Triangular Banerjee's Inequalities with Directions

Michael Wolfe

Oregon Graduate Institute
Department of Computer Science
and Engineering
19600 N.W. von Neumann Drive
Beaverton, OR 97006-1999 USA

Technical Report No. CS/E 92-013

April 1992

Triangular Banerjee's Inequalities with Directions

Michael Wolfe

April 22, 1992

Abstract

One of the more common tests for data dependence is Banerjee's Inequalities, which can easily be used to compute direction vectors. Banerjee recently extended his test to handle triangular loop limits. A simple method can be used to find direction vectors. This note studies the simple method, showing that it is often not very precise.

1 Introduction

One often-cited data dependence test is Banerjee's Inequalities [2, 4]. This test discovers whether there is a *real* solution to the dependence equation within the loop limits, given that the loop limits themselves are known and invariant. Most compiler implementations require more information than just whether the dependence equation has a solution; often the solution is characterized by a dependence distance or direction vector. Banerjee's Inequalities are easily extended to find dependence given a direction vector, still assuming that the loop limits are known and invariant [1, 10, 12]. When the loop limits are unknown, all competent implementations of this test assume unbounded loops and find the correct conservative result. However, the test was less than adequate when the loop limits were triangular, meaning the limits of the inner loop depended on the outer loop index.

Kennedy first studied how to extend Banerjee's Inequalities to handle triangular loop limits in simple common cases [7]. Banerjee's recent monograph generalizes this case to handle any triangular loop limits [3]. However, this does not explain how to derive the necessary direction vector information.

This short note explains one method that could be used to derive direction vector information by using false triangular loop limits. A second, more complex method, essentially re-deriving the bounds based on the direction vectors, is also summarized. We show by example that neither is very precise.

2 Data Dependence

The general form of a dependence problem is shown in the following nested loop:

```
for I_1 = l_1 to u_1 do
    for I_2 = l_2 to u_2 do
        ⋮
        for I_d = l_d to u_d do
            ...A[f_1(I_1, I_2, ..., I_d), f_2(I_1, I_2, ..., I_d), ..., f_s(I_1, I_2, ..., I_d)]
            ...A[g_1(I_1, I_2, ..., I_d), g_2(I_1, I_2, ..., I_d), ..., g_s(I_1, I_2, ..., I_d)]
        endfor
        ⋮
    endfor
endfor
```

The loop has the characteristics:

- There are d nested loops with *index variables* $I_1, I_2, \ldots, I_d$.
- The array A for which dependence is being tested has s dimensions; each of the two references therefore has s subscript expressions expressed as functions of the loop index variables, shown above as f_1, g_1, f_2, g_2, $\ldots$, f_s, g_s.
- The subscript expressions are classified into the following categories:
  - constant, if the entire expression reduces to a simple constant value:

$$f_m(I_1, I_2, \ldots, I_d) = f_{m,0}$$

  - linear, if the expression is a linear combination of the surrounding loop index variables with known constant coefficients:

$$f_m(I_1, I_2, \ldots, I_d) = f_{m,0} + f_{m,1}I_1 + f_{m,2}I_2 + \cdots + f_{m,d}I_d$$

  - nonlinear, if the expression contains other quantities.
- The loops have lower and upper limit expressions, l_1, u_1, l_2, u_2, $\ldots$, l_d, u_d that may also be classified as constant, linear or nonlinear.
- Here we assume the loops are *normalized* to have a step of one [4].

The dependence problem is to find whether there are values of the loop index variables, namely

$$i_1, i_2, \ldots, i_d, \quad \text{and} \quad j_1, j_2, \ldots, j_d$$

that satisfy all the following constraints simultaneously:

- $i_1, j_1, i_2, j_2, \ldots, i_d, j_d$ are all integer,
- $$\begin{array}{ccccc} l_1 & \le & i_1 & \le & u_1 \\ l_1 & \le & j_1 & \le & u_1 \\ l_2 & \le & i_2 & \le & u_2 \\ l_2 & \le & j_2 & \le & u_2 \\ \vdots & & \vdots & & \vdots \\ l_d & \le & i_d & \le & u_d \\ l_d & \le & j_d & \le & u_d \end{array}$$
- $$\begin{array}{rcl} f_1(i_1, i_2, \ldots, i_d) & = & g_1(j_1, j_2, \ldots, j_d) \\ f_2(i_1, i_2, \ldots, i_d) & = & g_2(j_1, j_2, \ldots, j_d) \\ \vdots & & \vdots \\ f_s(i_1, i_2, \ldots, i_d) & = & g_s(j_1, j_2, \ldots, j_d) \end{array}$$

