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

2004－HPC－99　(44)
2004／8／1

MATLABからC言語への変換における変数の動的解析削減手法

内藤祐美† 佐田宏史† 前川仁孝†
伊與田光宏† 宮崎収兄†

本稿では，MATLAB 言語の変数の属性情報を利用して C に変換し，実行時の変数の属性解析を削減する手法を提案する．MATLAB はインタプリタ形式で実行するため処理が遅い．そのため MALTAB 言語からオペランドの属性を用いて演算結果部の属性を解析し，Fortran に変換する手法が提案されている．従来の解析手法に加えて，使用された変数を再定義する逆依存関係がループ中に存在した場合，その構造を利用することや，各属性のうち 1 つの属性が分かれば他の 2 つの変数の属性を解析することができる．これにより変数の属性解析が削減されるので実行時の処理時間が短縮できる．

The Reduction Method of Dynamic Analysis of Variables in MATLAB to C Translator

Yumi naitou,† Hiroshi sata,† Yoshitaka maekawa,†
Mituhiro Iyoda† and Nobuyoshi Miyazaki†

This paper proposes a analysis method to reduce the run-time analysis of variables by using information of the MATLAB's variables. MATLAB is slow because of interpreter's execution. There is a research which translates MATLAB Language to FORTRAN Language by the analysis of MATLAB's attributes for improving processing time. Moreover, when anti-dependence exists in a loop, the proposed method decides attributes by any other two variables. Therefore, the proposed method reduces the processing time by elimination of attribute analysis at execution time.

1. はじめに

現在，シミュレーションなどを行う際のプロトタイピング言語として，開発期間を短縮するために C や Fortran90 などの高級言語よりも簡単に記述することができる MATLAB が広く利用されている．MATLAB はインタプリタ環境で処理を行うので，実数や複素数などの型の宣言を行う必要は無いが，実行時に型を決定するオーバヘッドがある．また，MATLAB は，扱う変数の形状やサイズが異なる場合においても，同一の変数名で使用できるので，入力されるデータによって変数の形状やサイズを変更する必要がある．このため，MATLAB は C や Fortran90 と比べて処理時間が非常に長くなる．

また，MATLAB コンパイラを利用して，MATLAB 言語を C に変換し，コンパイルしてから実行することで，インタプリタ環境で実行するよりも高速に処理できる．しかし，MATLAB コンパイラを使用して生成した C の変数は，

二次元配列の複素数で宣言される．そのため，変数に代入されるデータがスカラの実数などのとき，本来必要の無い無駄な演算をすることになる．この問題を解決するために，MATLAB プログラムの変数の属性を解析し，Fortran90 のコードに変換する手法が提案されている[1)2)]．本手法は Fortran90 のコードを生成する前に MATLAB 言語の変数の属性を解析することで実行時の変数を解析のオーバヘッドを削減する．この解析手法では 2 つのオペランドを用いて変数を解析する．しかし，オペランドの属性が両方とも分からないと属性を解析することができない．

また，行列が疎であるという情報を変数の属性に加えることで，疎行列構造の演算を扱うことができるようにする手法が提案されている[3)4)]．本手法は，行列が疎の場合は，ユーザが変数に疎であるという情報を持たせることで，MATLAB プログラムを CCS 形式などの疎行列に適したデータ構造を使用した Fortran90 のコードに変換することができる．しかし，ユーザ自身がデータ構造の情報を入力するため，変数の構造を考慮せずにプログラミング可能な MATLAB に比べ，

† 千葉工業大学 情報工学科
Department of Computer Science, Chiba Institute of Technology

ユーザにかかる負担は大きい．

そこで本稿では，静的解析により解析できず動的解析をしていた MATLAB の変数を，解析規則を拡張することで静的に解析できるようにして，処理を高速化する手法を提案する．本手法は，MATLAB 言語から高級言語に変換する際，オペランドの属性の情報から演算結果部の属性を決定する従来の手法に加えて，ループ中の変数の逆依存関係を利用することで，変数の一部を静的解析できるようにし，実行時に必要な変数の属性の解析を削減する．さらに，変数の属性の候補を持つことによって，従来の静的解析規則を拡張する．

