用言語記述よりも保守性や可読性の高い行列言語記述で行なうことが可能になる．

しかしながら，比較的単純でどのような非零要素パターンの疎行列でも汎用的に利用可能な CCS 形式や CRS 形式はベクトル計算機による行列計算には向いていない [4]．一方，CCS 形式などと同様に任意の非零要素パターンの疎行列を効率良く格納でき，かつ行列ベクトル積をベクトル計算機上で高速に処理可能な

† 広島市立大学
Hiroshima City University

疎行列データ構造として知られる JDS 形式 [5),6)] は，構造が複雑なため Fortran などの汎用コンパイル言語では記述が容易とは言い難く，実際多くのライブラリではサポートされていない．疎行列を操作するユーティリティである SPARSKIT[7)] は JDS 形式での行列ベクトル積ルーチンや CRS 形式との相互変換ルーチンを持っているが，これを使いこなすにはユーザがデータ構造の細部を把握しておくことが不可欠である．

これに対し我々は，CMC を拡張し，JDS 形式の扱いを可能にした．これによりユーザは，JDS 形式の構造を意識することなく JDS 形式による高速処理の恩恵を受けることが可能となった．本稿では，CMC の JDS 形式への対応について述べる．また，スカラ，ベクトル，および疑似ベクトル機構付きの計算機における実測データに基づき，JDS 形式による行列ベクトル積の性能評価について述べる．実測には，疎行列データ集である Matrix Market[8)] を利用した．

以下，2 章は各種疎行列データ構造についてまとめ，3 章は各疎行列データ構造による行列ベクトル積演算について述べる．4 章では CMC の JDS 形式への対応についてのべる．5 章では，Matrix Market を用いた複数種類の計算機上での各疎行列データ構造の計算結果を示し，実測結果に基づく CMC の評価を行う．6 章はまとめである．

2. 疎行列データ構造

CMC では疎行列データ構造として CRS（Compressed Row storage）形式，CCS（Compressed Column storage）形式，および MD（Multi Diagonal）形式に対応している．これらに加え，新たに JDS（Jagged Diagonal Storage）形式をサポートした．ユーザは，MATLAB をベースとした同一のソースコード記述に対して，関数の疎行列引数がそれぞれどの構造であるかを示す指示を与えることにより，各疎行列データ構造での計算を行なう Fortran 90 コードを得ることができる．本章では各疎行列形式の構造について述べる [5)]．

2.1 CRS 形式

CRS 形式は，行列中の行中の非零要素を連続したメモリ上に配置する疎行列データ構造である．比較的単純で扱い易く，任意の非零要素配置の疎行列を扱うことが出来る．SparseBLAS 等の数値計算ライブラリにおいても使用されている．

ここで，式 (1) に示す行列 A （次元数 $n = 5$，非零要素数 $nz = 10$）を例にとり，CRS 形式のデータ構造を具体的に示す．

val	1	6	2	−1	3	−2	4	7	−3	5
colind	1	2	2	1	3	2	4	5	3	5

rowptr	1	3	4	6	9	11

図 1 式 (1) の行列 A を CRS 形式で格納した様子

val	1	−1	6	2	−2	3	−3	4	7	5
rowind	1	3	1	2	4	3	5	4	4	5

colptr	1	3	6	8	9	11

図 2 式 (1) の行列 A を CCS 形式で格納した様子

$$A = \begin{pmatrix} 1 & 6 & & & \\ & 2 & & & \\ -1 & & 3 & & \\ & -2 & & 4 & 7 \\ & & -3 & & 5 \end{pmatrix} \tag{1}$$

CRS 形式を用いた行列データの格納には，値を記憶するための浮動小数ベクトル `val(`nz`)` と，場所記憶のための整数ベクトル `colind(`nz`)`，`rowptr(`n+1`)` を用いる．`val()` には行列中の非零要素の値を行方向に走査して順に格納する．`colind()` には対応する `val()` の要素の列インデックスを格納する．`rowptr()` にはベクトル `val()` 上での行の開始位置を格納するが，便宜上 `rowptr(`n+1`)=`nz+1 とする．式 (1) の行列 A を CRS 形式で格納したものは**図 1** に示す通りとなる．

2.2 CCS 形式

CCS 形式は行列中の列中の比零要素を連続したメモリ上に配置する疎行列データ構造である．CCS 形式は CRS 形式における行方向の走査を列方向に変更したものであり，CCS 形式で行列 A を格納したものは，CRS 形式で行列 A^T を格納したものと一致する．

