Chapter 3: Dependence Testing

by Dr. Ken Kennedy

February 14, 1995

Presented by

Vijay Pai

Important definitions

Subscript partitioning algorithms

Exact and inexact tests

ZIV, SIV, MIV tests

Dependence Testing Goals

Finding all dependences $\Leftarrow$ *required!*

Proving others independent $\Leftarrow$ *optional*

Taking advantage of information

- Vectorization
- Parallelization

Index variables

appear in looping constructs

Subscript expressions

actually refer to specific array locations

```
DO i = 1,N                  ⇐ i index
   A(i+2,i) = B(2*i)        ⇐ (i+2),(i),(2*i)
                              subscript expressions
```

Iteration Vector

- Ordered n-tuple
- Value of each index variable

Dependence Equations

$$\begin{aligned} \alpha &< \beta \\ f_i(\alpha) &= g_i(\beta), \forall i \end{aligned}$$

f and g subscript functions,

α and β iteration vectors,

i varies over subscripts

Induction variables

linear functions of loop indices

```
j = 2
DO  i = 1,N
    j = j + 2
    A(j) = A(j-1)
ENDDO
```

```
DO i = 1,N
   A(2i+2) = A(2i+1)
ENDDO
j = 2N + 2
```

Assume induction vars replaced

Undecidable for arbitrary subscripts

Conservative testing only!

Possible for simple subscripts

- Exact tests NP-complete
- Efficient conservative tests ok

Zero Index Variable — ZIV

Iteration-independent value

Single Index Variable — SIV

Linear in one index

`2*i + 4`

Multiple Index Variable — MIV

Linear in multiple indices

`2*i + 4*j + 2`

Test for dependence on each subscript

Conservative test

```
DO i
   A(i+1,i+2) = A(i,i) + C
ENDDO
```

Separable subscript expression

- Indices used in only this position
- Can test this position independently

```
DO i
   DO j
      DO k
         A(i, j, j) = A(i, j, k) + C
```

i separable, j and k coupled

```
DO i
   DO j
      DO k
         A(i+1, j+2, k-1) = A(i, j+1, k) + C
```