以下本稿では，2 章で MATLAB プログラムを Fortran90 のコードに変換するために変数を解析する従来手法について述べる．3 章では，提案手法であるループの逆依存関係を利用する解析方法，および変数の属性を従来よりも細かく解析する静的解析手法について述べる．4 章では提案手法を評価する．

2. MATLAB プログラムから Fortran90 のコードへの変換

MATLAB の演算では，変数は行列演算を基本とするので，スカラも 1 行 1 列の行列と見なして処理する．MATLAB は，変数の型の宣言などは不要であるが，オペランドの型や形状，オペレータによって動的に演算結果部の属性を決定するため，オーバヘッドが大きく処理時間が長くなる．MATLAB コンパイラを使用して，MATLAB 言語を C に変換してから実行することで，インタプリタ環境より高速に処理することができる．しかし，このコンパイラを使用して生成した C コードの変数は，基本的に全てが二次元配列の複素数で宣言される．このため，実数部と虚数部の演算をすることになるので，入力データがスカラであった場合には，スカラ演算と比べて処理に時間がかかる．そこで，MATLAB の変数の属性を解析し，実行時の解析を少なくする手法が提案されている[1)2)]．生成された Fortran90 のコードをコンパイルして，実行することで，MATLAB コンパイラが生成した C コードを，コンパイルして実行するよりも高速に処理を行うことができる[1)2)]．

本手法では，以下の 2 段階の処理を行うことで，MATLAB プログラムから Fortran90 のコードに変換する．

(1) 変数の属性である型，形状，サイズの決定．

(2) Fortran90 のコード出力．

MATLAB プログラムの変数の属性を解析して得た情報を利用し，Fortran90 のコードを生成する．Fortran90 では，変数の宣言を行う必要があるので (1) の処理を行い変数の属性の情報を得る．また，Fortran90 の行列計算を行う際の作業領域として一時変数が必要となる．そのため，一時変数も他の変数同様に (1) の処理を行い，変数の宣言をする．これらの変数の属性の解析には，MATLAB プログラムから Fortran90 のコードに変換する前に解析する静的解析と，実行時に解析する動的解析の 2 種類がある．以下，静的解析，および動的解析について述べる．

表 1 データの型の決定規則
Table 1 The Determination Rule of the Sharp of Data.

	B					
A	Φ	L	I	R	C	?
NULL (Φ)	Φ	L	I	R	C	?
LOGICAL (L)	L	L	I	R	C	?
INTEGER (I)	I	I	I	R	C	?
REAL (R)	R	R	R	R	C	?
COMPLEX (C)	C	C	C	C	C	C
UNKNOWN (?)	?	?	?	?	C	?

表 2 データの形状の決定規則 (乗算)
Table 2 The Determination Rule of the Rank of Data(mult).

A*B	B			
A	S	Vr	Vc	M
Scalar (S)	S	Vr	Vc	M
Row Vector(Vr)	Vr	-	M	-
Column Vector(Vc)	Vc	S′	-	Vc′
Matrix (M)	M	Vr′	-	M′
′Only if $A_y = B_x$;otherwise error				

2.1 MATLAB 言語の静的解析

静的解析では，MATLAB プログラムを SSA 形式[1)2)5)] に変換することで，変数の宣言に必要な変数の属性を解析する．変数の宣言を行うために必要な情報は属性と呼ばれ，型，形状，サイズの 3 つがある．この 3 つについて以下に示す．

- 型: COMPLEX，REAL，INTEGER，LOGICAL.
- 形状: Scalar，Row Vector，Column Vector，Matrix.
- サイズ: 行列やベクトルのサイズ.

