計算機アーキテクチャ 139－5
（2000．8．3）

データ依存解析のための二次不定方程式の分解

神戸和子 * 加古富志雄 †

* 奈良女子大学人間文化研究科
† 奈良女子大学理学部

概要

自動並列化コンパイラ技術において，逐次原始プログラムから並列性を抽出するために多くのデータ依存解析手法が提案されてきた．ほとんどの手法は解析対象を $\Sigma_{i=1}^{n} a_i x_i + c$($x$：整数変数，$c$：整数定数) であらわされる線形な添え字式に限定しているが，実際のプログラムコードには，非線形な添え字式が頻繁にあらわれる．本稿では，ベンチマークセット：Eispack，Linpack，Lapack の調査結果をもとに，非線形な添え字式のひとつである二次不定方程式を，特定の条件を満たす場合に一次不定方程式に分解する手法について述べる．一次式に分解してデータ依存解析を行うことで，効率的でより精度の高い解析結果が得られる．

Reduction of Quadratic Equations for Data Dependence Tests

Kambe Kazuko * Fujio Kako †

* Graduate School of Human Culture, Nara Women's University
† Faculty of Science, Nara Women's University

Abstract

A number of data dependence tests have been proposed for detecting parallelism in sequential programs. Despite the fact that nonlinear subscripts appear in real-life programs sometimes, most dependence tests restrict array subscripts to be linear function of loop index variables. The result of the experiment from Eispack, Linpack and Lapack have shown us that quadratic expressions that it is one of nonlinear subscripts can be expressed in a special form. This paper presents a technique that breaks down a quadratic equation into two linear equations if it satisfies some conditions, then it improve accuracy and efficiency of data dependence analysis.

1 はじめに

自動並列化コンパイラにおいて，逐次原始プログラムからより多くの並列性を抽出するために，配列変数に関するデータ依存解析はきわめて重要である．多くのデータ依存解析手法は，解析の対象を線形な添え字式に限定しているが，実際には配列参照の添え字式が非線形式である例は多い．非線形の添え字式のうち特に二次式は，三角行列を一次元化した配列の要素を参照しているために生じている．

以下のプログラムコードは，数値計算ライブラリ：Eispack，Linpack，Lapack からの例である．

```
IJ = 0
DO J = 1,N
  IK = 0
  DO K = 1,J
    JK = IJ + K
    KK = IK + K
    A[JK] = ...
      ... = A[KK]
```

```
      IK = IK + K
    ENDDO
    IJ = IJ + J
  ENDDO
```

上記のプログラムコードから誘導変数 **JK** と **KK** を消去すると，次のプログラムコードが得られる．

```
DO J = 1,N
  DO K = 1,J
    A[(J * J - J + 2 * K) / 2] = ...
      ... = A[(K * K + K) / 2]
  ENDDO
ENDDO
```

この場合のデータ依存解析問題は，

$$\begin{cases} (j^2 - j + 2k)/2 = (\hat{k}^2 + \hat{k})/2, \\ 1 \le j \le n,\ 1 \le k \le j,\ 1 \le \hat{k} \le j \end{cases}$$

を満たす整数解が存在するかどうかを判定することである．この二次不定方程式を直接解くことはひじょうに計算負荷が大きい．しかし，この問題は一次不定方程式に分解することが可能で，分解後の依存問題は，

$$\begin{cases} j = \hat{k}, \quad 1 \le j \le n, \quad 1 \le \hat{k} \le j, \\ k = \hat{k}, \quad 1 \le k \le j \end{cases}$$

となり，ループ繰越依存（loop carried dependence）が存在しないことが容易にわかる．

本稿では，二次不定方程式が特定の条件を満たす時に一次不定方程式に分解可能であることを示し，その手法を提案する．二次不定方程式に対応していない依存解析手法でも，一次式ならば適用することも可能であり，計算負荷も小さく，解析精度を高めることができる．本手法は，プログラムコード中の二次形式の添え字式は三角行列を一次元化した配列の参照にあらわれるもので，それらはすべて同じ形：$\frac{x(x-1)}{2} + y$ $(1 \le y \le x)$ であらわせるという観察を根拠としている．

