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

2003－HPC－96 (6)
2003／10／16

MATLABに基づく疎行列計算向けコード生成

川端英之† 鈴木睦†

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．従来，MATLAB コードの実行速度向上の試みとして Fortran などのコンパイル言語記述への変換が行なわれているが，疎行列に対する取り組みは無かった．これに対し我々は，疎行列の扱いを可能にし，さらに対角，三角などの行列形状情報の検出機能を付加したコード変換手法を提案する．試作した処理系を SOR 法や CG 法の MATLAB コードに適用して実測したところ，係数が疎行列の大規模コードにおいて，SOR 法で約 7 倍，CG 法でも約 3 倍，MATLAB よりも高速に実行でき，MATLAB をベースとした大規模数値計算コード開発の可能性が確認できた．

Translation of MATLAB Scripts for Sparse Matrix Computations

HIDEYUKI KAWABATA† and MUTSUMI SUZUKI †

MATLAB is a language and an execution environment for matrix computations, which is used in wide area. There have been studies on translation of MATLAB scripts into programs written in compiled language such as Fortran, intended to speed up their execution. However, non of those systems seems to be able to handle sparsity of matrices attaining high-performance of translated code. We propose a new translator of MATLAB which utilizes sparse data structures and, in addition, shapes of matrices, e.g., triangular, diagonal, etc. In this paper, we present the design of CMC, the prototype of our compiler. Experimental results show that the translated SOR code by CMC runs seven times as fast as the original MATLAB code, which confirms our method's effectiveness.

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サポートしている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[2]，疎行列データ構造に対応したものは見られない．

大規模数値計算のためには，疎行列データ構造の使用が不可欠である．しかし疎行列データ構造を用いる汎用高級言語記述は繁雑になりがちで，保守性も低い．そこで我々は，MATLAB をコンパイル言語に変換するコンパイラを拡張し，疎行列を扱う機能および行列形状に応じた変換機能を付加することを提案する．

具体例として次の機能を持つコンパイラを試作した．

- MATLAB 記述を Fortran90 記述に変換する．
- 疎行列データ構造も利用可能である．
- 行列の形状情報をコード生成に利用する．
- 言語拡張はせず，ユーザ指示は MATLAB のコメント形式の指示行を用いる．

開発した処理系 (以降，CMC☆☆☆と呼ぶ) は，MATLAB 言語のサブセットに対応しており，基本的な行列演算を扱える．CMC を用いれば，プロトタイピング言語として優れる MATLAB を用いて疎行列コードを開発することが可能になる．

本稿では，CMC の実現方式について述べる．また，SOR 法および CG 法の MATLAB コードに CMC を適用した実測結果を示し，提案手法の評価を行なう．

† 広島市立大学情報科学部
Faculty of Information Sciences, Hiroshima City University

☆ MATLAB is a trademark of The MathWorks, Inc.

☆☆ MATLAB Ver.6 は LAPACK を用いている．

☆☆☆ "a Compiler for Matrix Computations."

```
input: A ∈ R^{n×n}, x ∈ R^n, tol ∈ R
output: λ ∈ R, i ∈ N
  i ← 0
  λ ← 0
  while(true) begin
    i ← i + 1
    y ← Ax
    λ_new ← (y^T y)/(y^T x)
    exit if |λ − λ_new| ≤ tol.
    x ← y/||y||_2
    λ ← λ_new
  end
```

(a) The power method.

```
function [l,i] = powermethod(A, x, tol)
i = 0;
l = 0;
while 1
  i = i + 1;
  y = A * x;
  lnew = (y' * y) / (y' * x);
  if abs(l - lnew) <= tol, break, end
  x = y / norm(y);
  l = lnew;
end
```

(b) A MATLAB srcipt of the power method.

図 1 MATLAB のコードの例
Fig. 1 An example of MATLAB coding.

2. MATLAB およびその高速化について

2.1 MATLAB コードの概要

MATLAB コードにおける変数は行列を基本としており (スカラは 1 行 1 列の行列と見なされる), 列ベクトルと行ベクトルは区別される. また MATLAB では行列の積や転置などの操作が基本演算として言語仕様に取り込まれているので数値計算コードの記述が簡潔に行なえる. MATLAB コードの例を図 1 に示す. 図 1(a) はべき乗法のアルゴリズムの一般的な表現であり, 図 1(b) はそれを MATLAB 言語により関数として記述したものである. 両者の類似性は明らかである.

