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

2004－HPC－100　(8)
2004／12／17

高速な MEX-File を生成できる MATLAB コンパイラ

川 端 英 之[†]　北 村 俊 明[†]

MATLAB コードの高速実行のために，プログラムをコンパイル言語記述に変換する手法がいくつか提案され，特に大規模計算に関してその有効性が確認されている．しかしながら，プログラムを Fortran 記述に変換して MATLAB インタプリタとは独立に実行することを仮定しているものがほとんどで，MATLAB 処理系の豊富な組み込み関数や描画ルーチンとの連係のためにはユーザは依然として繁雑な作業をする必要があった．これに対し我々は，開発中の MATLAB コンパイラ CMC の機能を拡張し，スタンドアロン実行用の Fortran コードだけでなく，MATLAB インタプリタから直接呼び出せる MEX-file を生成する機能を実装した．本稿では，我々の用いた MEX-file 生成方式について述べる．数値実験では，自動生成された MEX-file が Fortran コード単独での実行と同等な速度で実行できることが確認できた．

A MATLAB Compiler for Optimized MEX-File Generation

Hideyuki Kawabata[†] and Toshiaki Kitamura [†]

Compilation of MATLAB programs for speeding up the execution of them has been studied and recognized as a promising approach especially for large-scale computations. However, many of existing compilation systems have been constructed as tools for generating efficient executables which run independently of the MATLAB interpreter. Thus, the rich set of utilities of the MATLAB system, which includes data analysis tools and visualization tools, has remained difficult to use with the compiler-generated codes. In this article, we show the newly developed functionality of our compiler for MATLAB scripts. Our system, CMC, can automatically generate MEX-files, or programs which are able to invoke in the MATLAB's interactive environment. Experimental results of CG programs processed by CMC confirm that the system is also effective for MEX-file generation.

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，複雑なコード記述の場合の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はあったが[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 (a Compiler for Matrix Computations) を開発し，MATLAB をベースとした大規模数値計算コード記述の可能性を示した[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も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[5)]（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できる．しかしながら CMC は MATLAB 記述から Fortran 90 サブルーチンを生成する機能しか持っていなかったため，MATLAB 処理系の持つ各種の機能との連係は必ずし

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

も容易ではなかった．

MATLAB 処理系は豊富な組み込み関数や描画ルーチンを持っており，これらを利用したデータ処理の容易さはユーザにとっての MATLAB の魅力の一つである．CMC によって高速化したルーチンを MATLAB インタプリタから呼び出すことができれば，ユーザは MATLAB の機能全てを享受しつつ，かつ計算負荷の大きい部分のみを高速化するということが可能になり，これはユーザにとって極めて好都合だといえる．しかしながら，例えば MATLAB から Fortran 90 によるコードを呼び出すためには，インターフェースの整合やデータ構造変換などの多くの処理を行なうゲートウェイルーチンを記述することが必要となる．これは一般のユーザにとっては煩わしい作業である．

これに対し我々は， CMC の機能を拡張し，MATLAB 記述をコンパイル言語化して高速化するという従来からある機能に加え，コンパイル言語化したコードを MATLAB インタプリタから直接呼び出せるようにするためのインターフェースルーチンの自動生成機能を実装した．すなわち，CMC に MEX-file を構成するためのソースファイルを出力する機能を加えた．拡張された CMC により自動生成された MEX-file による CG 法の実測では，MATLAB インタプリタから呼び出された MEX-file が， Fortran コード単独での実行の場合と同等な速度で実行できることが確認された．

本稿では，CMC であらたに可能となった MEX-file 生成機能について述べる．以下，2 章では MATLAB システムについて簡単にまとめ，3 章で CMC の MEX-file 生成機能を具体的に述べる．4 章では実測に基づく評価を行なう．5 章はまとめである．