Additionally, the dependence relation can be characterized with distance or direction information. The dependence distance is a vector, defined as $(d_1, d_2, \ldots, d_d)$, where d_k is defined as $j_k - i_k$ if the value is constant for all j_k, i_k that satisfy the dependence conditions above, and d_k is unknown otherwise (typically written "*"). The dependence direction is a vector, defined as $(\theta_1, \theta_2, \ldots, \theta_d)$, where θ_k is one of the relations $\{<, =, >\}$ if the relation $i_k \theta_k j_k$ holds for all i_k, j_k that satisfy the dependence conditions, and θ_k is unknown otherwise (in the most general case, θ_k may also be one of $\{\le, \ge, \ne\}$, or is likewise written "*" when it is unknown).

In a concrete example, the dependence relation for the loop

```
for I1 = 1 to 3 do
    for I2 = 1 to 10 do
        ⋮
        A[I1,2*I2] = ...
        ...= A[I1 - 1,I2] ...
        ⋮
    endfor
endfor
```

has solutions at

i_1	i_2	j_1	j_2
1	1	2	2
1	2	2	4
1	3	2	6
1	4	2	8
1	5	2	10
2	1	3	2
2	2	3	4
2	3	3	6
2	4	3	8
2	5	3	10

Thus, the dependence distance vector is $(1, *)$, since $j_1 - i_1$ is always 1, while $j_2 - i_2$ has values that range from 1 to 5, so is not constant. The dependence direction vector is $(<, <)$, since $i_1 < j_1$ and $i_2 < j_2$ is always true for dependence solutions.

3 Overview of Banerjee's Test

Banerjee's Inequalities applies when the subscript expressions are all linear, as defined above. It proceeds by ignoring condition 1 above, namely it detects whether there are *real* (not *integer*) solutions to the dependence equation. Also, it only applies to a single dependence equation. Since in the general case, there are d dependence equations, one of (at least) four heuristics is used to help this problem.

- The dependence equations can be solved separately and the results intersected to find any dependence [10, 11].
- In many cases, the subscript expressions use disjoint sets of index variables (as in the example above). In such cases, the dependence equations are said to be *separable*; it has been shown that solving separable dependence equations independently and combining the results gives an exact result [6].
- In order to find only simultaneous solutions, the d subscript expressions can be *linearized* into the single addressing function for the array access [4, 5]. This will result in a single dependence equation, but it has been shown that this is not as exact in all cases as solving the dependence equations separately [12]. A compromise is to solve each dependence separately and also solve the linearized dependence equation.
- By noticing that a solution to the original dependence equations must also solve any linear combination of the dependence equations, we can choose

some linear combination that gives certain advantageous properties. The Lambda Test chooses linear combinations such that in certain common circumstances, a simultaneous solution to all dependence equations can be proven [8].

Note that none of these heuristics guarantee an integer solution within the loop limits.

Given a single dependence equation from linear subscript expressions, the dependence equation for dimension m looks like:

$$f_{m,0} + f_{m,1}i_1 + f_{m,2}i_2 + \cdots + f_{m,d}i_d \quad = \quad g_{m,0} + g_{m,1}j_1 + g_{m,2}j_2 + \cdots + g_{m,d}j_d$$