変数解析で得られた結果は変数の属性値として各変数に与えられる．

MATLAB 言語は，GOTO 文を含まないので，簡単に SSA 形式で表すことができる．静的解析では，オペランドの属性の情報を利用して演算結果部の属性である型，形状，サイズを解析する．以下に変数の型の決定と形状及びサイズの決定について述べる．

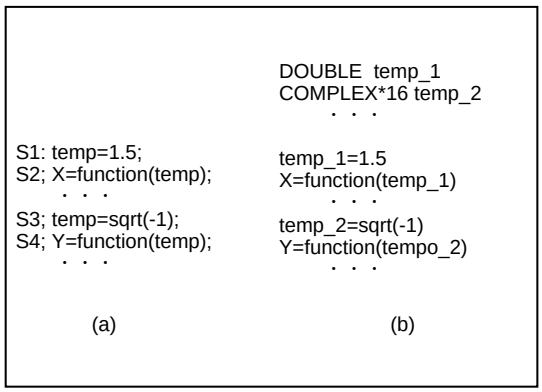

図 1 変数の型の例
Fig. 1 An Example of Type of Variables.

```
S1:     if(A_D1 .ne B_D1 .or. A_D2 .ne. B_D2) then
S2:         if(ALLOCATED(A)) DELLOCATE(A)
S3:         A_D1 =B--D2
S4:         A_D2 = B_D2
S5:         ALLOCATE(A(A_D1,A_D2))
S6:     end
S7:     do i=1 ,A_D1
S8:           do j = 1 ,A_D2
S9:                A(i,j)=B(i,j)+0.5
S10:          end
S11:    end
```

図 2 サイズを解析するための使用例
Fig. 2 An Example of Use of Tags for Size Analysis.

2.1.1 変数の型の決定

演算式の右辺の項であるオペランドの 2 つの型を使用して演算結果部の型を決定する．**表 1** に型の解析を行う規則を示す．表 1 中の NULL 型は解析していないことを表す型で，未定義であることを示す．また，UNKOWN 型は解析しても定義することができない変数の型である．解析結果が UNKOWN 型である変数は，実際の型は LOGICAL 型，INTEGER 型，REAL 型，COMPLEX 型のいずれかである．これらと COMPLEX 型の演算は，どの場合でも演算結果部の型が COMPLEX 型となるため，UNKOWN 型と COMPLEX 型の演算の演算結果部は COMPLEX 型になる．また，表 1 の規則では変数 A，変数 B が INTEGER 型同士のとき演算結果部の型は INTEGER 型となるが，例外として除算を行う場合は REAL 型となる．

2.1.2 変数の形状の決定

演算部の 2 つのオペランドの形状を用いて，演算結果部の形状を決定する．$A * B$ の演算結果部の形状の解析規則を**表 2** に示す．演算できない組合せはエラーとなり，表 2 中では “–” で表している．乗算以外の演算に関しても，表 2 と類似の規則を用いて解析する．このとき，形状を決定することができない変数は Unkown 型とする．列数または行数の一方だけが 1 と分かっているときの変数の型は Not Matrix 型，行数も列数も 1 でないと分かっているときは Not Scalar 型とする．静的に変数の形状を決定することで不必要なアドレス参照によるオーバヘッドを削減することができ，処理時間を短縮することができる．

2.2 MATLAB 言語の動的解析

変数の型や形状，サイズを静的解析で決定することができない場合には，実行時に他の変数の属性を利用して解析する．このため，変数の属性と他の変数との関係を表すタグを用意し，他の変数との関係を表すことで，変数の型やサイズを決定する．以下に，タグを用いた解析方法について述べる．

2.2.1 変数の型の決定

MATLAB プログラムは，同じ変数名で違う型の値を代入できるので，Fortrn90 コードを生成するときには，実数の場合，複素数の場合というように，場合分けして変数の型を宣言したコードを生成する．動的な変数の型の解析では，頻繁なタグの比較による時間的コストを下げるために，実数と複素数の区別のみを行う．MATLAB プログラムから Fortran90 のコードに変換する際に，実数及び複素数の演算式をそれぞれ生成する．また，前の変数の型と異なる型に再定義される変数は，それぞれの変数の型毎に異なる名前を割り当てる．