2. MATLAB システム : 言語とその実行方式

2.1 MATLAB インタプリタと M-file

MATLAB システムにおいてユーザが与えるソースプログラムは function M-file と呼ばれるテキストファイルである☆（以下，単に M-file と呼ぶ）．1 つの M-file は 1 つの関数から成っており，ユーザは適切な引数および返り値を受ける変数を指定してその関数を呼び出すことが出来る．MATLAB ではプログラムは基本的にインタプリタによって実行され，対話処理中あるいは他の関数記述の解釈実行中に，任意の関数を呼び出すことができる．

M-file 記述における変数は基本的には行列である．MATLAB では行列の積や転置などの操作が基本演算であり，数値計算コードの記述が簡潔に行なえる．M-file 記述の例を**図 1** に示す．図 1(a) はべき乗法の

☆ MATLAB 用の記述には script M-file と呼ばれる形式もあるが，実質的に同等の function M-file 形式に書き換えができるので，ここでは script M-file については考えない．

input: $A \in \mathbf{R}^{n\times n}$, $x \in \mathbf{R}^n$, $tol \in \mathbf{R}$
output: $\lambda \in \mathbf{R}$, $i \in \mathbf{N}$
$i \leftarrow 0$
$\lambda \leftarrow 0$
while(true) begin
  $i \leftarrow i+1$
  $y \leftarrow Ax$
  $\lambda_{\rm new} \leftarrow (y^T y)/(y^T x)$
  exit if $|\lambda - \lambda_{\rm new}| \le tol.$
  $x \leftarrow y/||y||_2$
  $\lambda \leftarrow \lambda_{\rm new}$
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) An M-file implementation of the power method

図 1 MATLAB のコードの例

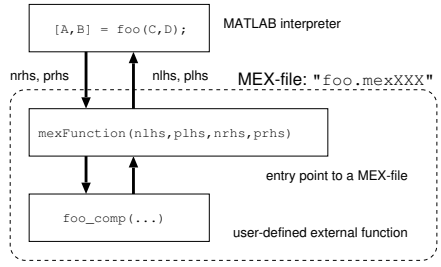

図 2 外部ルーチンの実行の様子

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 M-file 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，密か疎か，など）に基づき動的に決定される．実際，図 1(b) の関数を引数の `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 MEX-file 形式の利用 : C や Fortran を用いたコード記述

ユーザが MATLAB インタプリタから呼び出し可能な関数を用意する方法は，M-file 以外に，MEX-file 形式による方法がある．MEX-file 形式を用いれば，C や Fortran で記述したルーチンを MATLAB インタプリタから呼び出すことができるようになる．

図 2 は，MEX-file として与えられた外部ルーチンが MATLAB インタプリタから呼び出されている様

子の模式図である．図中の点線の枠で囲まれている部分が 1 つの MEX-file に相当する．MEX-file の構成上の制約は，エントリポイントが `mexFunction()` であること，およびその引数が予め定められた形式に従っている必要があるということであるが，通常は `mexFunction()` をゲートウェイルーチンとしてのみ用い，計算ルーチン（図中の `foo_comp()`）とは別に用意する．`mexFunction()` に含まれるべき処理は，MATLAB インタプリタから受け渡された引数を解析し，データ構造変換などを行なって適切な引数を用意して計算ルーチンを呼び出すこと，および，MATLAB インタプリタに返すべき引数の整形である．これらの処理を正しく行なう `mexFunction()` を記述することは，一般ユーザにとって必ずしも容易ではない．

なお，呼び出す側のコードは，呼び出されるルーチンがどのような形式で用意されているかを意識することなく記述できる．例えば図 1 (b) の M-file 中，関数 `abs()` や `norm()` が M-file で記述されたものか，MEX-file であるか，あるいは組み込みルーチンであるかによらず，実行できる．

2.3 M-file の MEX-file 化に関する得失

MATLAB インタプリタにおけるプログラムの実行では，行列データ構造の管理や演算実行において多くの動的処理がなされる．これによるオーバヘッドは，特に大規模疎行列計算コードなどの実行時には無視し難い[3),4)]．これに対し，ユーザがプログラムを記述する時点で静的に決定できることが多ければ，M-file でコードを記述する代わりに C や Fortran で記述して生成した MEX-file を用いることにより，動的処理の多くを排除することができ，高速処理が期待できる．