All indices separable

Partitioning subscripts

- Separable
- Minimal coupled groups

Partition subscripts

Classify ZIV/SIV/MIV

Test each partition

Merge direction vectors

Cartesian Product

$merge$	$(<,=)$	$(<,<)$
$(<)$	$(<,<,=)$	$(<,<,<)$
$(=)$	$(=,<,=)$	$(=,<,<)$

Subtract subscript expressions

Nonzero $\Rightarrow$ independent

Based on linear Diophantine

$a_1 i + c_1$ and $a_2 i + c_2$

Strong SIV

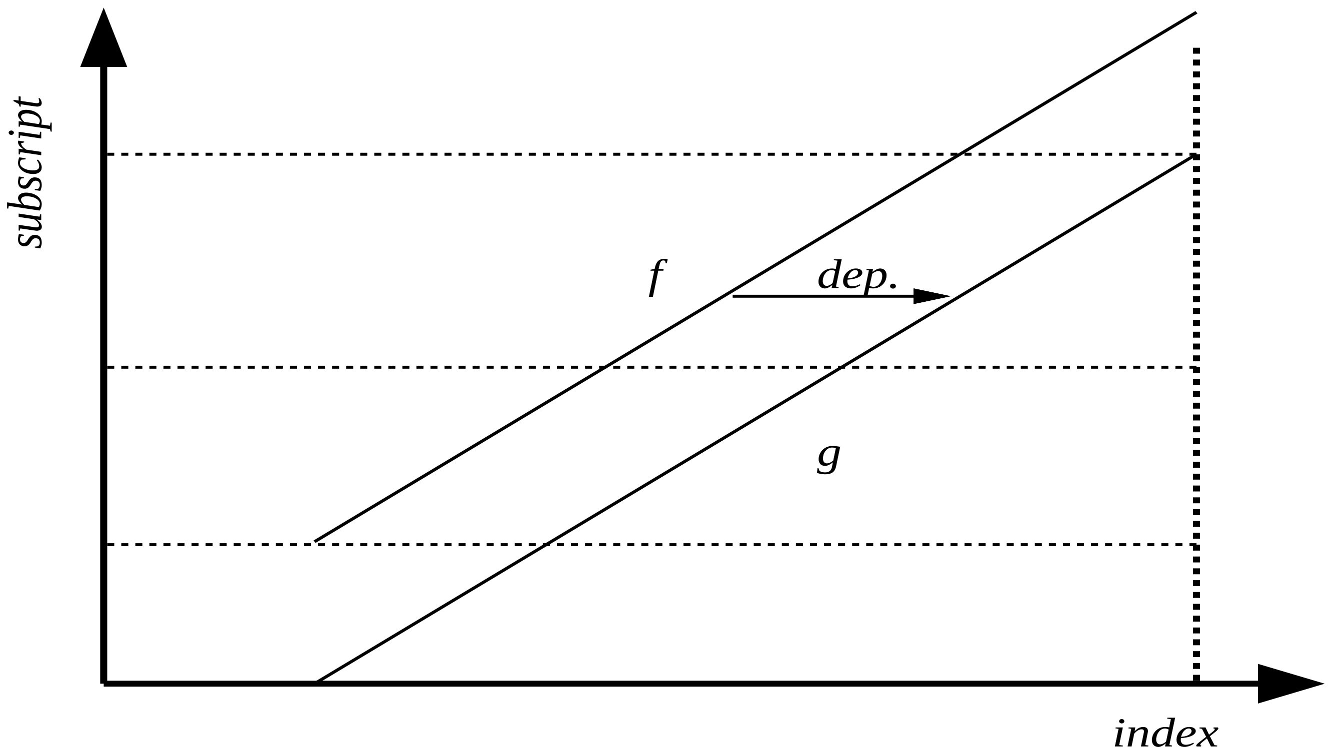
