論 文

Block Cyclic Reduction 기법에 의한 大型 Sparse Matrix 線型2階 偏微分方程式의 효율적인 竝列解 알고리즘

正會員 李 秉 洪* 正會員 金 正 善**

An efficient parallel solution algorithm on the linear second-order partial differential equations with large sparse matrix being based on the block cyclic reduction technique

Byung Hong LEE* Jung Sun KIM** *Regular Members*

要 約 선형2계 편미분 방정식의 일반식에 대한 계수 메트릭스를 (n−1)×(n−1) submatrices로 나누어서 block tridiagonal system으로 변환한 후 cyclic odd-even reduction 기법을 응용하여 large-grain data granularity로서 미지벡타를 구하는 block cyclic reduction 알고리즘을 작성했다.

그런데 이 block cyclic reduction 기법은 매 연산의 단계마다 병렬성이 변하여 병렬처리형 컴퓨터에는 적합하지 못하므로 이 기법을 변형해서 병렬성이 일정하며 실행시간이 보다 단축되는 block cyclic reduction 기법을 제안하고 이 기법에 의한 선형2계 편미분 방정식의 일반식의 解를 구하는 알고리즘을 작성하여 기존의 기법과 비교 고찰했다.

ABSTRACT The co-efficient matrix of linear second-order partial differential equations in the general form is partitioned with(n−1)×(n−1) submartices and is transformed into the block tridiagonal system.

Then the cyclic odd-even reduction technique is applied to this system with the large-grain data granularity and the block cyclic reduction algorithm to solve unknown vectors of this system is created.

But this block cyclic reduction technique is not suitable for the parallel processing system because of its parallelism changing at every computing stages.

So a new algorithm for solving linear second-order partial differential equations is presented by the block cyclic reduction technique which is modified in order to keep its parallelism constant, and to reduce gteatly its execution time.

Both of these algoriths are compared and studied.

*烏山專門大學 電子計算科
Dept. of Computer Science, Osan Junior College
**韓國航空大學 電子工學科
Dept. of Electronics Eng., HanKuk Aviation College
論文番號：90－57(接受1990. 3. 17)

Ⅰ. 緖 論

많은 물리적인 현상들을 수학적인 모델로 기술하면 편미분 방정식이 되며 이들 편미분 방정식은 유한 차분법에 의해서 Ax=b로 표시되는 선형방정식으로 표시 할수 있다.

여기서 A는 $A=[A_{ij}]_{n\times n}$의 matrix이고, X와 b는 열벡타로서 $X=[X_1,X_2,\cdots X_n]^t$, $B=[b_1,b_2,\cdots,b_n]^t$이다.

이러한 선형방정식을 해석하는 방법에는 고전적인방법으로 직접법과 반복법이 있으며 직접법에는 Gauss 소거법, Gauss-Jordan소거법, 인수분해법 (Crout 법, Doolittle 법, Choleski법) 등이 있고, 반복법에는 Jacobi반복계산법, Gauss-Seidel 반복계산법, SOR반복계산법등의 대표적인 방법들이 있다.

이들 고전적인 방법들은 모두가 본래 순차적으로 처리되는 순차적인 알고리즘 들이므로 상당한 변경이 없이는 병렬처리형 컴퓨터에 적합하지 않기 때문에 병렬처리형 컴퓨터의 출현과 함께 오래전부터 많은 새로운 병렬 알고리즘들이 제안되어 왔다.