Banerjee's Test effectively reassociates this to look like:

$$(f_{m,1}i_1 - g_{m,1}j_1) + (f_{m,2}i_2 - g_{m,2}j_2) + \cdots + (f_{m,d}i_d - g_{m,d}j_d) = (g_{m,0} - f_{m,0})$$

which is equivalent to

$$\sum_{k=1}^{d}(f_{m,k}i_k - g_{m,k}j_k) = (g_{m,0} - f_{m,0})$$

Banerjee's Inequalities then finds lower and upper bounds for each of the d terms in the summation, such that

$$LB_k \leq f_{m,k}i_k - g_{m,k}j_k \leq UB_k$$

The lower and upper bound depended on the coefficients $(f_{m,k}, g_{m,k})$ and the loop limits (l_k, u_k). Thus we have

$$\sum_{k=1}^{d} LB_k \leq \sum_{k=1}^{d}(f_{m,k}i_k - g_{m,k}j_k) \leq \sum_{k=1}^{d} UB_k$$

which gives

$$\sum_{k=1}^{d} LB_k \leq (g_{m,0} - f_{m,0}) \leq \sum_{k=1}^{d} UB_k$$

If either of these two inequalities does not hold, then the two array references must be independent.

The first extension to Banerjee's formulation was to compute different lower and upper bounds based on a direction vector element for that dimension. Thus, to test for direction vector $(\theta_1, \theta_2, \ldots, \theta_d)$, the test would find d lower and upper bounds such that

$$LB_k^{\theta_k} \leq f_{m,k}i_k - g_{m,k}j_k \leq UB_k^{\theta_k}$$

Note that the lower and upper bound depend now on the coefficients, the loop limits, and the corresponding direction vector [12]. The summation and the rest of the test would proceed as before.

4 Triangular Banerjee Inequality

The original Banerjee Inequalities assumed the loop limits were invariant. In many common cases, the lower or upper limits of the loop depend on outer loop indices, in particular they are often *linear* in the same sense that subscript expressions are linear. Thus a linear upper loop limit is expressed as

$$u_e(I_1, I_2, \ldots, I_{e-1}) = u_{e,0} + u_{e,1}I_1 + u_{e,2}I_2 + \cdots + u_{e,e-1}I_{e-1}$$

Banerjee's algorithm for dependence testing with these *triangular* loop limits computes lower and upper bounds with an algorithm like the following:

1. Given a dependence equation like

$$f_{m,1}i_1 - g_{m,1}j_1 + f_{m,2}i_2 - g_{m,2}j_2 + \cdots + f_{m,d}i_d - g_{m,d}j_d = g_{m,0} - f_{m,0}$$

2. Rename variables as follows:

coefficients		index	
old	new	old	new
$g_{m,0} - f_{m,0}$	a_0		
$f_{m,1}$	a_1	i_1	h_1
$-g_{m,1}$	a_2	j_1	h_2
$f_{m,2}$	a_3	i_2	h_3
$-g_{m,2}$	a_4	j_2	h_4
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$f_{m,d}$	a_{2d-1}	i_d	h_{2d-1}
$-g_{m,d}$	a_{2d}	j_d	h_{2d}

This changes the dependence equation to:

$$a_1h_1 + a_2h_2 + \cdots + a_{2d-1}h_{2d-1} + a_{2d}h_{2d} = a_0$$

The loop limit coefficients are used to fill in the limit matrices such that $L\hat{h} \le h \le U\hat{h}$ as follows:

$$\begin{pmatrix} l_{1,0} & 0 & & & & & & & \\ l_{1,0} & 0 & 0 & & & & & & \\ l_{2,0} & l_{2,1} & 0 & 0 & & & & & \\ l_{2,0} & 0 & l_{2,1} & 0 & 0 & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & & & \\ l_{d,0} & l_{d,1} & 0 & l_{d,2} & 0 & \cdots & l_{d,d-1} & 0 & \\ l_{d,0} & 0 & l_{d,1} & 0 & l_{d,2} & \cdots & 0 & l_{d,d-1} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ \vdots \\ h_{2d-1} \\ h_{2d} \end{pmatrix} \le \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ \vdots \\ h_{2d-1} \\ h_{2d} \end{pmatrix}$$

and similarly for the upper limit coefficients. Note that because of the way the coefficients are numbered, the coefficient matrices L and U will have zeroes in the odd numbered columns of the even numbered rows, and vice versa (except for the zero column).