図 1(b) のように, MATLAB では変数の型宣言などは不要で, 各演算子による計算処理内容はオペランドの型や形状 (ベクトルか行列かなどの区別) に基づき動的に決定される. 実際, 図 1(b) の関数 powermethod() を, 引数の A や x にスカラ値を与えて呼び出しても, 破綻なく計算される.

2.2 MATLAB における疎行列の扱い

MATLAB 実行環境には疎行列を扱うためのデータ構造も用意されている. 疎行列のデータ構造は, 行列の各列の非零要素を圧縮して行列全体を一次元配列により表現する方法で [3], Compressed Column Storage (CCS) 形式 [4] あるいは Compressed Sparse Column 形式 [5] と呼ばれるものと実質的に同等である. CCS

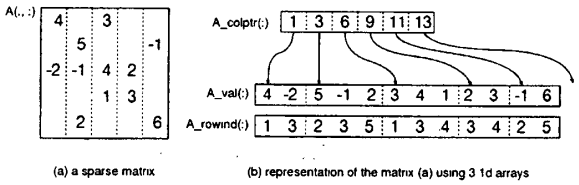

図 2 CCS 形式による疎行列の表現
Fig. 2 The structure of CCS form for sparse matrices.

形式は図 2 に示すように 3 本の一次元配列で一つの行列を表すもので, 疎行列の保持に要する記憶容量は非零要素の個数に比例する量で抑えられる.

MATLAB における行列計算コードの記述においては各行列が疎行列か否かを意識する必要はない. 例えば図 1(b) の関数は仮引数の行列 A に対応する実引数を疎行列にして呼び出しても, 動的な対処により適切に実行される.

2.3 MATLAB の高速化：コンパイル言語への変換

前述のように, MATLAB では各演算子による処理内容がオペランドの変数の型や形状, あるいはデータ構造に応じて動的に決められる. このインタプリタ実行に起因するオーバヘッドを削減するべく, Fortran90 などのコンパイル言語記述に変換して実行する方法が研究されている [2]. 実行時の環境の情報を取り込む just-in-time コンパイルの試みもある [6]. これらの方法は, MATLAB 記述中の各変数の型や形状を静的にあるいは実行直前に決定し, できるだけ動的処理を必要としないコードを生成しようとするものである.

しかし既存の手法には次のような欠点がある：

- 疎行列の扱いに対応していない.
- 行列の形状の推定において, その分類が粗く, 三角行列や対角行列を特別扱いしていない.

前者は手法の有効性を半減させる問題点であり, 後者も, 三角行列や対角行列が頻出するコードにおいてはその非効率性が露わになると考えられる.

3. CMC の開発

我々は, MATLAB コードをコンパイル言語に変換するにあたり, 疎行列データ構造の扱いを可能にし, 行列の形状をより詳細に分類することができるようにした処理系, CMC を試作した. CMC は MATLAB の関数記述ファイル (function M-file) を入力として読み込み, Fortran90 のサブルーチンに変換して出力する. 現状の CMC は, CCS 形式の疎行列データ構造を扱うことが出来る.

CMC による変換処理は, 次の二段階から成る.

(1) 変数の属性 (型, 形状, サイズ, 構造) の決定.
(2) Fortran90 によるコードの出力.

MATLAB では変数の属性は動的に決定されるが, Fortran90 に変換するにあたっては静的な情報が必要

となる．しかしプログラム中の各変数の属性をユーザに与えさせることにするとMATLABの「手軽さ」が損なわれるので，CMCが必要とするのは関数の引数に関する情報のみとした．

Fortran90のコード生成の際には，MATLABコードに現れている変数の宣言だけでなく，行列演算において必要となる作業領域も用意する．また計算処理の途中で適宜，作業領域の`allocate/deallocate`を行なう．

各々のステップについて，以下で説明する．

3.1 変数の属性の検出について

検出すべき変数の属性には次のものがある．

- **型**．論理型，整数型，実数型，複素数型の区別．
- **形状**．スカラ，列ベクトル，行ベクトル，行列，の区別．三角行列や対角行列などの形状も区別する．
- **サイズ**．行列やベクトルの次元数．
- **構造**．密行列か疎行列かの区別．