本稿の構成は以下の通りである．第2節で，本稿で扱うデータ依存解析問題について定義し，第3節で，分解アルゴリズムとアルゴリズムを導くために必要な定理について述べ，いくつかの適用例をあげる．第4節では関連研究について述べ，第5節でまとめる．

2 データ依存解析

プログラム中のループが並列実行可能であるかどうかを判定するためには，ループ中の配列要素についてデータ依存解析をする必要がある．以下のような多重ループモデルを考える時，

```
DO i1 = L1, U1
  DO i2 = L2, U2
    ...
    DO in = Ln, Un
S1:   A[f1(i1,...,in),...,fm(i1,...,in)] = ...
S2:   ...= A[g1(i1,...,in),...,gm(i1,...,in)]
    ENDDO
    ...
  ENDDO
ENDDO
```

配列 **A** に関して文 S1 と S2 の間でデータ依存の有無を判定する問題は，以下の不定方程式と不等式を満たす整数解 $i_j (1 \le j \le n)$ の存在を判定する問題に帰着する．

$$\begin{cases} f_j(i_1, \ldots, i_n) = g_j(\hat{i}_1, \ldots, \hat{i}_n), \quad 1 \le j \le m, \\ L_k \le i_k \le U_k,\ L_k \le \hat{i}_k \le U_k, \quad 1 \le k \le n. \end{cases}$$

本稿では，特にイタレーション領域が三角の二次不定方程式を対象とする．すなわち，$m = 1$ であり，U_k がインデックス変数 $i_{k-1} (k > 1)$ を含む場合である．

3 二次不定方程式の分解

本節では二次不定方程式の分解アルゴリズムと，アルゴリズムを導くために必要な諸定理について述べる．

3.1 二次不定方程式

図1は三角行列を一次元化した時に，(x, y) 要素が一次元配列においてどの要素に対応しているかを示している．

三角行列の (x, y) 要素は，(A) の場合は一次元配列の $\left(\frac{x(x-1)}{2} + y\right)$ 要素に，(B) では $\left(\frac{(x-1)(2n-x)}{2} + y\right)$ 要素に対応している．(B) において，1次元配列の任

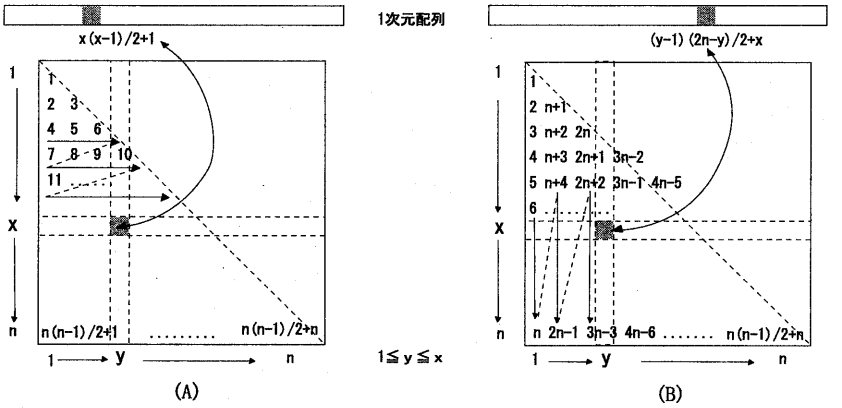

図 1: 三角行列の一次元化

意の要素 I に対して，$I \to \frac{n(n+1)}{2}+1-I$ という変換を考えると，

$$\frac{n(n+1)}{2}+1-\left(\frac{(y-1)(2n-y)}{2}+x\right)$$

$$=\frac{(n-y+1)(n-y)}{2}+n-x+1.$$

ここで $1 \le n-x+1 \le n-y+1$ である．これは $(x,y) \to (n+1-y, n+1-x)$ で変換された三角行列の要素に対応している．また，上三角行列は下三角行列を転置したものなので，(A), (B) と同様に考えることができる．

以上のことから，三角行列における (X,Y) 要素は一次元化した配列において，以下の形であらわせることがわかる．

$$\begin{cases} \frac{X(X-1)}{2}+Y, \\ 1 \le Y \le X, \\ X,Y: \text{係数が整数の一次式.} \end{cases} \quad (1)$$

定理 3.1 は，二次不定方程式が三角行列を一次元化した配列参照の添え字式の場合は，一次不定方程式に分解できることを保証し，定理 3.2 で，より一般的な二次不定方程式が，一次不定方程式に分解できる条件を与えている．