3. Set $n = 2d$.
4. Set $b_k^n = a_k$, $1 \le k \le 2d$, and $b_0^n = 0$.
5. Set $c_k^n = a_k$, $1 \le k \le 2d$, and $c_0^n = 0$.
6. Based on the values of b_n^n, c_n^n and the loop limit coefficients $L_{n,0}, \ldots, L_{n,n-1}$, compute new values $b_0^{n-1}, b_1^{n-1}, \ldots, b_{n-1}^{n-1}$ and $c_0^{n-1}, c_1^{n-1}, \ldots, c_{n-1}^{n-1}$, such that
$$b_0^{n-1} + \sum_{k=1}^{n-1} b_k^{n-1} h_k \le b_0^n + \sum_{k=1}^{n} b_k^n h_k$$
and
$$c_0^{n-1} + \sum_{k=1}^{n-1} c_k^{n-1} h_k \ge c_0^n + \sum_{k=1}^{n} c_k^n h_k$$
7. Set $n = n - 1$.
8. If $n > 0$, go to step 6; otherwise go to step 9.
9. By transitivity, $b_0^0 \le \sum_{k=1}^{n} a_k h_k \le c_0^0$. Test whether $b_0^0 \le a_0$ and $a_0 \le c_0^0$. If either of these inequalities fails, the references are independent.

5 Simple Example

Given the loop:

```
for I1 = 1 to 10 do
    for I2 = 1 to I1-1 do
        A(I1 - 2I2 + 22) = ...
        ...= A(-I1 + I2 + 14) ...
```

Algebraically, the dependence equation can be written:

$$\begin{pmatrix} f_0 & f_1 & f_2 \end{pmatrix} \begin{pmatrix} 1 \\ i_1 \\ i_2 \end{pmatrix} = \begin{pmatrix} g_0 & g_1 & g_2 \end{pmatrix} \begin{pmatrix} 1 \\ j_1 \\ j_2 \end{pmatrix}$$

or, filling in the coefficients

$$\begin{pmatrix} 22 & 1 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ i_1 \\ i_2 \end{pmatrix} = \begin{pmatrix} 14 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ j_1 \\ j_2 \end{pmatrix}$$

subject to the constraints that

$$\begin{pmatrix} l_{1,0} & 0 & \\ l_{2,0} & l_{2,1} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ i_1 \\ i_2 \end{pmatrix} \leq \begin{pmatrix} i_1 \\ i_2 \end{pmatrix} \leq \begin{pmatrix} u_{1,0} & 0 & \\ u_{2,0} & u_{2,1} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ i_1 \\ i_2 \end{pmatrix}$$

and similarly for j_1, j_2; again, filling in the coefficients, we have:

$$\begin{pmatrix} 1 & 0 & \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ i_1 \\ i_2 \end{pmatrix} \leq \begin{pmatrix} i_1 \\ i_2 \end{pmatrix} \leq \begin{pmatrix} 10 & 0 & \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ i_1 \\ i_2 \end{pmatrix}$$

Banerjee's triangular algorithm renames these variables to give the dependence equation:

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} = a_0$$

which is filled out to

$$\begin{pmatrix} 1 & 1 & -2 & -1 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} = -8$$

subject to the constraints that

$$\begin{pmatrix} L_{1,0} & 0 & & & \\ L_{2,0} & L_{2,1} & 0 & & \\ L_{3,0} & L_{3,1} & L_{3,2} & 0 & \\ L_{4,0} & L_{4,1} & L_{4,2} & L_{4,3} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} \leq \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix}$$

and

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} \leq \begin{pmatrix} U_{1,0} & 0 & & & \\ U_{2,0} & U_{2,1} & 0 & & \\ U_{3,0} & U_{3,1} & U_{3,2} & 0 & \\ U_{4,0} & U_{4,1} & U_{4,2} & U_{4,3} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix}$$

which gives

$$\begin{pmatrix} 1 & 0 & & & \\ 1 & 0 & 0 & & \\ 1 & 0 & 0 & 0 & \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} \leq \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} \leq \begin{pmatrix} 10 & 0 & & & \\ 10 & 0 & 0 & & \\ -1 & 1 & 0 & 0 & \\ -1 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix}$$

The algorithm proceeds by iteratively computing b and c coefficient vectors, such that

$$b_0^n + \sum_{k=1}^{n} b_k^n h_k \leq \sum_{k=1}^{d} a_k h_k \leq c_0^n + \sum_{k=1}^{n} c_k^n h_k$$