また，MEX-file を用いれば，MATLAB のデータ構造やライブラリに頼らずに計算ルーチンを記述できるという利点もある．例えば MATLAB の疎行列データ構造は CCS 形式である[6)]が，扱うデータの性質やアルゴリズムによっては他のデータ構造に変換して計算を行なう方が高速である場合がある[4)]．

なお，MEX-file 化が必ずしも高速化につながるとは限らない．MATLAB の開発元である MathWorks 社の提供する MATLAB Compiler (MCC) は，かつて，記述の容易な M-file を MEX-file に変換する機能を持っていた☆が，MCC による自動変換では全く高速化されない場合も多かった[2)~4)]．

3. CMC の拡張 : 高速な MEX-file を生成できるコンパイラの開発

既に述べたように，MATLAB プログラムのコンパイル言語化に基づく高速化手法は，MATLAB インタプリタから利用可能なままで高い性能を達成する簡単な方法が無かった．これに対して我々は，M-file から

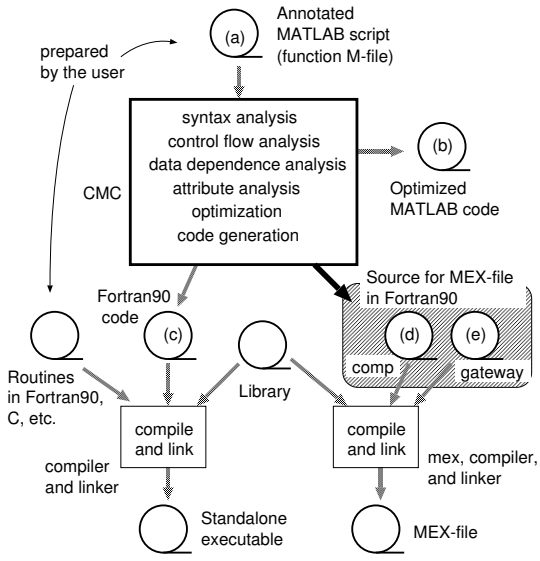

図 3 CMC による高速コードの生成

高速な Fortran 90 コードを生成できる行列言語コンパイラ CMC[3),4)] に拡張を加え，MEX-file を自動生成する機能を実装した．**図 3** に，拡張された CMC によるコード生成の流れを示す．CMC は，指示行付きの M-file（図 3 中の (a)）を入力として受け，ソースレベル最適化を適用した M-file (b)，Fortran 90 によるコード (c)，および，MEX-file 生成用のソースコード（図 3 中の網かけ部，(d) と (e)）を出力できる．図中の太い矢印で示される部分が新規に実装された部分である．

CMC が生成する MEX-file は，図 2 に示す通り，次の 2 つのコンポーネントから構成される：

- 計算ルーチン：図 3 中の (d)
- ゲートウェイルーチン：図 3 中の (e)

CMC によって生成された (d) および (e) のルーチンは Fortran 90 コンパイラおよび MathWorks 提供のコンパイラドライバ `mex` を用いて MEX-file に変換される．

上記それぞれのコンポーネントの生成手順について，**図 4** の M-file が入力として与えられた場合を例として用いつつ，以下の各節で述べる．

3.1 計算ルーチンの生成

CMC は，指示行付きの M-file （図 3 中の (a)）を Fortran 90 のサブルーチン（図 3 中の (c)）に変換する機能を既に持っており[3),4)]，これにより得られるサブルーチンは，MEX-file 用の計算ルーチン（図 3 中の (d)）とほぼ同等である．(c) と (d) の主な違いは，仮引数列および内部での変数宣言の方法の違いである．