AX=b 형태의 선형방정식에 대한 병렬해는 Heller에 의해 처음으로 제안 되었으며 그는 $O(N^4)$의 프로세서들로서 $O(\log^2N)$의 스탭들에 의하여 X가 계산 될 수 있음을 보였다.[6] 그리고 Wing과 Huang은 이러한 선형방식의 解 프로세서를 타스크 그래프(acycie directed graph)로 나타내고 이 그래프로부터 방정식을 푸는데 필요한 프로세서들의 수를 직접 구할 수 있음을 보여 주었다.[4]

matrix는 dense 한 것과 sparse 한 것이 있으며 또 일반적인 것과 특수한 것이 있는데 삼각행렬, 대각 행렬, 삼중대각행렬, 그리고 띠행렬 등은 특수한 행렬들이며 이들은 대개 잘 구조화된 행렬들이다. 선형 방정식의 대부분의 병렬 알고리즘은 이와같은 잘 구조화된 특수 행렬에 대하여 개발되었으며 이들은 종래의 순차적인 알고리즘에서 루프나 순환구조를 갖는 연산들을 고려하여 여러가지 새로운 기법들을 응용한 것이다.

새로운 기법으로서는 odd-even reduction, cyclic reduction, recursive doubling, quadrant interlocking, partition method등의 대표적인 기법들이다.

odd even reduction 기법은 Golub와 Hockncy가 처음으로 제안하고 Buzbee et al. 이 타원형 편미분방정식을 해석하기 위한 직접법 알고리즘을 개발 하는데 응용 했다.[1] 이 기법의 문제점은 data rearrangement overhead가 생기고 연산중에 병렬성이 변화하는 것인데 Gao는 이러한 문제점을 개선하는 알고리즘을 제안 했다.[3]

cyclic reduction 기법은 Hockncy와 Golub가 삼중대각 행렬식에 대한 알고리즘 개발에 처음으로 응용 했으며 Recursive doubling 기법은 순차적인 Gauss 소거 알고지즘에 병렬성을 부여하기 위해 Stone이 처음으로 제안했다.[2]

cyclic reduction이나 recursive doubling은 둘다 병렬처리형 컴퓨터에 응용 할수 있는 기법들이지만 병렬컴퓨터와 벡터컴퓨터에 다같이 응용 하기에는 적합 하지 못하다.

cyclic reduction기법이 STAR-100이나 CRAY-1과 같은 벡터컴퓨터에 효과적으로 응용될 수 있는데 반해 recursive doubling 기법은 ILLIAC Ⅳ와 같은 병렬컴퓨터에 효과적으로 응용된다.

partition method는 Wang이 제안한 방법으로서 병렬컴퓨터와 벡터 컴퓨터 모두 응용할 수 있으며 matrix을 여러개의 block들로 나누는 방법이다.[9]

Buzbee et al.은 matrix를 block diagonal matrix 로 변환해서 block reduction을 하여 解를 구하는 알고리즘을 제안했으며 Wallach et al.은 matrix을 여러개의 block들로 쪼개고 이들 block들에 대하여 Gauss-Seidel 알고리즘을 응용한 병렬알고리즘을 제안했다.[8]

본 논문에서는 병렬컴퓨터와 벡터컴퓨터에 효과적으로 응용 할 수 있는 선형2계 편미분방정식의 효율적인 병렬해 알고리즘을 제안하기 위하여 우선 선형2계 편미분 방정식의 matrix 형태를 block tridiagonal system으로 변환하고 이러한 block tridiagonal system에 대하여 cyclic odd-

-even reduction 기법을 응용 한다. 이때 생기는 문제점으로서 연산의 단계마다 변화하는 병렬성을 일정하게 만들기 위해 변형된 cyclic odd-even reduction 기법을 제안 한다.

본 논문의 작성은 제2절에서 선형2계 편미분방정식의 일반식에 대한 matrix 방정식을 유도하고 matrix를 block tridiagonal system으로 변환한다. 제3절에서는 cyclic odd-even reduction기법에 의해서 방정식을 순환적으로 reduction하여 解 벡터를 구하는 알고리즘을 작성 하고 이때 생기는 문제점으로서 병렬성의 변화를 개선할 수 있는 알고리즘을 제4절에서 간단한 식의 변경으로 제시한다. 제5절에서 이들 두 알고리즘에 대한 평가를 하기 위해 변형된 알고리즘과 기지의 알고리즘에 대한 산술연산의수, 병렬성, 실행시간, speed-up등을 고찰하고 이들에 대한 결과를 제 6절에서 검토하고 나서 결론을 맺는다.