定理 3.1 以下の依存方程式とループのインデックス変数に関する制約条件が与えられた時，

$$\begin{cases} \frac{x(x-1)}{2}+y=\frac{\hat{x}(\hat{x}-1)}{2}+\hat{y}, \\ 1 \le y \le x \le n,\ 1 \le \hat{y} \le \hat{x} \le n, \\ x,y,\hat{x},\hat{y}: \text{整数変数}, \\ n: \text{整数定数} \end{cases}$$

これをを満たす解集合は

$$\begin{cases} x=\hat{x}, & 1 \le y \le x \le n, \\ y=\hat{y}, & 1 \le \hat{y} \le \hat{x} \le n. \end{cases}$$

を満たす解集合と一致する．

(証明) $1 \le y \le x$ の任意の x と y に対して，$z=\frac{x(x-1)}{2}+y$ とする．$\frac{\hat{x}(\hat{x}-1)}{2}+\hat{y}=z$ かつ $1 \le \hat{y} \le \hat{x}$ であるような整数 $\hat{x}$ を選ぶ．

$x < \hat{x}$ とすると，$x=\hat{x}-\alpha$．ただし，$\alpha \ge 1$．$\frac{x(x-1)}{2}+y=\frac{\hat{x}(\hat{x}-1)}{2}+\hat{y}$ から，$\hat{x}$ を $x+\alpha$ で置き換える．$y=\alpha x+\frac{\alpha(\alpha-1)}{2}+\hat{y}$ となり，$y>x$ が得られる．これは y のとりかたに矛盾する．

$x > \hat{x}$ の場合も同様に，$x=\hat{x}+\alpha$ として x を $\hat{x}+\alpha$ で置き換えると，$\hat{y}>\hat{x}$ を得る．これは $\hat{y}$ のとりかたに矛盾する．

したがって，$x=\hat{x}$ である． □

さらに一般的な二次不定方程式について考える．

定理 3.2 以下のような二次形式の依存方程式と制約

条件が与えられた時，

$$\begin{cases} \Sigma_{j=1}^{n}\Sigma_{i=1}^{n}A_{ij}x_ix_j+\Sigma_{i=1}^{n}B_ix_i+C \\ \quad =\Sigma_{j=1}^{n}\Sigma_{i=1}^{n}\hat{A}_{ij}\hat{x}_i\hat{x}_j+\Sigma_{i=1}^{n}\hat{B}_i\hat{x}_i+\hat{C}, \\ L_i\le x_i\le U_i,\ L_i\le\hat{x}_i\le U_i, \\ A_{ij},B_i,C,\hat{A}_{ij},\hat{B}_i,\hat{C}\ :\ \text{実数定数} \\ L_i:\ \text{ループインデックス変数の下限}, \\ U_i:\ \text{ループインデックス変数の上限} \\ (1\le i\le n,\ 1\le j\le n). \end{cases} \tag{2}$$

もし，次の条件を満たす整数定数 a_i，b_i，c，d，$\hat{a}_i$，$\hat{b}_i$，$\hat{c}$，$\hat{d}$ $(1\le i\le n)$ が存在するならば，

$$\begin{cases} A_{ii}=\frac{a_i^2}{2}, \\ A_{ij}+A_{ji}=a_ia_j, \\ B_i=ca_i+b_i-\frac{a_i}{2}, \\ C=d+\frac{c(c-1)}{2}, \\ L_i\le x_i\le U_i\text{のもとで} \\ \quad 1\le\Sigma_{i=1}^{n}b_ix_i+d\le\Sigma_{i=1}^{n}a_ix_i+c, \\ (1\le i\le n,\ 1\le j\le n,\ i\ne j) \end{cases} \tag{3}$$

かつ

$$\begin{cases} \hat{A}_{ii}=\frac{\hat{a}_i^2}{2}, \\ \hat{A}_{ij}+Aji=\hat{a}_i\hat{a}_j, \\ \hat{B}_i=\hat{c}\hat{a}_i+\hat{b}_i-\frac{\hat{a}_i}{2}, \\ \hat{C}=\hat{d}+\frac{\hat{c}(\hat{c}-1)}{2}, \\ L_i\le\hat{x}_i\le U_i\text{のもとで} \\ \quad 1\le\Sigma_{i=1}^{n}\hat{b}_i\hat{x}_i+\hat{d}\le\Sigma_{i=1}^{n}\hat{a}_i\hat{x}_i+\hat{c}, \\ (1\le i\le n,\ 1\le j\le n,\ i\ne j) \end{cases} \tag{4}$$