図 4 の M-file において，CMC に対する指示行は `%cmc` で始まる行である．図 4 では，仮引数のうち `A` が

☆ MATLAB Compiler Ver.4 ではこの機能は省かれた．

```
 1:  function [x,i] = mycg(A,x0,b,tol)
 2:  %cmc integer,_auxarg ::  s
 3:  %cmc real*8,_ccs(5) ::  A(s,s)
 4:  %cmc real*8,_colvec ::  x0(s), b(s)
 5:  %cmc real*8,_scalar ::  tol
 6:
 7:  r = b-A*x0; rn = norm(r);
 8:  x = x0; p = r; i = 0;
 9:  tmpx00 = A * r;
10:  tmps01 = r' * r;
11:  while 1
12:    i = i + 1;
13:    alpha = tmps01 / (p' * tmpx00);
14:    x = x + alpha * p;
15:    rnew = r - alpha * tmpx00;
16:    if norm(rnew) / rn <= tol, break, end
17:    tmps02 = rnew' * rnew;
18:    beta = tmps02 / tmps01;
19:    p = rnew + beta * p;
20:    r = rnew;
21:    tmpx00 = A * p;
22:    tmps01 = tmps02;
23:  end
```

図 4 指示行付きの M-file （CG 法）

```
 1:  subroutine mycg(A_val, A_colptr, A_rowind,
 2: &                x0, b, tol, x, i, s)
 3:  implicit none
 4:  integer i, s
 5:  real*8 tol
 6:  real*8 A_val((s)*5)
 7:  integer A_colptr((s)+1), A_rowind((s)*5)
 8:  real*8 x0(s), b(s), x(s)
 9:   ...
10:  real*8 r(s)
11:   ...
```

図 5 生成される Fortran 90 ルーチン（スタンドアロン用）

行列，x0，b が列ベクトル，tol がスカラであることが指示されている．図 4 の 2 行目は，s が MATLAB のコード中には現れない補助変数であることを示している．補助変数は Fortran 90 コード生成時に仮引数の数を抑えてユーザの便宜を図るためのもので，スタンドアロン実行用のコード（図 3 の (c)）では**図 5** のようになる．一方，MEX-file 用の計算ルーチン（図 3 の (d)）の場合には，ゲートウェイルーチンの内部からの計算ルーチンの呼び出しをユーザが直接記述する必要がないので多少繁雑でも構わないし，引数の各行列のサイズを動的に変更することにも対応し易い☆ので，**図 6** に示すように，引数の行列の次元数は単純に各行列ごとに別々に渡すようにしている．

3.2 ゲートウェイルーチンの生成

図 4 の M-file の入力に対応するゲートウェイルーチン mexFunction() の出力例を**図 7** に示す．mexFunction() の構造は定型的であるので，図 7 を具体例として参照しつつ生成手順を説明する．

☆ CMC では，行列の次元数を補助変数 s1，s2，··· の関数，すなわち f(s1, s2, ···) の形で記述できる．しかし s1，s2，··· は mexFunction() の内部で動的に決定できるとは限らないので，次元数をそのまま引数として渡す方法を採用した．

```
 1:  subroutine mycg(A_val, A_colptr, A_rowind,
 2: &       A_d1, A_d2, A_nzmax,
 3: &       x0, x0_d1, b, b_d1, tol, x, x_d1, i)
 4:  implicit none
 5:  integer i
 6:  real*8 tol
 7:   ...
 8:  real*8 A_val(A_nzmax)
 9:  integer A_colptr((A_d2)+1), A_rowind(A_nzmax)
10:  integer A_d1, A_d2, A_nzmax
11:  real*8 x0(x0_d1), b(b_d1), x(x0_d1)
12:  integer x0_d1, b_d1, x_d1
13:   ...
14:  real*8 r(b_d1)
15:   ...
```

図 6 生成される Fortran 90 ルーチン（MEX-file 用）

表 1 mexFunction() の引数

nlhs	MEX-file から MATLAB システムに返される変数の個数
plhs(*)	MEX-file から MATLAB システムに返される変数へのポインタ配列
nrhs	MATLAB システムが MEX-file へ渡す引数の個数
prhs(*)	MATLAB システムが MEX-file へ渡す引数の変数へのポインタ配列