コード中の各変数出現および式中の中間変数の型や形状などの決定には，次の情報をソースとして用いる．

- ユーザからの指示．
- コード中の定数．
- 組み込み関数の返り値．例えば関数`length()`の返り値の型は整数型である．また関数`triu()`は，返り値が行列の場合の形状は下三角である．

以下，各属性の検出手順について述べる．なお型およびサイズの決定手順は文献2)の方法とほぼ同様である．

3.1.1 変数の型の決定

変数の型は，次の階層関係に基づいて決定する．

論理型 ≺ 整数型 ≺ 実数型 ≺ 複素数型

例えば整数型と実数型の変数の積は実数型とみなす．ただし，整数型どうしの除算結果は実数型にするなどの特別扱いが必要である．

3.1.2 変数の形状の決定

MATLABでは各演算子の計算結果の形状がオペランドによって決まる．そこで，各演算について，**表1**に示す規則を適用して演算結果の形状を決定する．

3.1.3 変数のサイズの決定

演算子とオペランドとから計算結果のサイズが決定できる．例えば式`C=A*B`における行列`C`のサイズは，`A`，`B`の一方がスカラなら他方のサイズと同じで，両方ともスカラでないなら(`A`の行数 ×`B`の列数)となる．

3.1.4 変数の構造の決定

ベクトルやスカラは疎行列データ構造にはせず，行列の場合のみ疎行列データ構造の利用を考慮する．次のようにオペランドの構造に基づいて演算結果の行列のデータ構造を決定する．

- 疎行列どうし/疎行列とスカラ：演算結果は疎行列．
- 密行列と疎行列/密行列どうし：演算結果は密行列．
- 組み込み関数の出力が行列の場合は，基本的には入力のデータ構造と同じにする．

表1 データの形状の変換規則
Table 1 Data structure translation rules.

(a) `A*B`の計算結果の形状

Aの形状	Bの形状						
	S	R	C	U	L	D	F
Scalar	S	R	C	U	L	D	F
Row vec.	R	—	S	R	R	R	R
Column vec.	C	F	—	—	—	—	—
Upper tri. mat.	U	—	C	U	F	U	F
Lower tri. mat.	L	—	C	F	L	L	F
Diagonal mat.	D	—	C	U	L	D	F
Full mat.	F	—	C	F	F	F	F

(b) `A+B`, `A-B`, `A./B`, `A.*B`の計算結果の形状

Aの形状	Bの形状						
	S	R	C	U	L	D	F
Scalar	S	R	C	U†	L†	D†	F
Row vec.	R	R	—	—	—	—	—
Column vec.	C	—	C	—	—	—	—
Upper tri. mat.	U†	—	—	U	F	U	F
Lower tri. mat.	L†	—	—	F	L	L	F
Diagonal mat.	D†	—	—	U	L	D	F
Full mat.	F	—	—	F	F	F	F

†：+/-の場合は，スカラ値がゼロでなければFにする．

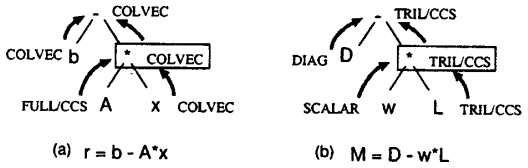

図3 式中の中間変数と左辺の型などの決定
Fig. 3 Type inference for intermediate variables

図3に，式中の中間変数の型や形状などの決定例を示す．図中，葉に相当する変数の情報は既に得られているものとする．図3(a)および(b)の式において，仮に変数`r`および`M`についてユーザ指示がなかった場合は，図に示すように`r`は列ベクトル，`M`は下三角の疎行列と決定される．なお図中の枠をつけた部分では，疎行列に対する計算の際には作業領域の確保が必要となる．

3.2 MATLABからFortran90への変換

Fortran90の演算子のいくつかは，MATLAB同様行列演算に対応しているため，密行列に対する計算に限っては単純な演算子の置換えでFortran90記述に変換できる．一方，疎行列の行列演算はCCS形式での計算に書きあらためる必要がある．