式 (2) の二次不定方程式は次の一次不定方程式に分解できる．

$$\begin{cases} \Sigma_{i=1}^{n}a_ix_i+c=\Sigma_{i=1}^{n}\hat{a}_i\hat{x}_i+\hat{c}, \\ \Sigma_{i=1}^{n}b_ix_i+d=\Sigma_{i=1}^{n}\hat{b}_i\hat{x}_i+\hat{d}, \\ L_i\le x_i\le U_i,\ L_i\le\hat{x}_i\le U_i(1\le i\le n) \end{cases} \tag{5}$$

(証明) 式 (2) が条件 (3) と (4) を満たす時に，一次不定方程式 (5) が導かれることを証明すれば充分である．式 (2) の A_{ij}，B_{ij}，C，$\hat{A}_{ij}$，$\hat{B}_i$，$\hat{C}$ を a_i，b_i，c，d，$\hat{a}_i$，$\hat{b}_i$，$\hat{c}$，d で置き換えて整理すると，式 (2) は，

$$\frac{(\Sigma_{i=1}^{n}a_ix_i+c)(\Sigma_{i=1}^{n}a_ix_i+c-1)}{2}+\Sigma_{i=1}^{n}b_ix_i+d$$

$$=\frac{(\Sigma_{i=1}^{n}\hat{a}_i\hat{x}_i+\hat{c})(\Sigma_{i=1}^{n}\hat{a}_i\hat{x}_i+\hat{c}-1)}{2}+\Sigma_{i=1}^{n}\hat{b}_i\hat{x}_i+\hat{d}$$

となる．定理 3.1 から，一次不定方程式 (5) が得られる． □

3.2 アルゴリズム

図 2 に n 変数の二次不定方程式を分解するアルゴリズムを示す．

アルゴリズム適用前に方程式の両辺に 2 をかけて，分子を消去しておく．アルゴリズムは右辺と左辺それぞれに適用される．入力は，各係数が整数で，x_1 が $A_{11}>0$ の最外側のループインデックス変数である二次式，出力はふたつの一次式である．

まず，A_{ij} から $a_1>0$ として整数 a_i を求める．$L_i\le x_i\le U_i$ のもとで $min(\Sigma_{i=1}^{n}a_ix_i)+c_{ini}=1$ を満たす c_{ini} と，c_{ini} に対する b_1 を計算する．$1\le\Sigma_{i=1}^{n}a_ix_i+c$ なので少なくとも $c_{ini}\le c$，かつ $\Sigma_{i=1}^{n}b_ix_i+d\le\Sigma_{i=1}^{n}a_ix_i+c$ を満たす c を求めることが目的である．$c=c_{ini}+H$，次に $c'=c+1$ の場合に，不等式：$1\le\Sigma_{i=1}^{n}b_ix_i+d\le\Sigma_{i=1}^{n}a_ix_i+c$ を満たしているか調べる．もしいづれの場合も不等式を満足しなければ，当該二次式は分解不可能であり，アルゴリズムは false を返す．