At step n, the new vector $b_{0:n-1}^{n-1}$ is computed from the old values of $b_{0:n-1}^n$, b_n^n, and the limit coefficients $L_{n,1:n-1}$ and $U_{n,1:n-1}$ as follows:

- If $b_n^n > 0$, set $b_{0:n-1}^{n-1} = b_{0:n-1}^n + b_n^n \times L_{n,1:n-1}$.
- If $b_n^n < 0$, set $b_{0:n-1}^{n-1} = b_{0:n-1}^n + b_n^n \times U_{n,1:n-1}$.

and similarly for c. Banerjee defines the *positive part* and *negative part* of a real number x, written x^+ and x^-, respectively, as:

$$\begin{aligned} x^+ &= \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \\ x^- &= \begin{cases} 0, & \text{if } x \geq 0 \\ x, & \text{if } x < 0 \end{cases} \end{aligned}$$

Using these definitions, the computation of b and c simplifies to:

$$\begin{aligned} b_{0:n-1}^{n-1} &= b_{0:n-1}^n + (b_n^n)^+ \times L_{n,1:n-1} + (b_n^n)^- \times U_{n,1:n-1} \\ c_{0:n-1}^{n-1} &= c_{0:n-1}^n + (c_n^n)^+ \times U_{n,1:n-1} + (c_n^n)^- \times L_{n,1:n-1} \end{aligned}$$

In this example, the b and c coefficients are computed as follows

b_0	b_1	b_2	b_3	b_4		a_0		c_0	c_1	c_2	c_3	c_4
0	$+\ 1h_1$	$+\ 1h_2$	$-\ 2h_3$	$-\ 1h_4$	$\leq$	-8	$\leq$	0	$+\ 1h_1$	$+\ 1h_2$	$-\ 2h_3$	$-\ 1h_4$
1	$+\ 1h_1$	$+\ 0h_2$	$-\ 2h_3$		$\leq$	-8	$\leq$	-1	$+\ 1h_1$	$+\ 1h_2$	$-\ 2h_3$	
3	$-\ 1h_1$	$+\ 0h_2$			$\leq$	-8	$\leq$	-3	$+\ 1h_1$	$+\ 1h_2$		
3	$-\ 1h_1$				$\leq$	-8	$\leq$	7	$+\ 1h_1$			
-7					$\leq$	-8	$\leq$	17				

Thus the left hand side of the dependence equation is bounded by -7 and 17; since the right hand side -8 does not lie within these bounds, the two references must be independent.

6 Adding Direction Vector Constraints

A simple method to use a direction vector constraint for this dependence test is to replace one (or more) of the loop limits with an appropriate non-strict inequality. For instance, if we wish to test for dependence with a $(<)$ direction in the outermost loop, we want to enforce the inequality $h_1 < h_2$, since these are the renamed indices for that loop. Because we are only interested in integer solutions, we can simplify this to $h_1 + 1 \leq h_2$. In general, to test for a $(<)$ direction for loop at level l, we want to enforce $h_{2l-1} + 1 \leq h_{2l}$. The triangular Banerjee algorithm can utilize this information by replacing the lower limit for h_{2l} by the appropriate coefficients.

For example, take the slightly modified example:

```
for I1 = 1 to 10 do
    for I2 = 1 to I1-1 do
        A(I1 - 2I2 + 20) = ...
        ...= A(-I1 + I2 + 14) ...
```

The algebraic form of the dependence equation, after renaming, is:

$$\begin{pmatrix} 1 & 1 & -2 & -1 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} = -6$$

Using the direction vector hierarchy [5], the compiler would first test for dependence without any direction vector constraints, equivalent to a $(*,*)$ direction. The triangular Banerjee test finds the same bounds as before, namely -7 and 17, so the compiler must refine one of the dependence directions. Suppose the compiler chooses to refine the inner loop direction, so it should testing for dependence with a $(*,<)$ direction. The $(<)$ for the inner loop means that the compiler should enforce $h_3 + 1 \leq h_4$. It can do this by modifying the the fourth row of the L limit matrix to:

$$\begin{pmatrix} 1 & 0 & & & \\ 1 & 0 & 0 & & \\ 1 & 0 & 0 & 0 & \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} \leq \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} \leq \begin{pmatrix} 10 & 0 & & & \\ 10 & 0 & 0 & & \\ -1 & 1 & 0 & 0 & \\ -1 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix}$$