CMCによる変換例を**図4**に示す．図4(a)のMATLAB記述は，行列`A`が密行列か疎行列かによって図4(b)あるいは図4(c)のように変換される☆．

3.2.1 ユーザ指示行について

CMCへ入力するMATLABの関数記述では，仮

☆ 現状のCMCでは図4(c)のように変換するが，疎行列演算の場合もプリミティブをライブラリ化しておくこともできる．

```
r = b - A*x;
```

(a) A MATLAB code (A: matrix, r,b,x: column vectors).

```
r = b - MATMUL(A,x)
```

(b) Corresponding Fortran90 code with dense A.

```
allocate(Xr1(2500))
Xr1 = 0
do Xk = 1, 2500
  do Xiptr = A_colptr(Xk), A_colptr(Xk+1)-1
    Xi = A_rowind(Xiptr)
    Xr1(Xi) = Xr1(Xi) + (A_val(Xiptr)) * (x(Xk))
  enddo
enddo
r = (b) - (Xr1)
deallocate(Xr1)
```

(c) Corresponding Fortran90 code with sparse A using CCS format.

図 4 MATLAB から Fortran90 への変換例
Fig. 4 Examples of MATLAB to Fortran90 translation.

```
function [x,i] = sor0s(A,x0,b,tol)
%CMC integer,parameter ::  s=50*50
%CMC real*8,_CCS(5) ::  A(s,s)
%CMC real*8,_COLVEC ::  x0(s), b(s)
%CMC real*8,_SCALAR ::  tol
w = 1.8;
D = diag(diag(A)); L = -tril(A,-1); U = -triu(A,1);
r0 = norm(b-A*x0);
M = D-w*L; N = (1-w)*D+w*U; b2 = w*b;
x = x0; i = 0;
while 1
  i = i+1;
  x = M\(N*x+b2);
  if norm(b-A*x)/r0 <= tol, break, end
end
```

図 5 MATLAB で記述した SOR 法のコード
Fig. 5 The SOR method in MATLAB.

引数の属性値はユーザからの指示を受ける．図 5 に，SOR 法のコードに指示行を加えた例を示す．指示行は%CMC で始まる行で，記述は Fortran90 の変数属性指定に類似の形式である．

現状の CMC の実装で指定できる情報は次の通り：

- 型宣言：real*8, integer, logical.
- 形状指定：長方形行列 (_FULL；デフォルト)，行ベクトル (_ROWVEC)，列ベクトル (_COLVEC)，上三角行列 (_TRIU)，下三角行列 (_TRIL)，対角行列 (_DIAG)，スカラ (_SCALAR).
- 行列の構造指定：密行列 (_DENSE；デフォルト) あるいは疎行列 (_CCS)．疎行列であれば，各列の非零要素数の平均値 n を _CCS(n) のように指定.
- 行列やベクトルの (論理的な) サイズの指定.

次章では図 5 のコードを用いた実測データを示すが，ここで予めコード中の表現について述べておく：

- diag() は，引数が行列であれば対角要素を列ベクトルとして返す．引数がベクトルであればそれを対角要素として配置した対角行列を返す.
- tril(A), triu(A) はそれぞれ行列 A の下三角部分，上三角部分の取り出し，を行なう.
- y をベクトルとすると norm(y) は $||y||_2$ を返す.
- 演算子 '\' は「左からの除算」である．例えば x=A\b は連立一次方程式 $Ax = b$ の直接法による求解である．この計算に際し MATLAB は行列 A の形状を動的に検査し，下三角であれば前進代入，上三角であれば後退代入，密行列であれば LU 分解，などの処理を行なう.

3.2.2 変数の形状の詳細な分類の効果

CMC は対角行列や三角行列などの形状を解析する機能を有する．例えば図 5 中の M や N がそれぞれ下三角，上三角行列であることを判定できる．このような詳細な形状の把握には次の利点がある：

- 対角行列の値の保持は一次元配列で済ませられる.
- 三角行列や対角行列が演算のオペランドになっていれば，零要素の参照を省くことができる.
- 代入文の左辺変数が三角行列や対角行列であれば，零要素に帰する値のための計算を省ける.
- 演算子 '\' の左側に対角行列や三角行列があれば，MATLAB のような動的な形状検査 (コストは次元数 n に対して $O(n^2)$) を省き，前進代入や後退代入などを直接適用することができる.