- mexFunction() の引数（図 7 の 1 行目）は，**表 1** に示す通りで，その個数および型は予め決められている．例えば MEX-file が [x,i]=mycg(M,v,f,t); という記述で MATLAB インタプリタから呼ばれたとすると，スカラ変数 nlhs と nrhs はそれぞれ 2，4 が入り，配列 prhs には変数 M，v，f，および t のデータを指すポインタが入れられ，また配列 plhs には変数 x と i に対応するデータへのポインタを入れるべき配列へのアドレスが入れられた状態で，制御が mexFunction() に渡される☆☆．
- 図 7 の 4∼6 行目は，mexFunction() 内部で使用する MATLAB のライブラリルーチン（データ構造操作のためのルーチン群）の型宣言である．
- 図 7 の 7∼16 行目は，計算ルーチンを呼び出すための実引数構成用の変数群の宣言である．準備が必要な変数の情報は，ソースである M-file （図 3 中の (a)）から静的に得ることができる．
- 図 7 の 19∼44 行目では，計算ルーチンを呼び出すための実引数の準備を行なっている．例えば第 1 引数 A （図 4 参照）は CCS 形式の実疎行列であるので，ポインタ prhs(1) によって指される MATLAB のデータ構造から 6 つの変数（A_val_p，A_colptr_shftd，A_rowind_shftd，A_d1，A_d2，

☆☆ ここでは簡単のためエラーチェックには触れない．本来ならば，mexFunction() の内部ではまず最初に引数の行列の型やサイズなどが計算ルーチンで想定しているものと一致するか否かを確認するべきである．

```
 1:  subroutine mexFunction(nlhs, plhs, nrhs, prhs)
 2:  implicit none
 3:  integer plhs(*), prhs(*), nlhs, nrhs
 4:  integer mxGetPr, mxGetM, mxGetN
 5:  integer mxCreateDoubleMatrix, mxGetJc, mxGetIr, mxGetNzmax
 6:  real*8 mxGetScalar
 7:  integer A_val_p
 8:  integer A_colptr_p, A_rowind_p
 9:  integer,dimension(:),allocatable ::  A_colptr_shftd
10:  integer,dimension(:),allocatable ::  A_rowind_shftd
11:  integer A_d1, A_d2, A_nzmax
12:  integer x0_p, x0_d1
13:  integer b_p, b_d1
14:  real*8 tol
15:  integer x_p, x_d1
16:  integer i, i_p, i_real
17:
18:* A
19:  A_val_p = mxGetPr(prhs(1))
20:  A_colptr_p = mxGetJc(prhs(1))
21:  A_rowind_p = mxGetIr(prhs(1))
22:  A_d1 = mxGetM(prhs(1))
23:  A_d2 = mxGetN(prhs(1))
24:  A_nzmax = mxGetNzmax(prhs(1))
25:  allocate(A_colptr_shftd(A_d2+1))
26:  allocate(A_rowind_shftd(A_nzmax))
27:  call mxCopyPtrToInteger4(A_colptr_p, A_colptr_shftd, A_d2+1)
28:  call mxCopyPtrToInteger4(A_rowind_p, A_rowind_shftd, A_nzmax)
29:  A_colptr_shftd = A_colptr_shftd + 1
30:  A_rowind_shftd = A_rowind_shftd + 1
31:* x0
32:  x0_p = mxGetPr(prhs(2))
33:  x0_d1 = mxGetM(prhs(2))
34:* b
35:  b_p = mxGetPr(prhs(3))
36:  b_d1 = mxGetM(prhs(3))
37:* tol
38:  tol = mxGetScalar(prhs(4))
39:* x
40:  plhs(1) = mxCreateDoubleMatrix(x0_d1, 1, 0)
41:  x_p = mxGetPr(plhs(1))
42:* i
43:  plhs(2) = mxCreateDoubleMatrix(1, 1, 0)
44:  i_p = mxGetPr(plhs(2))
45:* computation
46:  call mycg(%val(A_val_p), A_colptr_shftd, A_rowind_shftd,
47: &   A_d1, A_d2, A_nzmax, %val(x0_p), x0_d1, %val(b_p), b_d1,
48: &   tol, %val(x_p) , x_d1, i)
49:* return
50:  i_real = i
51:  call mxCopyReal8ToPtr(i_real, i_p, 1)
52:  return
53:  end
```