Ⅱ. 線型2階 偏微分 方程式의 MATRIX

2개의 독립변수 X,Y,로 쓰여진 선형2계 편미분방정식의 일반식은 함수 U(x,y)의 정의역을 경계값이 정해진 R이라고 할때 $(x,y)\in R$에 대하여 다음과 같이 된다.[16]

$$aU_{XX}+2bU_{XY}+cU_{YY}+dU_X+eU_Y+fU+g=0 \qquad (2\text{-}1)$$

여기서 a,b,c,d,e,f,g,는 상수 또는 x,y의 함수이며 R의 경계를 ∂R이라고 하면 $(x,y)\in\partial R$에 대하여 $U(x,y)=G(x,y)$이다. R를 바둑판 모양으로 공간분할 하였을 때(x_r,y_s)에서 식 2-1을 유한차분 근사식으로 나타내면 다음과 같이 된다.

$$a\left[\frac{U_{r+1,s}-2U_{r,s}+U_{r-1,s}}{h^2}\right]+2b\left[\frac{U_{r+1,s+1}-U_{r+1,s-1}-U_{r-1,s+1}+U_{r-1,s-1}}{4hk}\right]+c\left[\frac{U_{r,s+1}-2U_{r,s}+U_{r,s-1}}{K^2}\right]+d\left[\frac{U_{r+1,s}-U_{r,s}}{h}\right]+e\left[\frac{U_{r,s+1}-U_{r,s}}{k}\right]+fu+g=0$$

이 식을 정리하여 계수들을

$$\alpha_1=\frac{2a}{h^2}+\frac{2c}{K^2}+\frac{d}{h}+\frac{e}{K}$$
$$\alpha_2=\frac{C}{K^2}+\frac{e}{K}$$
$$\alpha_3=\frac{b}{2Kh}$$
$$\alpha_4=\frac{a}{h^2}+\frac{d}{h}$$
$$\alpha_5=\frac{C}{K^2}$$
$$\alpha_6=\frac{a}{h^2}$$

로 놓고 미지수의 첨자 r과 s의 범위를 각각 $1\leqq r\leqq n-1$, $1\leqq s\leqq m-1$로 하며 경계 조건은 Dirichlet 경계조건에 의한다. 그리고 벡터 u와 b를