CCS 形式を用いた行列データ格納には，値を記憶するための浮動小数ベクトル `val(`nz`)` と，場所記憶のための整数ベクトル `rowind(`nz`)`，`colptr(`n+1`)` が必要となる．`val()` には行列中の非零要素の値を列方向に走査して順に格納する．`rowind()` には対応する `val()` の要素の行インデックスを格納する．`rowptr()` にはベクトル `val()` 上での列の開始位置を格納するが，便宜上 `colptr(`n+1`)=`nz+1 とする．式 (1) の行列 A を CCS 形式で格納したものを**図 2** に示す．

2.3 JDS 形式

疎行列を JDS 形式で格納する際には，まず行列中の非零要素を行方向に左詰めしたものを考える．このときの様子を**図 3** に示す．行ごとの要素数はそれぞれ異なるため，この状態でそのまま二次元配列に格納すると行列の行あたりの要素数のばらつきが大きい場合に無駄が多くなる．そこで JDS 形式では**図 4** のように行当たりの非零要素数で降順に並べ替えた後に列方向に走査して順に 1 次元配列に格納する．

ここで行列中の 1 行中の非零要素数の最大数を

1	6	
2		
−1	3	
−2	4	7
−3	5	

図 3 行列 A を左詰めした様子

−2	4	7
1	6	
−1	3	
−3	5	
2		

図 4 行を降順に並べ替えた様子

val	−2	1	−1	−3	2	4	6	3	5	7
colind	2	1	1	3	2	4	2	3	5	5

jdmax	3	perm	4	1	3	5	2

jdptr	1	6	10	11

図 5 式 (1) の行列 A を JDS 形式で格納した様子

jdmax としたとき，JDS 形式を用いた行列データ格納には値記憶のための浮動少数ベクトル val(nz) と，場所記憶のための整数ベクトル colind(nz)，perm(n)，jdptr(jdmax+1) が必要となる．val() には行列中の非零要素の値を前述の方法で順に格納する．colind() には対応する val() の列インデックスを格納する．jdptr() には図 4 における列の開始位置を格納するが，便宜上 jdptr(jdmax+1)=nz+1 とする．perm() には図 3 から図 4 への行の並べ替えを記憶しておく．式 (1) の行列 A を JDS 形式で格納したものを**図 5** に示す．

3. 疎行列データ構造を用いた行列ベクトル積

疎行列を用いた大規模数値計算においては巨大な疎行列とベクトルの積（あるいは疎行列と少数列の密行列の積）の計算が頻出する．そのため，疎行列データ構造の選択の際には行列ベクトル積の処理速度が一つの指標となる．本章では CMC で対応した疎行列構造である CRS， CCS， および JDS 形式について，行列 A とベクトル積 x の行列ベクトル積 $b = Ax$ の計算手順を比較し，各々の特徴についてまとめる．

3.1 CRS および CCS 形式の行列ベクトル積

疎行列を CRS 形式および CCS 形式で扱う場合の行列ベクトル積演算のコードを**図 6** および**図 7** に示す [5]．図中，疎行列 A は A_で始まる変数名のベクトルに，ベクトル x は x に，また行列 A の次元数 n は変数 n に，それぞれ格納されていると仮定している．

CRS 形式は行列中の非零要素を行方向に，CCS 形式は行列中の非零要素を列方向に連続して格納しているので，行列ベクトル積演算中の行列の走査もその方向に沿って行なうのが効率的である．両疎行列データ構造に関して，ベクトル行列積は 2 重のループ構造で記述される．内側ループの繰り返し回数は各行（もしくは列）の非零要素数で，また外側ループの繰り返し回数は行列の次元数 n となる．

```
 do Xk=1,n
   Xs=0
   do Xiptr=A_rowptr(Xk),A_rowptr(Xk+1)-1
     Xs=Xs+(A_val(Xiptr))
&           *(x(A_colind(Xiptr)))
   enddo
   b(Xk)=Xs
 enddo
```

図 6 CRS 形式の行列ベクトル積コード

```
 x=0
 do Xk=1,n
   do Xiptr=A_colptr(Xk),A_colptr(Xk+1)-1
     Xi=A_rowind(Xiptr)
     b(Xi)=b(Xi)+(A_val(Xiptr))*(x(Xk))
   enddo
 enddo
```