With the modified loop limit, the b and c coefficients are computed as follows

$$\begin{array}{lcccccccr}
b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4 & & a_0 & & c_0 \quad c_1 \quad c_2 \quad c_3 \quad c_4 \\
\hline
0 + 1h_1 + 1h_2 - 2h_3 - 1h_4 & \leq & -7 & \leq & 0 + 1h_1 + 1h_2 - 2h_3 - 1h_4 \\
1 + 1h_1 + 0h_2 - 2h_3 & \leq & -7 & \leq & -1 + 1h_1 + 1h_2 - 3h_3 \\
3 - 1h_1 + 0h_2 & \leq & -7 & \leq & -4 + 1h_1 + 1h_2 \\
3 - 1h_1 & \leq & -7 & \leq & 6 + 1h_1 \\
-7 & \leq & -7 & \leq & 16
\end{array}$$

There is some freedom of choice here; we could alternatively have chosen to rearrange the matrices and change the upper limit of h_3 to be $h_4 - 1$. In that case, the dependence equation would be:

$$\begin{pmatrix} 1 & 1 & -1 & -2 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_4 \\ h_3 \end{pmatrix} = -7$$

with the limits:

$$\begin{pmatrix} 1 & 0 & & & \\ 1 & 0 & 0 & & \\ 1 & 0 & 0 & 0 & \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_4 \\ h_3 \end{pmatrix} \leq \begin{pmatrix} h_1 \\ h_2 \\ h_4 \\ h_3 \end{pmatrix} \leq \begin{pmatrix} 10 & 0 & & & \\ 10 & 0 & 0 & & \\ -1 & 0 & 1 & 0 & \\ -1 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ h_1 \\ h_2 \\ h_4 \\ h_3 \end{pmatrix}$$

Now the computed b and c coefficients are:

$$\begin{array}{rllll}
b_0 \quad b_1 \quad b_2 \quad b_4 \quad b_3 & & a_0 & & c_0 \quad c_1 \quad c_2 \quad c_4 \quad c_3 \\
\hline
0 + 1h_1 + 1h_2 - 1h_4 - 2h_3 & \leq & -7 & \leq & 0 + 1h_1 + 1h_2 - 1h_4 - 2h_3 \\
2 + 1h_1 + 1h_2 - 3h_4 & \leq & -7 & \leq & -2 + 1h_1 + 1h_2 - 1h_4 \\
5 + 1h_1 - 2h_2 & \leq & -7 & \leq & -3 + 1h_1 + 1h_2 \\
-15 + 1h_1 & \leq & -7 & \leq & 7 + 1h_1 \\
-14 & \leq & -7 & \leq & 17
\end{array}$$

The difference between the computed bounds, $-14 : 17$ and $-7 : 16$, is potentially significant. In both cases this test will assume dependence with $(<)$ direction, since -7 does lie within the limits. However, there actually is no such dependence. The values taken by the left hand subscript $I_1 - 2I_2 + 21$ are:

I_1 \ I_2	1	2	3	4	5	6	7	8	9
1									
2	21								
3	22	20							
4	23	21	19						
5	24	22	20	18					
6	25	23	21	19	17				
7	26	24	22	20	18	16			
8	27	25	23	21	19	17	15		
9	28	26	24	22	20	18	16	14	
10	29	27	25	23	21	19	17	16	13

is 8. However, the algorithm doesn't get a chance to take advantage of tighter limits placed on the solution space by other limits. In this case, the value of -7 is reached when h_3 has value 9.