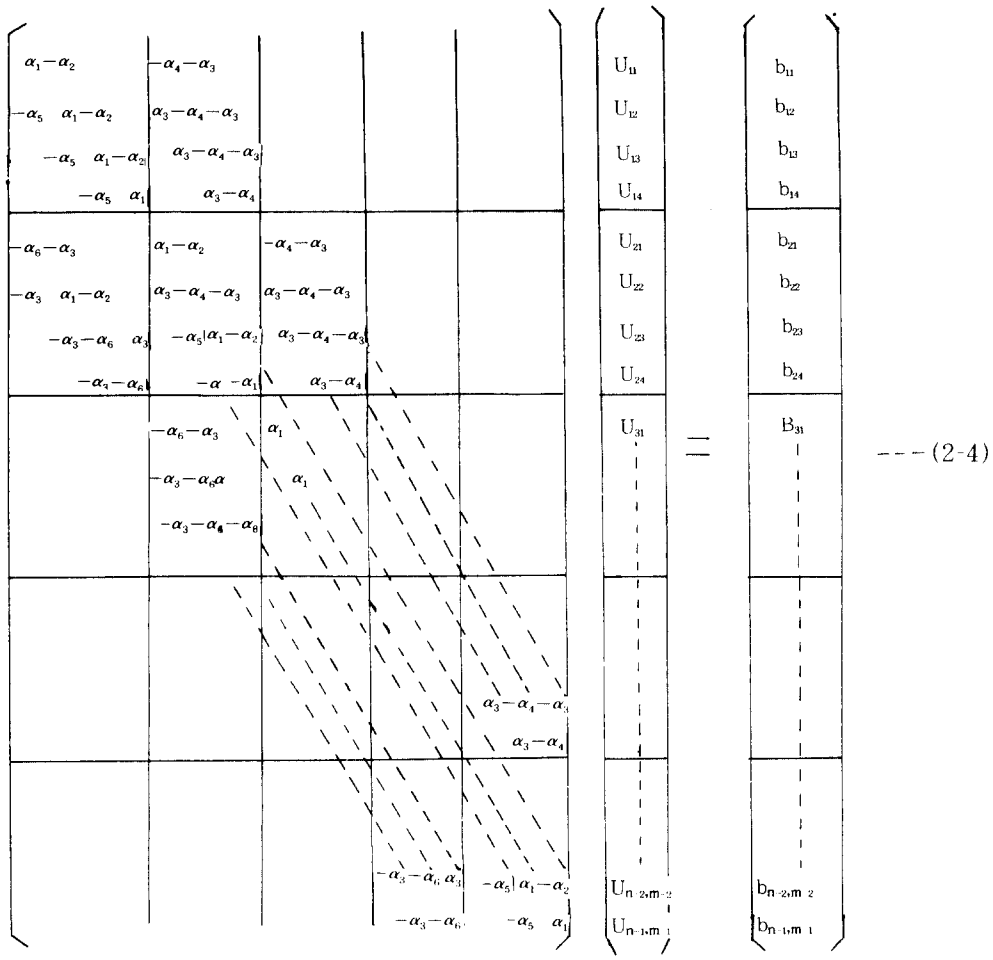
$$U=[U_{11},U_{12},\cdots,U_{1,m-1},U_{21},U_{22},\cdots,U_{n-1,m-1}]^t$$
$$b=[b_{11},b_{12},\cdots,b_{1,m-1},b_{21},b_{22},\cdots,b_{n-1,m-1}]^t \qquad (2\text{-}2)$$

로 정의하면 다음과 같은 간단한 선형방식이 된다.

$$Qu=b \qquad (2\text{-}3)$$

여기서 Q은 L×L matrix이며 다음 식 2-4와 같은 형태의 matrix로 된다. 이 matrix를 다음과 같이 $(m-1)\times(m-1)$ submatrix로 분활하면 $(n-1)\times(n-1)$ block tridiagonal matrix가 된다.

$$A=\begin{bmatrix}\alpha_1 & -\alpha_2 & & \\ -\alpha_5 & \alpha_1 & -\alpha_2 & \\ & -\alpha_5 & \alpha_1 & -\alpha_2 \\ & & -\alpha_5 & \alpha_1\end{bmatrix}_{(m-1)\times(m-1)}$$



---(2-4)

$$B=\begin{bmatrix} -\alpha_4 & -\alpha_3 & & \\ \alpha_3 & -\alpha_4 & -\alpha_3 & \\ & \alpha_3 & -\alpha_4 & -\alpha_3 \\ & & \alpha_3 & -\alpha_4 \end{bmatrix}_{(m-1)\times(m-1)}$$

$$C=\begin{bmatrix} -\alpha_6 & \alpha_3 & & \\ -\alpha_3 & -\alpha_6 & \alpha_3 & \\ & -\alpha_3 & -\alpha_6 & \alpha_3 \\ & & -\alpha_3 & -\alpha_6 \end{bmatrix}_{(m-1)\times(m-1)}$$

이때 U와 b는

$$U_j=[U_{j1},\ U_{j2},\cdots,U_{j(m-1)}]^t$$
$$b_j=[b_{j1},\ b_{j2},\cdots,b_{j(m-1)}]^t$$
$(j=1,2,3,\cdots,n-1)$

로 되고 식 2-4는 다음과 같이 된다.