図 7 CCS 形式の行列ベクトル積コード

大規模疎行列においては一般に，行列中の 1 行あたりの非零要素数は僅かであり，疎行列の次元数が大きい場合であっても，図 6 および図 7 における内側ループの繰り返し回数は非常に少ないことが多い．内側ループはベクトル化可能であるが，そのベクトル長は短く，またループ交換も不可能であるため，結果として図 6 および図 7 のコードによる行列ベクトル積はベクトル処理には不向きであるといえる．

3.2 JDS 形式の行列ベクトル積

JDS 形式の行列ベクトル積演算のコードを**図 8** に示す．このコードは SPARSEKIT [7] における amuxj() と permvec() を合わせたものに相当する．

JDS 形式に関しても行列ベクトル積は 2 重ループ構造で記述される．内側ループの繰り返し回数は左詰めしたの後の各列の長さであり，この値は外側ループの反復ごとに変化するが，各行の非零要素数のばらつきが少ない場合にはほぼ n に近いものとなる．また外側ループの繰り返し回数は各行について非零要素数の最大値 jdmax となる．すなわち，一般の大規模疎行列における行列ベクトル積において JDS 形式を用いた場合には，内側ループの反復回数は非常に多く，外側ループの反復回数は僅かなものとなる．内側ループはベクトル化可能であることや，浮動小数点演算回数は疎行列データ構造の如何によらないことを考えると，JDS 形式による行列ベクトル積の計算は，CRS 形式や CCS 形式を用いた場合と比較してベクトル計算機での実行により適した方法であるといえる．

ところで，JDS 形式では行列中の行中の非零要素を左詰めにした後に列方向に沿って各要素が連続して格納されるので，記憶領域上で連続した位置関係にある要素どうしが行列における “隣り合った要素” どうしであるとは限らない．このため，最内側ループ中でメモリ上での連続的な要素参照を行なう場合には，得られたベクトルの各要素を perm() の参照によって置換

```
 allocate(Db(n))
 Db=0
 do Xk=1,jdmax
   tmp=A_jdptr(Xk)-1
   do Xiptr=A_jdptr(Xk),A_jdptr(Xk+1)-1
     Db(Xiptr-tmp)=Db(Xiptr-tmp)
&      +(A_val(Xiptr))*(x(A_colind(Xiptr)))
   enddo
 enddo
 do Xk=1,n
   b(A_perm(Xk))=Db(Xk)
 enddo
 deallocate(Db)
```

図 8 CMC に実装した JDS 形式の行列ベクトル積コード

しておく必要がある．

4. CMC の JDS 形式への対応

CMC は，MATLAB における関数を Fortran 90 のサブルーチンに変換する機能を持つ．変換に際しては，関数の引数の属性値についてはユーザから指示を受け，関数内で新規に値が定義される変数の属性値は自動的に決定する．各変数の属性値は，整数や実数などの基本データ型，行列／ベクトルのサイズ，形状，および（疎行列の場合の）データ構造の種類，の 4 つの項目から成る [3),4)]．

CMC において，CCS，CRS，および MD 形式への対応は，次のように行なわれてきた：

- 各疎行列データ構造について，ベクトルや密行列との積，および，同じ構造どうしの行列積を行なうライブラリルーチンをそれぞれ用意（任意の構造どうしの間での構造変換ルーチンも用意）．
- コード変換に際して，ソース中の行列演算の出現を適宜ライブラリルーチン呼び出し（あるいはそれをインライン化した表現）に置き換える．ただし異なる構造どうしの行列積が行なわれる場合には，一方を他方の構造に変換して同じ構造にしてから行列積を行なうようなコードを出力する．

JDS 形式への対応にあたっては，上と同様の手順は取らなかった．JDS 形式は，どのような非零要素配置の疎行列であっても行列ベクトル積を効率的に行なうことができるように設計されたデータ構造であるが，従来 CMC で対応していた疎行列データ構造とは異なり，行列の列方向，行方向，あるいは対角方向に沿った連続的な要素参照が困難な構造である．このため，JDS 形式どうしあるいは JDS 形式と他の疎行列の行列積を直接計算するのは効率的ではないと考えられる．そこで，このたび新たに JDS 形式に対応するにあたっては，次の方法をとった：