二次不定方程式の両辺について，それぞれアルゴリズムを適用して一次式が得られたら，式 (5) に分解できる．

3.3 適用例

本節では，アルゴリズムの適用例をいくつか示す．

図 3 では，左側の二次不定方程式は，

$$\frac{(i+1)i}{2}+i+1=\frac{\hat{i}(\hat{i}-1)}{2}+\hat{i}-1.$$

とあらわされ，右側の一次式が得られる．この例ではデータ依存は存在しない．

図 4 の二次方程式は，以下のようにあらわせる．

$$\frac{(k-j+1)(k-j)}{2}+1=\frac{\hat{k}(\hat{k}-1)}{2}+\hat{k}-\hat{j}+1.$$

したがって依存問題は右側のように簡単化される．

図 5 の二次式は図 1 の (B) の例である．アルゴリズム適用前に両辺を変換しておく．左辺は

$$\frac{n(n+1)}{2}+1-\frac{(-i^2+2ni-2n+i+2j)}{2}$$

入力: 2をかけて分子をはらった n 変数の二次式: $\Sigma_{j=1}^{n}\Sigma_{i=1}^{n}A_{ij}x_{ij}+\Sigma_{i=1}^{n}B_ix_i+C$, ただし, x_1 は $A_{11}>0$ の最外側のループインデックス変数 ; インデックス変数とその上限, 下限: $L_i \le x_i \le U_i$, $1 \le i \le n$.
出力: 一次式: $\Sigma_{i=1}^{n}a_ix_i+c$ と $\Sigma_{i=1}^{n}b_ix_i+d$. もし分解ができない場合は, false を返す.
if $A_{ii}=a_i^2$, $A_{ij}+A_{ji}=2a_ia_j$ を満たす $a_i(a_1>0)$ が存在する $(1 \le i \le n, 1 \le j \le n, i \ne j)$ **then**
  $min(\Sigma_{i=1}^{n}a_ix_i)+c_{ini}=1$ となる c_{ini} を求める
  $b_1 \leftarrow (B_1-2a_1c_{ini}+a_1)/2$; $H \leftarrow \lfloor b_1/a_1-1 \rfloor$
  if $H<0$ **then**
    $H \leftarrow 0$
  endif
  $c \leftarrow c_{ini}+H$; $b_i \leftarrow (B_i+a_i)/2-a_i(c_{ini}+H)$; $d \leftarrow C-(c_{ini}+H)(c_{ini}+H-1)/2$
  if $1 \le \Sigma_{i=1}^{n}b_ix_i+d \le \Sigma_{i=1}^{n}a_ix_i+c$ **then**
    return $\Sigma_{i=1}^{n}a_ix_i+c$ と $\Sigma_{i=1}^{n}b_ix_i+d$
  else
    $c' \leftarrow c+1$; $b_i' \leftarrow b_i-a_i$; $d' \leftarrow d-c$;
    if $1 \le \Sigma_{i=1}^{n}b_i'x_i+d' \le \Sigma_{i=1}^{n}a_ix_i+c'$ **then**
      return $\Sigma_{i=1}^{n}a_ix_i+c'$ と $\Sigma_{i=1}^{n}b_i'x_i+d'$
    endif
  endif
endif
return false

図 2: 分解アルゴリズム

$$= \frac{(n-i+1)(n-i)}{2}+n-j+1,$$

右辺は

$$\frac{n(n+1)}{2}+1-\frac{(-\hat{i}^2+2n\hat{i}-\hat{i}+2\hat{j})}{2}$$

$$= \frac{(n-\hat{i})(n-\hat{i}-1)}{2}+n-\hat{j}+1$$

となる. 結果は図の右側の連立一次式である.

図 4 の簡単化された連立不定方程式においては, 変数 $\hat{k}$ が, 両方の一次方程式にあらわれているので, 分離して依存テストを行えない. このような場合は, 多次元配列の添え字式を同時に解析できる手法, 例えばスタンフォード大学による SUIF[1], Omega テスト [3] または Power テスト [2] などを利用するほうが解析精度がよいことに注意すべきである.

4 関連研究

多くのデータ依存解析手法が知られているが, 非線形な添え字式を解析できるものはそれほど多くはない.

Maslov は, 多次元の長方形配列を一次元化した配列を参照するため多変数の一次式になっている添え字式を, 次元ごとに分解して, ひとつの変数のみを含む一次式に変換する rectanglur delinearization[4] と呼ばれる手法を, Omega テストに実装している. 手法適用後の添え字式は, ループの多重度と同じ次元数を持つ. この方法は多項式を扱うことはできない. そこでさらに多項式を一次式に簡単化する方法 [5] も提案している. 因子化と線形化という操作で減次を繰り返し, 一次方程式と不等式の集合に変換する. しかし、操作を繰り返すたびに方程式と不等式が増えてしまい, 操作終了後にこれらをふたたび簡単化しなければならない.

イリノイ大学の Polaris[7] に採用されている Range テスト [6] は非線形な添え字式を解析できる. 多重ループにおけるあるイタレーションでアクセスされる配列要素の範囲が, 他のイタレーションでアクセスされる要素の範囲と, オーバーラップしなければ依存がないという考えを基本としている. しかし, このテストは依存がないことを証明することが目的で, もし解が存在しても解集合を求めることはできない.