図 1 に変数の型を決定する例を示す．図 1(a) の MATLAB プログラムは図 1(b) のような Fortran90 のコードに変換される．図 1(a) 中の変数 temp には S1 で実数，S3 で複素数が代入されるが，Fortran90 のコードでは違う変数として扱う必要がある．そこで，Fortran90 のコードを生成する際に，図 1(b) ように型の違うデータを入力する毎にそのデータにあわせ Fortran90 のコード内で宣言する．

2.2.2 変数のサイズの決定

変数が持っているメモリ領域の大きさを変更するときにタグを使用する．例えば，$A = B + 1.5$ の式の中の変数 B の形状が Unknown 型であった場合，**図 2** のようなコードが生成される．図 2 の A_D1，A_D2，B_D1，B_D2 の各変数は，それぞれ変数 A の行数，列数，変数 B の行数，列数を表すタグである．図 2 は，変数 B のサイズと変数 A のサイズが違う場合に，変数 B と同じサイズにするためのコードである．ただし，この式が変数 B の最初の定義であるときは，図 2

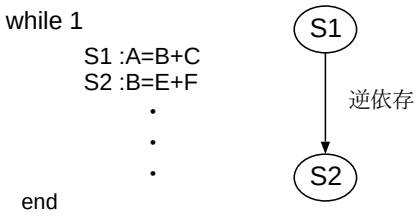

図 3 逆依存グラフ
Fig. 3 The Graph of Anti-dependence.

中の S1 および S2 による再定義は必要ない．初めて定義された変数およびループ中で変化しない変数に対しては，初めに領域を割り当てたあとは新たに割り当てテストを行わない．

3. 静的解析の拡張

前章で述べた静的解析手法では，2 つのオペランドの属性の情報がなければ演算結果部の属性を静的に決定することができないので，動的解析を行うコストが大きくなる．ループ中で逆依存関係のある変数は，イタレーション毎に属性が変化しないことを利用して変数の解析を行う．

図 3 の例では，ループ中の後方の S2 で定義された変数 B の属性の情報を用いて，ループ中の前方の S1 に存在する変数 A の属性解析を行うことができることを表している．つまり，ループ中に逆依存が存在するとき，後方で定義される変数の属性が決定されると，前方で定義される変数の属性が判明する．この情報を利用してより多くの変数に対して静的解析を行うことができるため，動的解析の際の属性を解析するコストを削減することができる．また，一方のオペランドが分かっていれば変数の属性を一意に決定することはできなくとも，属性の候補数を削減することが可能である．さらに，従来の属性変換規則を拡張し，多くの変数を静的に解析することで，動的に行う属性解析を削減する．

以下，3.1 節で従来の型の解析を拡張する方法について，3.2 節では従来の形状解析を拡張した解析方法について述べる．

3.1 変数の型解析規則の拡張

従来手法では，2 つのオペランドから演算結果部の変数の型を決定する．本稿ではこれに加えて，演算結果部の型の情報を利用してオペランドの解析を行う．これによりコード量を削減でき，動的に解析する量も削減できる．例えば，逆依存の関係から演算結果部の型が実数であると分かった場合，表 1 よりオペランドの型は複素数ではなく，少なくとも一方は実数であることが分かる．また同様に，演算結果部が整数の場合，双方のオペランドは実数および複素数ではなく，少なくとも一方は整数であることがわかる．これらのように，演算結果部の型を利用して解析することで属性の型の候補の数を減らすことができる．また，オペランドの 1 つの型が分かる場合，演算結果部の型が複素数でなければもう一方のオペランドの型の候補を減らすことができる．例えば，図 3 の変数のうち，S1 の変数 B の型が REAL 型，変数 C の型が INTEGER 型であったとき，変数 A の型は従来手法で解析すると表 1 より REAL 型と決定する．S2 の変数 E，変数 F の型が UNKNOWN 型であるとき，従来手法では型の決定を行うことができない．提案手法を用いて解析した場合，S1 で変数 B の属性が REAL 型と分かっているので，逆依存関係である S2 の変数 B のオペランド，変数 E，変数 F の型の候補は表 1 より REAL 型，INTEGER 型，LOGICAL 型となる．また，少なくとも変数 E，変数 F の一方の属性が REAL 型であることが分かる．