- JDS とベクトルあるいは密行列の積は，直接計算する．

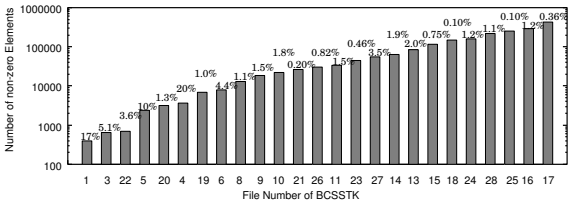

図 9 BCSSTK の各行列中の非零要素数と非零要素の占める割合

- JDS 形式と他の疎行列との積などの演算が行なわれる場合は，JDS 形式の疎行列を他方のデータ構造に変換して計算する．

なお，JDS 形式の疎行列を他の構造に変換する処理は，非零要素数に比例する回数のメモリ参照で行なえる．

5. 実測および評価

本章では，JDS 形式を用いた場合の計算性能の評価について，実測結果を踏まえて述べる．実測としては，複数種類の計算機環境において，疎行列データを CCS，CRS，JDS の各形式で扱った場合のそれぞれについて行列ベクトル積や CG 法による連立一次方程式の求解を行なった．

5.1 疎行列データ

実測を行なうにあたっては，疎行列データ集 Matrix Market[8)] からいくつかの疎行列を用いた．Matrix Market は構造解析や固有値分析などさまざまな分野で現れる疎行列を集めたものである．この度はその中から Harwell-Boeing Collection の一部，BCSSTK01〜28 を用いた．なお，No.2 は密行列であり，No.07 および 12 は それぞれ No.06 および 11 と同じ疎構造であるため，省いた．**図 9** に，用いた疎行列の非零要素数と行列中に占める非零要素数の割合を，非零要素数の昇順に各行列を並べた形で示す．横軸の数字は BCSSTK のファイルナンバー，縦軸は非零要素数である．各グラフ上の数字は，それぞれの行列中の非零要素の割合である．

5.2 計算機環境

実測に用いた計算機やソフトウェア環境は次の通り．

(1) スカラ計算機☆（以降，HPC）

- FUJITSU PRIMEPOWER HPC2500（SPARC64V，1 ノード主記憶 512GB）を 1 CPU で実行
- Fortran 90 コンパイラ: frt（Fujitsu Fortran Compiler Driver Version 5.6）
- frt オプション: `-Kfast_GP=3 -X9`

(2) 擬似ベクトル処理機構付計算機☆☆（以降，MPP）

- HITACHI SR8000/mpp （RISC マイクロプロ

☆ 京都大学学術情報メディアセンターのシステムを使用
☆☆ 東京大学情報基盤センターのシステムを使用

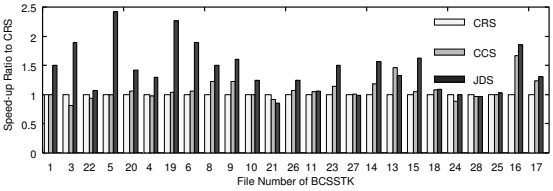

図 10 HPC 上での行列ベクトル積の速度比較

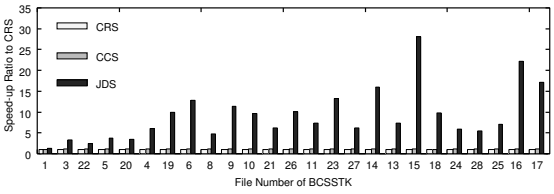

図 11 SX5 上での行列ベクトル積の速度比較

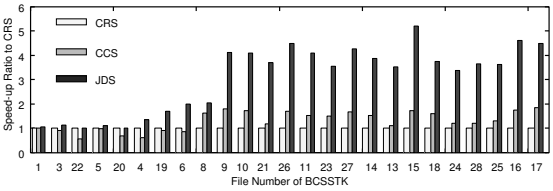

図 12 MPP 上での行列ベクトル積の速度比較

セッサ，1 ノード主記憶 16GB）を 1CPU で実行

- Fortran 90 コンパイラ: f90（最適化 FORTRAN90 V01-06-/B）
- f90 オプション: `-Os -pvec -noparallel`

(3) ベクトル計算機☆（以降，SX5）

- NEC SX-5 （ベクトルプロセッサ，1 ノード主記憶 128GB）を 1CPU で実行
- Fortran 90 コンパイラ: sxf90（FORTRAN90/SX Version 2.0 for SX-5，f90 driver and compiler for SX Rev.302）
- sxf90 オプション: `-C vopt`