以上のように，三角行列や対角行列の検出は記憶領域，計算量の削減に寄与し得る．ただし，三角行列への対処はそれが密行列でないと効果が少ないので，現状の CMC では，三角行列については密行列の計算の場合に限り形状情報をコード生成に反映させる.

3.2.3 疎行列用の領域について

疎行列を CCS 形式でメモリに保持する場合に，どの程度の容量を確保すればよいかは単純には決定できない．CMC では関数の引数が疎行列である場合は指示行による指定に頼り (3.2.1 節参照)，疎行列どうしの和/差の保持する記憶領域の大きさは，双方の疎行列の大きさの合計値にする．しかし疎行列どうしの積などは非零要素数を予め決定できないので，そのような場合は (作業領域用変数をユーザが明記した上で) _CCS(n) によりユーザが大きさを宣言するものとする.

なお MATLAB では，確保してある領域を越えた位置への代入が試みられた時点で動的に記憶領域が拡張される．ただしそのオーバヘッドは大きい.

3.2.4 変換後のコードの実行

CMC の出力は Fortran90 のサブルーチンであり，他の Fortran90 ルーチンなどとリンクして実行する．出力されるサブルーチンの仮引数は，MATLAB コードにおける仮引数と返り値のみである．なお，大規模疎行列のデータなどは，呼び出し元において用意しておくことを想定している．疎行列に指定した仮引数の行列は，CCS 形式に基づく 3 本の一次元配列で与える．図 6 に，図 5 を CMC で変換して得られるコードの先

表 2 SOR 法の実測結果
Table 2 Experimental results on the SOR method

問題	次元数/反復回数	MATLAB [sec] dense	MATLAB [sec] sparse	CMC/dense [sec] noshape	CMC/dense [sec] shape	CMC/sparse [sec] noshape	CMC/sparse [sec] shape
$Sx=u$ S : sparse	400/64	0.73(9.1)	0.08(1.0)	0.350(4.4)	0.272(3.4)	0.0105(0.13)	0.00964(0.12)
	900/81	8.91(42.)	0.21(1.0)	4.97(24.)	3.01(14.)	0.0311(0.15)	0.0289(0.14)
	2500/247	—	1.88(1.0)	—	—	0.274(0.15)	0.265(0.14)
	4900/472	—	7.23(1.0)	—	—	1.10(0.15)	1.04(0.14)
	8100/755	—	19.9(1.0)	—	—	3.22(0.16)	2.88(0.14)
$Fx=v$ F : dense	100/322	0.13(1.0)	0.49(3.8)	0.0962(0.74)	0.0729(0.56)	0.117(0.90)	0.115(0.88)
	200/1156	1.69(1.0)	6.71(4.0)	1.31(0.78)	0.951(0.56)	1.56(0.92)	1.56(0.92)
	300/2404	7.78(1.0)	34.8(4.5)	6.05(0.78)	4.30(0.55)	7.27(0.93)	7.12(0.92)
	400/4020	22.2(1.0)	141(6.4)	17.4(0.78)	12.7(0.57)	28.2(1.3)	25.4(1.1)

反復回数は行ごとの各実行について全て同じであった．括弧内の数値は各行における数値の相対比で，下線部は行ごとの最速値である．

```
SUBROUTINE sor0s(A_val,A_colptr,A_rowind,x0,b,tol,x,i)
IMPLICIT NONE
INTEGER,PARAMETER ::  s = 50*50
REAL*8 A_val(s*5)
INTEGER A_colptr(s+1), A_rowind(s*5)
...
```

図 6 図 5 に対して CMC が生成したコードの一部
Fig. 6 A part of the generated code from Fig.5 by CMC.