5 まとめ

本稿では二次不定方程式が特定の条件を満たす場合に, 一次不定方程式に分解する方法を示した.

$$\frac{1}{2}(i^2+3i+2) = \frac{1}{2}(\hat{i}^2+\hat{i}-2) \quad \Longrightarrow \quad i+1=\hat{i}$$
$$2 \le i \le n,\ 2 \le \hat{i} \le n \qquad\qquad i+1=\hat{i}-1$$
$$2 \le i \le n,\ 2 \le \hat{i} \le n$$

図 3: 例 (1)

$$\frac{1}{2}(k^2-2kj+j^2+k-j+2) = \quad \Longrightarrow \quad k-j+1=\hat{k}$$
$$\frac{1}{2}(\hat{k}^2+\hat{k}-2\hat{j}+2) \qquad\qquad 1=\hat{k}-\hat{j}+1$$
$$1 \le k \le n,\ 1 \le \hat{k} \le n \qquad\qquad 1 \le k \le n,\ 1 \le \hat{k} \le n$$
$$1 \le j \le k,\ 1 \le \hat{j} \le \hat{k} \qquad\qquad 1 \le j \le k,\ 1 \le \hat{j} \le \hat{k}$$

図 4: 例 (2)

$$\frac{1}{2}(-i^2+2ni-2n+i+2j) = \quad \Longrightarrow \quad i=\hat{i}+1$$
$$\frac{1}{2}(-\hat{i}^2+2n\hat{i}-\hat{i}+2\hat{j}) \qquad\qquad j=\hat{j}$$
$$1 \le i \le n,\ 1 \le \hat{i} \le n \qquad\qquad 1 \le i \le n,\ 1 \le \hat{i} \le n$$
$$i+2 \le j \le n,\ \hat{i}+2 \le \hat{j} \le n \qquad\qquad i+2 \le j \le n,\ \hat{i}+2 \le \hat{j} \le n$$

図 5: 例 (3)

本手法適用後は、二次不定方程式を扱うデータ依存解析問題よりもはるかに計算負荷が少なく，より正確な解析結果を得ることが可能である．また，GCD テストと Banerjee テスト [8] の組み合わせといった近似的なデータ依存解析手法に対しても，精度よく解析できる機会を与える．

多次元データ依存解析テストが必要な場合も生じるが，依存問題はたかだか二連立不定方程式を扱うものであり，大きな計算負荷とはならない．

参考文献

[1] D. Maydan, J. Hennesy, and M. Lam. *Efficient and Exact Data Dependence Analysis for Parallelizing Compilers.* Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, 1–14, June 1991

[2] Michael Wolfe and Chau-Wen Tseng. *The Power Test for data dependence.* IEEE Trans. Parall. Distrib Syst., 591–601, Sept 1992.

[3] William Pugh. *The Omega test: a fast and practical integer programming algorithm for dependence analysis.* Communication of the ACM, 8:102–114, August 1992

[4] Vadim Maslov. *Delinearization: an efficient way to break multiloop dependence equations.* In Proceedings of the ACM SIGPLAN '92 Conference on Programming Language Design and Implementation, 27(7):152–161, June 1992.

[5] Vadim Maslov and William Pugh. *Simplifying Polynomial Constraints Over Integers to Make Dependence Analysis More Precise.* Technical Report CS-TR-3109, University of Maryland, College Park, 1993.

[6] William Blume and Rudolf Eigenmann. *The Range Test: A Dependence Test for Symbolic, Non-linear Expressions.* Proceedings of the Conference on Supercomputing, 528–537, nov 1994

[7] W. Blume and R. Eigenmann and K. Faigin, J. Grout, J. Hoeflinger and D. Padua and P. Petersen, W.Pottenger and L. Rauchwerger and P. Tu and S.Weatherford. *Polaris: The Next Generation in Parallelizing Compilers.* Technical Report 1375, Center for Supercomputing Research and Development(CSRD), University of Illinois Urbana-Champaign.

[8] U. Banerjee. *Dependence Analysis for Supercomputing.* Kluwer Academic Publishers, 1988.