計時は MPP ではサービスサブルーチン `xclock()` を，HPC 及び SX5 では標準ライブラリ関数 `gettimeofday()` をリンクして用いた．

5.3 行列ベクトル積

疎行列を CRS, CCS, JDS の各形式で扱った場合の行列ベクトル積の実行結果を**図 10**，**図 11**，および**図 12** に示す．各図において，横軸の数字は用いた疎行列の BCSSTK のファイルナンバーを表す（図 9 と同じ順序になるように左から右へ非零要素数で昇順に並べている）．それぞれの行列についての 3 本のバーは左から CRS, CCS, JDS 形式での実行結果を表す．縦軸は CRS 形式での実行速度に対する速度比を表している．

図 10 は HPC における各疎行列データ構造を用いての行列ベクトル積の実測結果である．ほとんどの行列において JDS 形式が最も優れていることがわかる．JDS 形式は CRS 形式に対して平均 1.4 倍，最大で 2.4 倍の実行速度であった．

図 11 は SX5 での行列ベクトル積の実測結果を示している．SX5 では全ての疎行列について JDS 形式が抜き出て良い結果となっている．CRS 形式に対して JDS 形式は平均 9.4 倍，最大 28.0 倍の実行速度であった．3.2 節で述べたように，JDS 形式による行列ベクトル積は CRS 形式や CCS 形式による場合と比較してループの最内側の反復回数が大きいが，これによりベクトルプロセッサの特性が発揮されたことが確認できる．

図 12 は MPP での行列ベクトル積の実測結果を示している．MPP でも全ての疎行列データについて JDS 形式を用いた場合が最も高速であった．CRS 形式での実行に対して JDS 形式では平均 3.1 倍，最大倍 5.2 倍，高速であった．MPP が搭載している擬似ベクトル処理機構はループ内の参照データを参照時までに浮動小数点レジスタにロード完了するための機構であり，これもやはり規則的な演算が行なわれるループの繰り返し回数が大きい場合に有効に働く．図 12 に示す結果は，ベクトルプロセッサの場合と同様，JDS 形式による行列ベクトル積は擬似ベクトル処理によって効率良く行なわれることを示している．

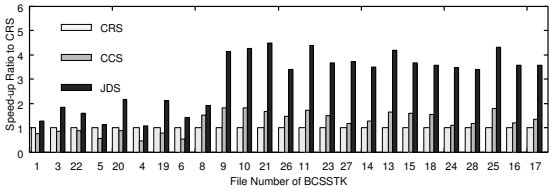

図 13 MPP 上での CG 法の速度比較

5.4 CG 法による連立一次方程式の求解

正定値対称で疎な係数行列を持つ連立一次方程式の代表的なソルバである CG 法においては，疎行列が絡む演算は行列ベクトル積のみで，その他はベクトルの内積などである．そのため，疎行列データ構造として JDS 形式を用いれば，CCS 形式や CRS 形式を用いる場合よりも高速に計算が行なえるものと期待できる．それを確認するために，CG 法のコードを MATLAB で記述し，疎行列のデータ構造を CRS，CCS，および JDS の各形式にすべく適切な指示行を与えて CMC で Fortran 90 に変換したコードを用いて実測を行なった．

MPP による各疎行列データ構造を用いた CG 法の実測結果を**図 13** に示す．縦軸，横軸は前節の行列ベクトル積の場合と同様である．MPP における行列ベクトル積の結果である図 12 と比較すると，ほぼ同様に JDS 形式を用いたものが最も高速であった．

☆ 大阪大学サイバーメディアセンターのシステムを使用

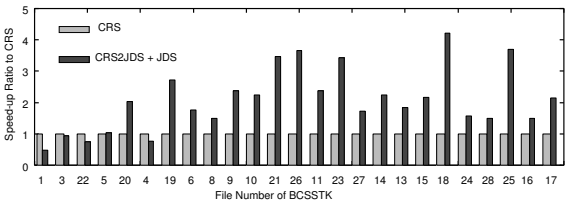

図 14 SX5 上でのデータ構造変換を含んだ行列ベクトル積との速度比較

5.5 JDS 形式へのデータ構造変換に要するコストについて

これまで述べてきたように，JDS 形式は行列ベクトル積を用いた計算において良い性能を示している．しかしながら，行列データの入出力や計算中における行列のデータ構造を全て JDS 形式に統一することは必ずしもできない．一般に利用されるデータ構造が CRS 形式や CCS 形式であることを踏まえれば，JDS 形式を用いるにあたっては，計算速度だけでなくデータ構造変換にかかるコストも考慮する必要がある．