頭部分を示す．行列 A がベクトル A_val, A_colptr, A_rowind で表されている．

4. 実測と評価

開発した CMC の有効性の評価のために，SOR 法および CG 法による連立一次方程式の求解コードを MATLAB で記述し，実測した．

用いた計算機やソフトウェア環境は次の通り☆：

- 富士通 GP7000F (SPARC64-GP, 主記憶 24GB) を使用．SMP システムだが 1 CPU のみ使用．
- MATLAB のバージョン：6.5.0.180913a (R13)
- Fortran90 コンパイラ：frt (Fujitsu Fortran Compiler Driver Version 5.3 P-id: 912528-02)
- frt コンパイルオプション：-Kfast_GP=2 -X9

計時は，MATLAB では組み込み関数 cputime を利用した．Fortran90 コードの実行においては，標準ライブラリ関数 gettimeofday() をリンクして用いた．

数値実験で用いたデータは次の通りである．

- 密行列 F と列ベクトル v：
  $F=(f_{i,j})\in\mathbf{R}^{n\times n}$, $v\in\mathbf{R}^n$,
  $$f_{i,j}=\begin{cases}1 & (i=j)\\ -3^{-|i-j|} & (i\neq j)\end{cases}$$
  $v=(1,0,\ldots,0)^T$
- 疎行列 S と列ベクトル u：
  二次元正方領域において Laplace 方程式
  $\partial^2u/\partial x^2+\partial^2u/\partial y^2=0$
  を正方格子上の自然な順序付けの 5 点差分近似により解く際に現れる係数行列 S と，ディリクレ境界条件として一辺で $u=1$，他の 3 辺で $u=0$ としたときの右辺ベクトル u．

行列 F, S とも実正定値対称である☆☆．

4.1 SOR 法の実測結果

連立一次方程式 $Fx=v$ および $Sx=u$ を SOR 法により解いた．用いたコードは図 5 である．初期解はゼロとし，残差ノルムの比が $tol=10^{-6}$ 以下になった時点で反復を終了した．加速係数は 1.8 とした．

実測結果は**表 2** に示す．MATLAB は 2 通り，CMC は 4 通りの実行方法で実測した．表中，dense および sparse は，係数行列を保持するデータ構造を密行列 (二次元配列)，疎行列 (CCS 形式) のいずれにしたかの違いを表す．CMC については更に，対角行列や三角行列の形状を検出して出力コードに反映させた場合 (shape) と省略した場合 (noshape) について実測した．表 2 中，CMC/dense+noshape の欄の結果は従来手法のみにより得られるが，その他の右端 3 列分のデータは CMC の開発により新たに得られたものである．

4.1.1 問題 $Sx=u$ の場合

係数行列が疎の場合，dense はいずれも低速である．

CMC/sparse+shape は MATLAB/sparse より 7 倍以上高速である．CMC/sparse の noshape と shape との違いは，図 5 における対角行列 D を一次元配列で扱うか CCS 形式で扱うかの違いに過ぎないため，大きな差はない．

4.1.2 問題 $Fx=v$ の場合

sparse はいずれも低速である．MATLAB では特に，密行列を sparse で扱うオーバヘッドが顕著である．

CMC/dense+shape は CMC/dense+noshape よりも 1.4 倍程度高速である．CMC/dense における shape と noshape の差は，問題 $Sx=u$ の場合の CMC/sparse における shape と noshape の差よりも大きい．密行列計算では三角行列などの形状情報の把握の有無が実行速度に与える影響が大きいと言える．

4.2 CG 法の実測結果

CG 法により方程式 $Sx=u$ と $Fx=v$ を解いた結果

☆ 京都大学学術情報メディアセンターのシステムを使用．
☆☆ $Fx=v$ は実際は直接法で解く方が効率的である．

表 3 CG 法の実測結果
Table 3 Experimental results on the CG method

(a) the CG method: $Sx = u$ (S: sparse matrix)

次元数/反復	MATLAB/sparse [sec]	CMC/sparse [sec]
900/72	0.08(1.0)	0.0157(0.20)
2500/117	0.33(1.0)	0.102(0.31)
4900/162	0.92(1.0)	0.301(0.33)
8100/205	1.89(1.0)	0.685(0.36)
12100/249	3.73(1.0)	1.32(0.35)

(b) the CG method: $Fx = v$ (F: dense matrix)

次元数/反復	MATLAB/dense [sec]	CMC/dense [sec]
200/112	0.11(1.0)	0.0806(0.73)
400/214	0.80(1.0)	0.583(0.73)
600/316	2.79(1.0)	1.94(0.70)
800/417	7.42(1.0)	4.59(0.62)