$a_1 = a_2$

Weak SIV

$a_1 \neq a_2$

- Weak Zero : $a_2 = 0$ or $a_1 = 0$
- Weak Crossing : $a_1 = -a_2$

Strong SIV



$d = i' - i = \frac{c_1 - c_2}{a}$

Dependence only if d integer and $|d| \leq U - L$

Strong SIV

```
DO i = 1,N
   A(i+2N) = A(i+N)+C
ENDDO
```

$$
\begin{aligned}
d &= \frac{2N - N}{1} \\
&= N \\
N &> N - 1
\end{aligned}
$$

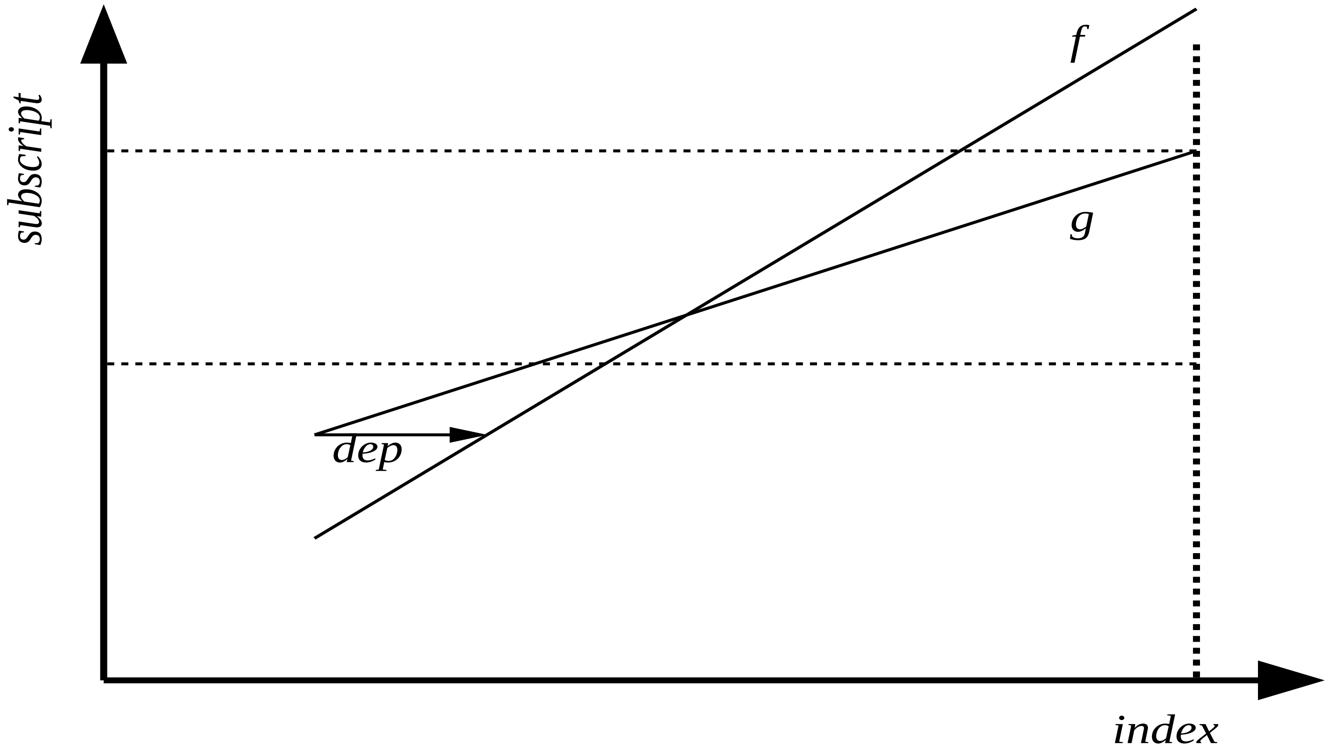
No dependence!

```
DO i = 1,99
   A(i+1) = A(i)+C
ENDDO
```