7 Alternate Method

An alternate method to add direction vector constraints to the triangular Banerjee test is to derive bounds algebraically, as was done for the rectangular test [10, 12]. Suppose, for instance, we wanted to test for a (<) direction at loop nest level l; we would be looking at h_{2l-1} and h_{2l}, where we must enforce $h_{2l-1} < h_{2l}$ or $h_{2l-1} \leq h_{2l} - 1$. Let $p = 2l - 1$ and $q - 2l$. The derivation uses the inequality chain:

$$\begin{array}{ccccccc} l_p & \leq & h_p & < & h_q & \leq & u_q \\ 0 & \leq & h_p - l_p & \leq & h_q - l_p - 1 & \leq & u_q - l_p - 1 \end{array}$$

By example, the derivation of the lower bound proceeds as follows:

$$\begin{aligned} & b_p h_p + b_q h_q \\ & \quad = \quad b_p(h_p - l_p) + b_p l_p + b_q(h_q - l_p - 1) + b_q l_p + b_q \\ & \quad \geq \quad b_p^+(h_p - l_p - 1) + b_q(h_q - l_p - 1) + (b_p + b_q)l_p + b_q \\ & \quad = \quad (b_p^+ + b_q)(h_p - l_p - 1) + (b_p + b_q)l_p + b_q \\ & \quad \geq \quad (b_p^+ + b_q)^+(u_q - l_p - 1) + (b_p + b_q)l_p + b_q \end{aligned}$$

Again, imprecision arises because the upper limit u_p and the lower limit l_q do not figure into the bound. By contrast, with rectangular loop limits, u_p is the same as u_q; with triangular limits, if u_p depends on h_1, u_q will depend instead on h_2, the corresponding loop index.

What conclusions can we make? This dependence test is not particularly well suited for direction vector calculations. The problem is the inability to take into account more than one lower or upper limit on a loop index. Other dependence tests, a la the Power Test [13] or the Stanford Sieve [9], seem more well suited to this task.

References

[1] John R. Allen and Ken Kennedy. Automatic translation of Fortran programs to vector form. *ACM Trans. on Programming Languages and Systems*, 9(4):491–542, October 1987.

[2] Utpal Banerjee. Data dependence in ordinary programs. M.S. thesis UIUCDCS-R-76-837, Univ. Illinois, Dept. Computer Science, November 1976.

[3] Utpal Banerjee. *Dependence Analysis for Supercomputing.* Kluwer Academic Publishers, Norwell, MA, 1988.

[4] Utpal Banerjee, Shyh-Ching Chen, David J. Kuck, and Ross A. Towle. Time and parallel processor bounds for Fortran-like loops. *IEEE Trans. on Computers*, C-28(9):660–670, September 1979.

[5] Michael Burke and Ron Cytron. Interprocedural dependence analysis and parallelization. In *Proc. SIGPLAN '86 Symp. on Compiler Construction*, pages 162–175, Palo Alto, CA, June 1986.

[6] David Callahan. Dependence testing in PFC: Weak separability. Technical Report Supercomputer Software Newsletter #2, Rice Univ., Dept. Computer Science, August 1986.

[7] Ken Kennedy. Triangular Banerjee inequality. Technical Report Supercomputer Software Newsletter #8, Rice Univ., Dept. Computer Science, October 1986.

[8] Zhiyuan Li, Pen-Chung Yew, and Chuan-Qi Zhu. An efficient data dependence analysis for parallelizing compilers. *IEEE Trans. Parallel and Distributed Systems*, 1(1):26–34, January 1990.

[9] Dror E. Maydan, John L. Hennessy, and Monica S. Lam. Efficient and exact data dependence analysis. In *Proc. ACM SIGPLAN '91 Conference on Programming Language Design and Implementation*, pages 1–14, Toronto, June 1991.

[10] Michael Wolfe. Optimizing supercompilers for supercomputers. PhD Dissertation UIUCDCS-R-82-1105, Univ. Illinois, Dept. Computer Science, October 1982. (available from Univ. Microfilms Inc., document 83-03027).

[11] Michael Wolfe. *Optimizing Supercompilers for Supercomputers.* Research Monographs in Parallel and Distributed Computing. Pitman Publishing, London, 1989. (also available from MIT Press).

[12] Michael Wolfe and Utpal Banerjee. Data dependence and its application to parallel processing. *International J. Parallel Programming*, 16(2):137–178, April 1987.

[13] Michael Wolfe and Chau-Wen Tseng. The Power test for data dependence. *IEEE Trans. Parallel and Distributed Systems*, 1991. (accepted for publication).