$$\begin{bmatrix} A & B & & & \\ C & A & B & & \\ & \ddots & \ddots & \ddots & \\ & & C & A & B \\ & & & C & A \end{bmatrix}_{(n-1)\times(n-1)} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ \vdots \\ U_j \end{bmatrix}_{1\times(n-1)} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_j \end{bmatrix}_{1\times(n-1)} \quad \cdots(2\text{-}5)$$

Ⅲ. BLOCK-CYCLIC REDUCTION 알고리즘

식 2-5로 부터

$$AU_1+BU_2=b_1 \qquad (3\text{-}1(a))$$
$$CU_{j-1}+AU_j+BU_{j+1}=b_j \quad (j=2,3,\cdots,(n-2)) \qquad (3\text{-}1(b))$$
$$CU_{(n-2)}+AU_{(n-1)}=b_{(n-1)} \qquad (3\text{-}1(c))$$

로 쓸수 있다.

여기서 $n'=n-1$ $n=2^{k+1}$이라 하고 k는 양의 정수라고 가정하면 식 3-1(b)로 부터 세개의 인접된 방정식은

$$CU_{j-2}+AU_{j-1}+BU_j=b_{j-1}$$
$$CU_{j-1}+AU_j+BU_{j+1}=b_j$$
$$CU_j+AU_{j+1}+BU_{j+2}=b_{j+1}$$

로 되며 첫번째 식에 C를, 두번째 식에 $-A$를, 그리고 세번째 식에 B를 각각 곱해서 더하면 다음과 같이 된다.

$$C^2U_{j-2}+(2BC-A^2)U_j+B^2U_{j+2}=Cb_{j-1}+Bb_{j+1}-Ab_j \qquad (3\text{-}2)$$

$$\begin{bmatrix} 2BC-A^2 & B^2 & & & \\ C^2 & 2BC-A^2 & B^2 & & \\ & C^2 & 2BC-A^2 & B^2 & \\ & & \ddots & \ddots & \ddots \\ & & & C^2 & 2BC-A^2 \end{bmatrix} \begin{bmatrix} U_2 \\ U_4 \\ \vdots \\ U_{j-1} \end{bmatrix} = \begin{bmatrix} Cb_1+Bb_3-Ab_2 \\ Cb_3+Bb_5-Ab_4 \\ \vdots \\ Cb_{j-2}+Bb_j-Ab_{j-1} \end{bmatrix} \quad \cdots\cdots (3\text{-}3)$$

이 새로운 방정식은 j=2,4,…,n−2에 대하여 U의 짝수 첨자로 된 미지수 만을 포함하는 blcok tridiagonal system이며 다음 식3-3과 같이 된다.

이와 같은 reduction 과정은 하나의 block이 될때까지 반복적으로 실행될 수 있다. 순환적으로 프로시저를 정의하기 위하여

$$A^{(0)}=A,\ B^{(0)}=B,\ C^{(0)}=C$$
$$b_j^{(0)}=b_j$$

이라고 하면 r=1,2,…,k에 대하여

$$A^{(r)}=2B^{(r-1)}C^{(r-1)}-(A^{(r-1)})^2$$
$$B^{(r)}=(B^{(r-1)})^2$$
$$C^{(r)}=(C^{(r-1)})^2$$
$$b_j^{(r)}=C^{(r-1)}b_{j-1^{r-1}}+B^{(r-1)}b_{j+1^{r-1}}-A^{(r-1)}b_j^{(r-1)} \qquad (3\text{-}4)$$

되고, 식 3-2는

$$C^{(r)}U_{j-2^r}+A^{(r)}U_j+B^{(r)}U_{j+2^r}=b_j^{(r)} \qquad (3\text{-}5)$$

로 되며, 각 stage에서 다음과 같은 $(2^{k-r+1}-1)\times(2^{k-r+1}-1)$의 matrix 형태에 새로운 선형방정식이 만들어진다. 여기서 r은 reduction의 회수를 표시 하며 j는 $n-2^r$까지 2^r씩 증가한다.