$$
\begin{aligned}
d &= \frac{1}{1} \\
&= 1 \\
1 &\leq 98
\end{aligned}
$$

Dependence!

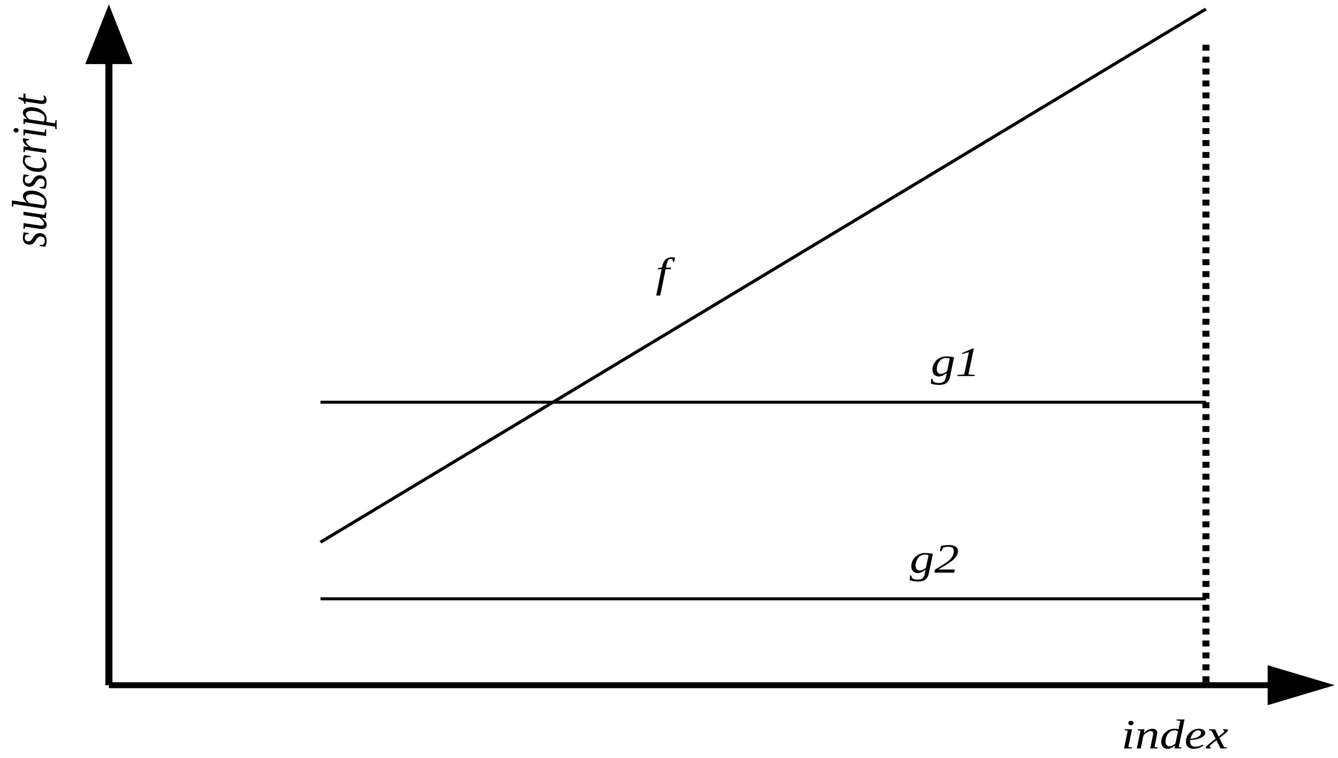
Weak SIV



$$a_1 i + c_1 = a_2 i^{'} + c_2$$

Dependence only if $\exists$ int. sol'n $(L \leq i, i^{'} \leq U)$

Weak Zero SIV



$i = \frac{c_2 - c_1}{a_1}$

Dependence only if i int. and $L \leq i \leq U$

Weak Zero SIV

```
DO i = 1,N
   Y(i,N) = Y(1,N)+Y(N,N)
ENDDO
```

$$i = 1, N$$

$$1 \leq 1 \leq N$$

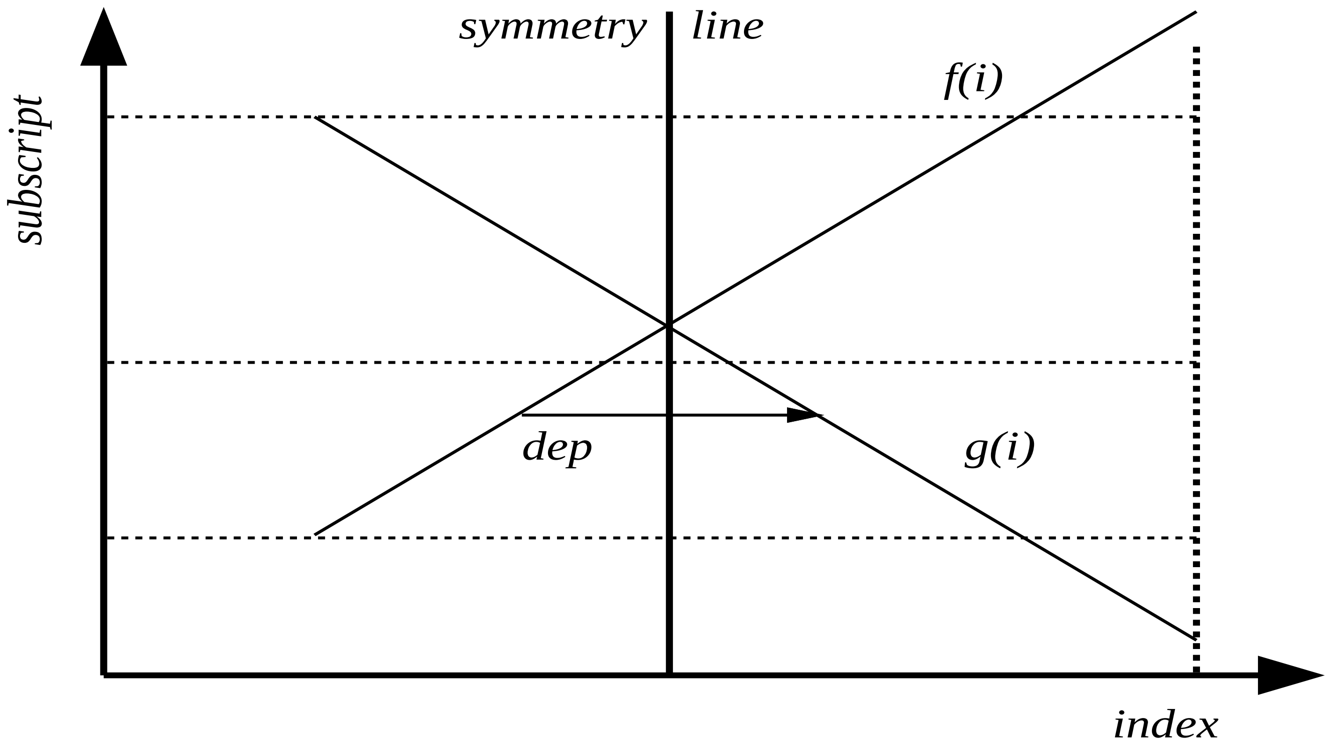
$$1 \leq N \leq N$$

Dependences!