図 7 `mexFunction()` の出力例

`A_nzmax`）を抽出する必要がある（図 7 の 19〜30 行目参照）．なお，MATLAB 内での CCS 形式におけるインデックスベクトルは 0 を先頭と見なす値で保持される．これを現状の CMC が生成する計算ルーチンでの扱いに合わせるために，図 7 の 29〜30 行目では先頭のインデックスを 1 と見なすように値をシフトしている．

- 図 7 の 46〜48 行目では，図 6 に示す計算ルーチン `mycg()` を呼び出している．ここで，実引数を修飾している`%val()` は，Fortran においてサブルーチンに引数変数のアドレスではなく値自身を渡すための記述で，多くの Fortran 処理系で実装されている．この機能により，ポインタ変数の扱いが可能になる☆．
- 図 7 の 50〜51 行目では，返り値のうち整数であるものを実数に変換している．MATLAB における通常の変数の型は実数であるため，計算ルーチンとの間で引数を介して整数値の授受を行なう場合は，このような型変換処理が必要となる．

☆ 図 7 ではポインタ変数の型は integer としている．

3.3 MCC と CMC：MEX-file 生成機能に関する比較

MathWorks 提供の MATLAB Compiler (MCC) は，ほとんど全ての M-file 記述を，機能を損なうことなく MEX-file に変換することができた．しかしながら，MCC の最適化機能は基本的なスカラ演算に限定されていたため，コードの実行速度はほとんど改善されず，M-file の構文解析時の僅かなロスが省かれる程度であった．しかもこのロスは，MATLAB 7 で実装された JIT Accelerator により無視できるまでに削減され，結果として，MCC を用いて M-file から MEX-file を生成することは，高速化の観点からはほとんど無意味になった．

一方，CMC で MEX-file を生成する機能は，基本的にユーザ指示行に頼る specialization による最適化をベースとするコード生成機能に基づいており，M-file の引数の型に制約を加える代わりに，高速化を実現するものである．この方法は，M-file で記述する関数の引数についてユーザ指示行を必要とするが，本来関数利用者が知っていなければならない情報をコードに明記することを求めるだけであるし，特に MEX-file 生成の場合は行列の次元数や疎行列の非零要素数の見積り値を静的に把握する必要もないので，ユーザに多大な負担を課すものではない．加えて言えば，CMC の要求する指示行は，関数のインターフェースを明確化してコードの可読性や保守性を高める効果もあると考えられる．

4. 実測および評価

本章では開発した処理系を用いた実測結果を示す．

4.1 測 定 環 境

用いたアプリケーションは CG 法による連立一次方程式の求解コードで，図 4 に示す通りである．連立一次方程式の係数行列は二次元正方領域において Laplace 方程式 $\nabla^2 P = 0$ 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実正定値対称疎行列を用いた．右辺ベクトルは，ディリクレ境界条件として一辺で $P = 1$，他の 3 辺で $P = 0$ とする場合に対応するものとした．

実測では，次のそれぞれの実行時間を測定した：

- MATLAB インタプリタのみによる実行
- CMC により自動生成した MEX-file による実行
- CMC により自動生成した Fortran 90 コードの単独実行

実験は以下の 2 種類の計算機上で行なった☆☆．両システムはいずれも SPARC64 ベースだが，演算速度や使用ソフトウェアのバージョンが異なっている．

計算機 1 富士通 HPC2500 (SPARC64V)．SMP クラスタだが 1CPU のみを使用．

☆☆ 京都大学学術情報メディアセンターのシステムを使用．

表 2 計算機 1 における CG 法の実行時間 [秒]

未知数の数	MATLAB	MEX	単独 1	単独 2
70×70	0.100	0.0410	0.0426	0.0469
90×90	0.220	0.0851	0.0889	0.0991
110×110	0.420	0.167	0.169	0.179
130×130	0.860	0.573	0.567	0.297
150×150	2.20	1.63	1.23	1.12
170×170	3.62	2.66	2.20	1.75