表 3 従来手法と提案手法の形状情報
Table 3 The Rank Information on the Conventional Method and the Proposal Method.

従来手法	提案手法
Not Matrix	S, Vr
	S, Vc
	Vr, Vc
	S, Vr, Vc
Not Scalar	Vr, Vc
	Vr, M
	Vc, M
	Vr, Vc, M
Unknown	S, Vr, M
	S, Vc, M
	S, Vr, Vc, M

3.2 変数の形状変換規則の拡張

MATLAB プログラムを静的に解析する際，従来手法では解析前の未定義としていた変数は提案手法では Scalar 型，Row Vector 型，Column Vector 型，Matrix 型の 4 つの型を形状の候補として持つ変数として解析する．従来手法で Not Scalar 型，Not Matrix 型，Unkown 型として表した変数の形状を，本手法では**表 3** に示すように，変数は 4 つの形状のうちのいくつかを形状の候補として持つ．表中の S は Scalar 型，Vr は Row Vector 型，Vc は Column Vector 型，M は Matrix 型を示す．例えば，従来手法では Not Matrix 型で表した形状を，提案手法では形状の候補を Scalar 型，Row Vector 型，Column Vector 型とし解析を行う．

```
r=A' * b-A' * A * X;
P=r;

while 1
        if abs(r)>0.000001
              break
         end

         a=(r' * r) / (P' * A' * A * P);
         X=X+a*P;
         r1=r-a * A' *A *P;
         B=(r1' * r1) / (r' * r);
         P=r1 + B * P;
         r=r1;
end
```

図 4 MATLAB で書いた CG 法のプログラム
Fig. 4 A MATLAB Program of a CG Method.

一方のオペランドの形状が Scalar 型でない 1 つの形状に決定するとき，演算結果部およびもう一方のオペランドの形状の情報は，表 2 の解析規則を用いてエラーとなる形状の候補を減らすことができる．例えば，演算 $A * B$ を行う場合に，変数 B の形状が Column Vector 型のとき，表 3 より変数 A が Column Vector 型および Matrix 型のときは演算結果部の形状がエラーとなるので，演算結果部の形状候補から Column Vector 型および Matrix 型の 2 つを減らすことができる．また，変数 B の形状がスカラのときは，変数 A および演算結果部の形状候補の数を削減することができない．変数 B の形状が複数存在する場合，変数 B の候補それぞれに対して変数 A の演算結果部の候補の数を削減し，各候補の OR をとる．演算結果部の形状が決定できるときも，オペランドの形状が決まったときと同様の解析を行う．

これらのように，提案手法では，動的に属性を決定する変数を静的に解析することができる．また，決定できなくても実行時に変数の属性を決定する候補の数を減らすことができるので，実行する際に動的に変数を解析するコストが少なくなるため処理時間が短縮できると考えられる．

4. 生成されるコードの評価

提案する手法の有効性を示すため，CPU が Pentium4 3.4GHz，メモリが 2GByte の計算機上で評価を行った．OS は Linux 2.4.26，コンパイラは gcc Version 3.3.3，コンパイルのオプションとして-O を用いた．

従来手法による解析方法で解析した結果をもとに変換した C コードと，本稿で提案する解析方法を用いて解析した結果をもとに変換した C コードをコンパイルし，実行した結果を比較する．**図 4** に評価に用いる MATLAB 言語で記述した CG 法プログラムを示す．このプログラムを実行する際，変数 A に実数型の Matrix 型，変数 b に実数型の Row Vector 型のデータを入力した．

表 4 静的解析後に得られた形状情報
Table 4 The Rank Information after Static Analysis.