図 14 に，SX5 における実測で “CRS 形式で行列ベクトル積を行なった場合” に対して “CRS 形式から JDS 形式へ変換した後に JDS 形式で行列ベクトル積を行なった場合” がどの程度高速化されたかを示す．前者は図中の “CRS” に，後者は “CRS2JDS + JDS” に，それぞれ対応しており，図の縦軸は後者が前者に対して何倍速かったかの比率を表す．図 14 より，計算実行に先んじてデータ構造変換を行なった場合でも，JDS 形式による行列ベクトル積の効率の良さのおかげで全体の計算処理に要する時間を短縮できるということが確認できる．しかしながら，非零要素数の少ない（行列の次元数自体が小さい）行列の場合には CRS 形式を用いたものよりも実行速度が低下している．すなわち要素数の少ない疎行列では効果的なベクトル処理を行えないためにデータ構造変換に要する時間が “隠せない” ことが分かる．CMC でのコード変換にあたっては，疎行列の大きさによって行列ベクトル積を行う部分の実行形態を（動的あるいは静的に）選択できるコードを生成するようにするなどの工夫が求められる．

6. ま と め

本稿では，MATLAB に基づく行列言語処理系である CMC の拡張，すなわち JDS 形式への対応について述べた．Matrix Market の疎行列データを用いた実測による比較では，JDS 形式を用いることにより，CRS 形式の場合よりも，スカラ計算機で最大 2.4 倍，ベクトル計算機で最大 28 倍，擬似ベクトル処理機構を持つ計算機で 5.2 倍，それぞれ高速な処理が確認された．特に，ベクトル計算機や擬似ベクトル処理マシンでは，実測に用いた全ての疎行列データに対して JDS 形式が最速であった．

疎行列データ構造の一種である JDS 形式は，広く用いられている CRS 形式や CCS 形式よりも複雑であるが，ベクトル計算機上での行列ベクトル積に有利である．JDS 形式に対する CMC でのサポートは，データ構造やアルゴリズムの詳細を意識することなく高速な大規模数値計算コード開発を可能にするという CMC の開発目的の実現のために有意義であると言える．

今後の課題としては，SuperLU[10] などの外部の行列計算ライブラリと CMC の連携機能の実装や，並列実行可能なコード生成の検討などが挙げられる．

謝辞 本研究の一部は広島市立大学特定研究費（一般研究費，課題番号 4111）の助成による．

参 考 文 献

1) http://www.mathworks.com/
2) De Rose, L., Padua, D.: Techniques for the translation of MATLAB programs into Fortran 90, *ACM Trans. Programming Languages and Systems*, Vol.21, No.2, pp.286–323 (1999).
3) 川端英之，鈴木睦: 疎行列に対応した行列言語コンパイラ CMC の開発，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG11(ACS7), pp.378–392, (2004).
4) Kawabata, H., Suzuki, M., and Kitamura, T.: A MATLAB-Based Code Generator for Sparse Matrix Computations, *Proc. Second Asian Symposium on Programming Languages and Systems (APLAS2004)*, *LNCS*, Vol.3302, pp.280–295, (2004).
5) Barrett, R., et al.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM (1994).
6) Saad, Y.: Krylov Subspace Methods on Supercomputers, *SIAM J. Sci. Stat. Comput.*, Vol.10, pp.1200–1232 (1989).
7) Saad, Y.: Sparsekit: a Basic Tool Kit for Sparse Matrix Computations, *Technical Report*, Computer Science Department, University of Minnesota (1994).
8) http://math.nist.gov/MatrixMarket/
9) Gilbert, J. R., Moler, C., Schreiber, R.: Sparse Matrices in MATLAB: Design and Implementation, *SIAM J. Matrix Anal. Appl.*, Vol.13, No.1, pp.333–356 (1992).
10) Demmel, J.W., Gilbert, J.R., and Li, X.S.: *SuperLU Users' Guide*, http://crd.lbl.gov/~xiaoye/SuperLU/ (2003).