MEX, 単独 1: `f90` を使用／単独 2: `frt` を使用

表 3 計算機 2 における CG 法の実行時間 [秒]

未知数の数	MATLAB	MEX	単独 1	単独 2
70×70	0.57	0.257	0.255	0.175
90×90	1.22	0.571	0.561	0.397
110×110	2.27	1.10	1.10	0.737
130×130	3.73	1.88	1.87	1.22
150×150	6.02	2.81	2.87	1.92

MEX, 単独 1: `f90` を使用／単独 2: `frt` を使用

- MATLAB : 7.0.0.1990 R14
- Fortran 90 コンパイラ／オプション :
  - `frt` (Fujitsu Fortran Compiler Driver Version 5.6) ／`-Kfast_GP=3 -X9`
  - `f90` (Sun WorkShop 6 update 2) ／`-fast`

計算機 2 富士通 GP7000F (SPARC64-GP) . SMP システムだが 1 CPU のみ使用.

- MATLAB : 6.5.1.199709 R13 (SP1)
- Fortran 90 コンパイラ／オプション :
  - `frt` (Fujitsu Fortran Compiler Driver Version 5.5) ／`-Kfast_GP=3 -X9`
  - `f90` (Sun WorkShop 5.0) ／`-fast`

計時は MATLAB では組み込み関数 `cputime` を利用した. MEX-file および単独実行においては，標準ライブラリ関数 `gettimeofday()` をリンクして用いた.

4.2 実 測 結 果

実測結果を**表 2** および **表 3** に示す. 表 2 および表 3 はそれぞれ計算機 1 および計算機 2 での実測結果である. MEX-file のコンパイルは富士通の `frt` では行なえなかったため，Sun の `f90` のみを用いた. 両表中，“単独 1” および “単独 2” は，それぞれ用いた Fortran 90 コンパイラが `frt` および `f90` の場合の結果である. 表の各行が，同一の未知数の数（すなわち問題規模）に対する実行時間を表す.

両表とも，“MEX” および “単独 1” の実行結果はほぼ等しい. すなわち，CG 法の実行に要する時間が，ルーチンを MATLAB インタプリタ内部から呼び出した場合とスタンドアロンで実行した場合でほとんど変わらないと言える☆. また “MEX” および “単独 1” の実行速度は，MATLAB インタプリタによる実行速度よりも 2 倍前後高速である. この数値は以前報告した値[3),4)] よりも低いが，用いた Fortran コンパイラの違いに起因するものと考えられる. 実際，両表中の “単独 1” と “単独 2” 結果からも，状況によっては大きな差が見られることが確認できる.

☆ “MEX” および “単独” での実行では，計算ルーチンの内容は同じでも，アロケートされる疎行列の大きさなどが異なる. そのため両者の実行速度が等しくなるとは限らない.

5. ま と め

本稿では，行列言語コンパイラ CMC の MEX-file 生成機能について述べた. 本機能の実現により，MATLAB コード（function M-file）の実行の高速化手段として，Fortran 90 のサブルーチンに変換して別に用意したルーチンとリンクして実行する方法の他に，高速な MEX-file を生成して MATLAB インタプリタから直接呼び出す方法が新たに加えられた. 高速な MEX-file の生成機能は，MATLAB システムの豊富な組み込み関数によるデータ処理と高速な大規模数値計算の実行とのシームレスな接続を可能とし，ユーザにとっての MATLAB システムの使い易さの向上に大きく貢献するものと考えられる. 数値実験では，CMC を用いて生成した MEX-file を MATLAB インタプリタから呼び出した場合と，全て Fortran 90 で実行ファイルを構成した単独実行とで，実行速度に差は見られず，いずれの場合も MATLAB インタプリタでの実行よりも 2 倍前後高速化できた.

今後の課題としては，より多くのアプリケーションに対する CMC の適用と性能評価が挙げられる.

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, Oct. 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, Nov. 2004.
5) Barrett, R., et al.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, 1994.
6) 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.