$$\begin{bmatrix} A^{(r)} & B^{(r)} & & & \\ C^{(r)} & A^{(r)} & B^{(r)} & & \\ & C^{(r)} & A^{(r)} & B^{(r)} & \\ & & \ddots & \ddots & \ddots \\ & & & C^{(r)} & A^{(r)} \end{bmatrix} \begin{bmatrix} U_{2^r} \\ U_{2^{r+1}} \\ \vdots \\ U_{i2^r} \\ \vdots \end{bmatrix} = \begin{bmatrix} b_{2^r}^{(r)} \\ b_{2^{r+1}}^{(r)} \\ \vdots \\ b_{i2^r}^{(r)} \\ \vdots \end{bmatrix}$$

여기서 i=1,2,… ·····(3-6)

그리고 reduction의 K steps 후에는 하나의 block으로 되고 reduction은 끝난다. 이때 식 3-6은 다음과 같이 된다.

$$A^{(k)}U_{2^k}=b_{2^k}^{(k)} \tag{3-7}$$

윗 식으로부터 $U_{2^k}=(A^{(k)})^{-1}b_{2^k}^{(k)}$에 의해서 직접 구해지며 또 reduction의 K−1 step 후에는 식 3-6은 다음과 같이 된다.

$$A^{(k-1)}U_{2^k-2^{k-1}}+B^{(k-1)}U_{2^k}=b_{2^k-(2^{k-1})}^{(k-1)} \tag{3-8(a)}$$

$$C^{(r-1)}U_{2^k-(2^{k-1})}+A^{(k-1)}U_{2^k}=B^{(k-1)}U_{2^k+(2^{k-1})}=b_{2^k}^{(k-1)} \tag{3-8(b)}$$

$$C^{(k-1)}U_{2^k}+A^{(k-1)}U_{2^k+(2^{k-1})}=b_{2^k+(2^{k-1})}^{(k-1)} \tag{3-8(c)}$$

따라서 식 3-8(a)와 식 3-8(c)에 식 3-7로부터 U_{2^k}을 계산해서 각각 대입하면 $U_{2^k-(2^{k-1})}$와 $U_{2^k+(2^{k-1})}$을 각각 구할 수 있다.

마찬가지로 나머지 미지수 U_j는

$$U_j=(b_j^{(r-1)}-C^{(r-1)}U_{j-2^{r-1}}-B^{(r-1)}U_{j+2^{r-1}})/A^{(r-1)} \tag{3-10}$$

로 구해진다. 여기서 r=k,k−1,⋯,2,1이고 j는 2^{r-1}에서 $n-2^{r-1}$까지 2^{r-1}씩 증가한다. 그리고 $U_0=U_n=0$이다.

Ⅳ. 변형된 Block Cyclic reduction 알고리즘

block cyclic reduction은 전술한 식의 유도 과정에서 보는 바와 같이 매 연산의 단계마다 병렬성이 변하는 것을 알 수있으며 이와같이 병렬성이 변하는 알고리즘은 병렬컴퓨터에서는 적합하지 못하다.

Gao는 recursive doubling 기법에 의해서 이것을 개선한 바가 있지만[3] 사실 상 block cyclic reduction을 변형하면 병렬성이 일정하게 유지되는 알고리즘이 될수 있다.