を表 3 に示す．反復の停止基準や初期解は SOR 法の場合と同じにした．CG 法のコードでは shape/noshape による違いは現れなかった．表 3(b) は従来手法のみを用いて得られる結果であるが，参考のために示す．

CG 法は基本的な行列ベクトル演算から構成されるので MATLAB に都合のよいアプリケーションの一つであるが，表 3(a) より，CMC は MATLAB よりも約 3 倍高速であることが分かる．

MATLAB に対する CMC の高速化率は，係数行列が密行列の場合よりも疎行列の場合のほうが大きい．MATLAB における疎行列データ構造の扱いに伴うオーバヘッドの大きさが現れているといえる．

5. 関連研究

MATLAB を Fortran90 に変換する研究は 2) など多いが，疎行列の扱いへの対処が見られない．MATLAB の開発元による MCC (MATLAB Compiler)[1] は疎行列の扱いを含めて C 言語化できるが，動的処理はライブラリに委ねられ，基本的に高速化にはつながらない．これに対し，我々が開発した CMC は，MATLAB の有する記述性の高さを損なうこと無く，疎行列を扱う数値計算コードを生成できる．

関数型言語記述から疎行列コードを生成する試みがある [7]．また，Fortran で記述された密行列用コードを元にして疎行列コードを生成する研究もある [8),9)]．これらはいずれも疎行列コード記述の繁雑さを低減することが主眼であるが，それらのソースとなる関数型言語による記述や Fortran による密行列コード記述が，可読性や保守性に優れているかどうかは，意見が分かれるところだと思われる．自動変換系の実装の容易さも問題で，例えば Fortran の密行列コードから行列の転置操作を自動検出するのも容易ではないだろう．

疎行列を扱うプログラム開発をユーザが直接行なえるように，疎行列用のプリミティブの標準化も試みられている [5]．しかし多数の引数の指定を必要とするライブラリ呼び出しを直接記述することは容易ではないし，コードの可読性にも難点がある．一方，CMC は，これらのルーチンの呼び出しを含むコード生成に対応できるように拡張することもできると考えられる．

6. まとめ

本研究では，MATLAB で記述された行列計算コードを Fortran90 に変換して高速化する処理系 CMC を開発した．実測に基づき，結論として次が言える：

- 疎行列を扱う高速な数値計算コードの生成が可能であることが分かった．MATLAB に対し，SOR 法で約 7 倍，CG 法で約 3 倍，高速化できた．
- 行列の形状を詳細に把握することにより，密行列コードも高速化できることが分かった．SOR 法における実測では，1.4 倍高速化できた．

今後の課題としては，MATLAB との互換性の向上，様々なデータ構造への対応，などが挙げられる．

謝辞 本研究の遂行にあたり，広島市立大学津田孝夫名誉教授に有益な御助言を賜った．

参考文献

1) Mathworks Inc. homepage. www.mathworks.com.
2) De Rose, L., Padua, D.: Techniques for the translation of MATLPAB programs into Fortran 90, *ACM Trans. Programming Languages and Systems*, Vol.21, No.2, pp.286–323 (1999).
3) 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).
4) Barrett, R., et al.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM (1994).
5) Duff, I. S., et al.: Level 3 Basic Linear Algebra Subprograms for Sparse Matrices: A User-Level Interface, *ACM Trans. Math. Softw.*, Vol.23, No.3, pp.379–401 (1997).
6) Almasi, G., Padua, D.: MaJIC: Compiling MATLAB for Speed and Responsiveness, *Proc. PLDI'02*, pp.294–303 (2002).
7) Fitzpatrick, S., Clint, M., Kilpatrick, P.: The Automated Derivation of Sparse Implementations of Numerical Algorithms through Program Transformation, *Tech. Rep.* 1995/Apr-SF.MC.PLK, Dept. Comput. Sci., The Queen's University of Belfast (1995).
8) Bik, A.J.C., Wijshoff, H.A.G.: Compilation Techniques for Sparse Matrix Computations, *Proc. ICS'93*, pp.416–424 (1993).
9) Bik, A.J.C., et al.: The Automatic Generation of Sparse Primitives, *ACM Trans. Math. Softw.*, Vol.24, No.2, pp.190–225 (1998).