Peel off iterations...

```
Y(1,N)=Y(1,N)+Y(N,N)
DO i = 2,N-1
   Y(i,N) = Y(1,N)+Y(N,N)
ENDDO
Y(N,N)=Y(1,N)+Y(N,N)
```

Weak Crossing SIV



$i = \frac{c_2 - c_1}{2a_1}$

Dependence only if i int. or int. +.5 and $L \leq i \leq U$

Eliminated by loop splitting

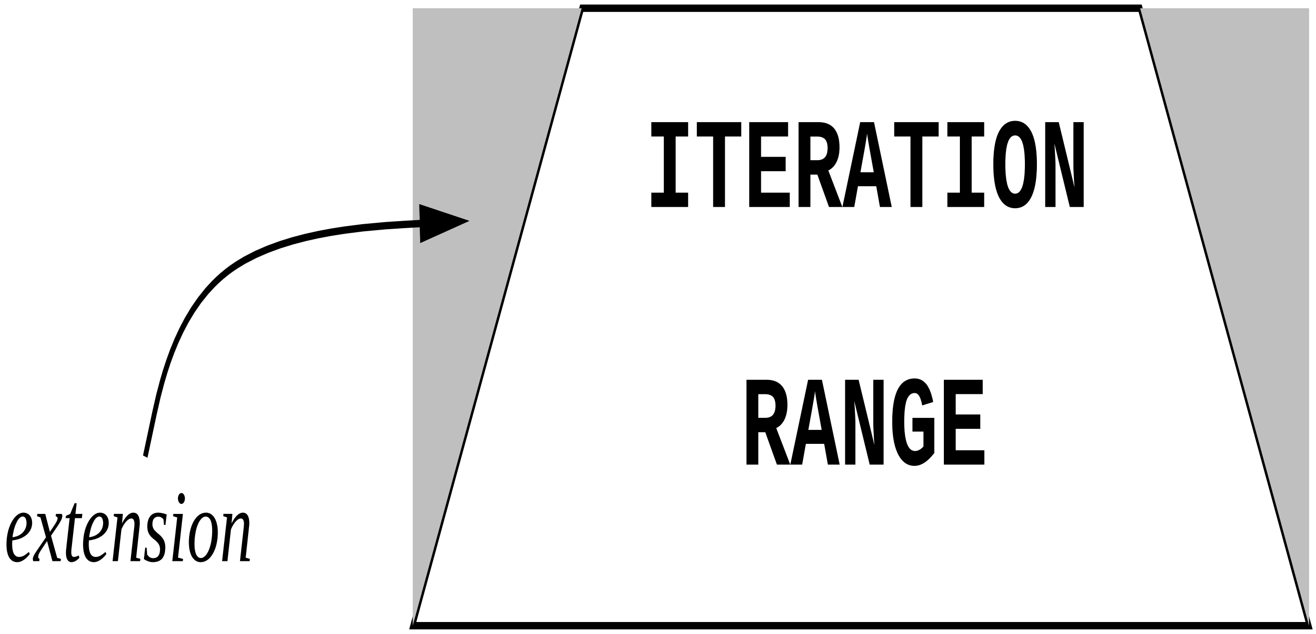
```
DO i = 1,N
   A(i) = A(N+1-i)+C
ENDDO
```

$$i = \frac{N+1-0}{2}$$

```
DO i = 1,(N+1)/2
   A(i) = A(N+1-i)+C
ENDDO
DO i = (N+1)/2+1,N
   A(i) = A(N+1-i)+C
ENDDO
```

Rectangularize the region

$\Rightarrow$ Simple conservative test



Linear Diophantine equation

$$a_1 i + a_0 = b_1 i' + b_0$$

$$a_1 x - b_1 y = b_0 - a_0$$

$$GCD(u, v) \Rightarrow \langle gcd, n_u, n_v \rangle$$

$$n_u u + n_v v = gcd(u, v)$$

GCD generates Diophantine solutions

$$\begin{aligned} GCD(a_1, -b_1) &= \langle g, n_a, n_b \rangle \\ x_k &= n_a(\frac{b_0 - a_0}{g}) + k\frac{b_1}{g} \\ y_k &= n_b(\frac{b_0 - a_0}{g}) + k\frac{a_1}{g} \end{aligned}$$

Dependences are solutions in range

Algorithm generates direction vectors

ZIV, SIV relatively straightforward

Linear SIV common case

Can we handle MIV?

Yes, but not so well...

Complex linear mappings: $Z^m \rightarrow Z$

Linear Diophantine equations

- Restricted range
- NP complete

Reduce complexity conservatively

- Search for real solutions
- Use simple analysis

$a_0 + a_1x_1 + \cdots + a_nx_n = b_0 + b_1x_1 + \cdots + b_nx_n$

Dependence only if

$gcd(a_1, \cdots, a_n, b_1, \cdots, b_n)$ divides $b_0 - a_0$

Nowhere near exact!

Banerjee Test

- Better MIV testing
- Analytic method

Delta Test

- Handles coupled subscripts efficiently
- Usually exact