解析結果	従来手法	提案手法
S,Vr,Vc,M	28	4
S,M	N/A	1
S,Vr	N/A	11
S,Vc	N/A	7
S	N/A	5

表 5 CG 法の処理時間
Table 5 Expreimental Results of the CG Method.

行列サイズ	従来手法	提案手法	高速化率
100 × 100	0.19	0.18	1.014
200 × 200	9.10	8.09	1,124
300 × 300	60.52	59.58	1.015
400 × 400	137.77	135.37	1.017

形状の静的解析において，従来手法では全ての変数を決定することができず，Unknown 型のままである．一方，提案手法では静的解析された変数の形状は**表 4** のようになった．解析する変数の数は一時変数を含めて全部で 28 個である．MATLAB 言語の静的解析時にはタグを生成し，実行時にタグを更新し，タグを用いて解析することにより，変数の宣言を行う．実行時に変数の形状を再定義するためには，二次元配列の変数は列数，行数に対する 2 つのタグを更新しなければならないが，一次元配列の変数のタグの更新は 1 つだけでよい．Scalar 型である変数はタグの更新は行わない．従来手法では 28 個全ての変数が Unknown 型であるため二次元配列で宣言される．一方，提案手法では Scalar 型に決定できる 5 個の変数は Scalar 型で，形状の候補に Matrix 型が存在しない 18 個の変数は一次元配列で宣言されるため，タグの更新回数が少なく，高速に処理を行うことができると考えられる．

二次元配列や一次元配列で宣言された変数にスカラのデータが代入された場合，二次元配列では二重ループ，一次元配列では一重ループを用いて演算が行われるため，Scalar 型の変数を用いて処理するよりも時間がかかる．

表 5 に変数 A の行列サイズを 100 × 100，200 × 200，300 × 300，400 × 400 としたときの，従来手法により解析して得られたコードと提案手法により解析して得られたコードを，コンパイルして実行したと

きの処理時間を示す．表 5 より，提案手法の方が従来手法よりも処理時間が短いことがわかる．高速化率は 1.01 倍から 1.12 倍の間で変化すると考えられる．

5. おわりに

本稿では，MATLAB 言語で記述されたコードを C に変換する際に動的に解析する部分を静的に解析することで実行時の処理を削減し，高速化する手法を提案した．提案した手法を評価した結果，従来手法よりも提案手法の方がわずかながら速くなることが確認できた．今後は，シミュレーションのプログラムなどに対して提案する手法を応用することで，有効性を確かめる．

参考文献

1) De Rose, L. Padua, D.:Techniques for the translation of MATLAB programs into Fortran 90, ACM trans. Programming Languages and Systems,Vo.21, No.2, pp.285-323(1999)
2) De Rose., Padua, D. :A MATLAB to Fortran 90 Translator and its Effectiveness(1996) にじげん配列の
3) 川端英之, 鈴木睦:MATLAB に基づく疎行列計算向けコード生成, 情報処理学会ハイパフォーマンスコンピューティング研究会報告, 2003-HPC-96(2003)
4) 川端英之, 鈴木睦:MATLAB 記述のコンパイレーションのプログラムに対しに有効瀬を確かめて行く，mta 今回手提案した複数種類の疎行列データ構造への対応，情報処理学会計算機アーキテクチャ・ハイパフォーマンスコンピューティング合同研究会報告, 2003-HPC-96(2003)
5) 田中育男:コンパイラの構成と最適化，朝倉書店，pp320-pp358(1999)
6) 笠原博徳:並列処理技術，コロナ社，pp114-116 (1991)