식 3-4의 정의로부터

$$A_j^{(0)}=A_j,\ B_j^{(0)}=B_j,\ C_j^{(0)}=C_j$$
$$b_j^{(0)}=b_j\ \ j=1,2,\cdots,n-1$$

이라고 하면 r=1,2,⋯,k+1에 대해서

$$\left.\begin{aligned}A_j^{(r)}&=2B_j^{(r-1)}C_j^{(r-1)}-(A_j^{(r-1)})^2\\B_j^{(r)}&=(B_j^{(r-1)})^2\\C_j^{(r)}&=(C_j^{(r-1)})^2\\b_j^{(r)}&=C_j^{(r-1)}b_{j-2^{r-1}}^{(r-1)}+B_j^{(r-1)}b_{j+2^{r-1}}^{(r-1)}-A_j^{(r-1)}b_j^{(r-1)}\end{aligned}\right] \tag{4-1}$$

되고 j<=0 및 j>=n에 대해서

$$\left.\begin{aligned}A_j^{(r)}&=1\\B_j^{(r)}&=C_j^{(r)}=b_j^{(r)}=0\end{aligned}\right] \tag{4-2}$$

또는 $P_j^{(r)}=(1,0,0,0)$로 정의한다. 또 식 3-6은

$$C_j^{(r)}U_{j-2^r}+A_j^{(r)}U_j+B_j^{(r)}U_{j+2^r}=b_j^{(r)} \tag{4-3}$$

로 변형하고 이 식에다 식 4-2를 대입하면

$$U_0=0$$
$$U_n=0 \tag{4-4}$$

가 된다.

식 4-3에서 r=1인 경우

$$C_j^{(1)}U_{j-2}+A_j^{(1)}U_j^{(1)}+B_j^{(1)}U_{j+2}=b_j^{(1)} \tag{4-5}$$

(i) j가 짝수 인경우

$$\begin{bmatrix} A_2^{(1)} & B_2^{(1)} & & & \\ C_4^{(1)} & A_4^{(1)} & B_4^{(1)} & & \\ & \ddots & \ddots & \ddots & \\ & & C_n^{(1)} & A_n^{(1)} & B_{n-2}^{(1)} \end{bmatrix} \begin{bmatrix} U_2 \\ U_4 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ U_{2^k} \\ U_n \end{bmatrix} = \begin{bmatrix} b_2^{(1)} \\ b_4^{(1)} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ b_{2^k}^{(1)} \\ b_n \end{bmatrix} \cdots (4\text{-}6)$$

로 되고 다음 식 4-6과 식 4-7의 matrix 방정식이 된다.

(ii) j가 홀수 인 경우

$$\begin{bmatrix} A_1^{(1)} & B_1^{(1)} & & & \\ C_3^{(1)} & A_3^{(1)} & B_3^{(1)} & & \\ & \ddots & \ddots & \ddots & \\ & & C_{n-1}^{(1)} & A_{n-1}^{(1)} & \end{bmatrix} \begin{bmatrix} U_1 \\ U_3 \\ \vdots \\ U_{2^i-3} \\ U_{2^i-1} \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_{2^i-3}^{(1)} \\ b_{2^i-1} \end{bmatrix} \quad \cdots(4\text{-}7)$$

이와 같이 순환적으로 반복해서 reduction 하면 k+1 step에서는 U_j의 값이

$$U_j = b_j^{(k+1)} / A_j^{(k+1)}$$

로 reduction 마지막 단계에서 모든 j에 대해서 U_j의 값이 동시에 계산된다.

V. Data Granularity 와 알고리즘의 평가

1. Data granularity

matrix는 여러가지 크기의 단위로 나누어 질수 있다. 이 단위(data granularity)는 병렬성과 밀접한 관계가 있으며 large-grain, medium-grain, fine-grain등으로 나누어 생각할 수 있다. 그러나 여기서는 large-grain과 fine-grain의 두가지로 크게 구분하여 생각하겠다.

large-grain으로 구현하는 경우 기본적인 단위는 submatrix이며 submatrix 하나마다 하나의 프로세서를 할당한다. 이와같은 경우에 식 2-4의 matrix Q는 n′×n′개의 submatrix로 나누어지며 각 프로세서는 submatrix와 미지벡터 U_j와의 내적을 계산하고 그결과를 기지 Vector b_j의 subvector에서 뺀다. 이와같은 large-grain 구현에 있어서는 각 프로세서는 submatrix와 두개의 subvector를 기억하기 위한 대규모 local 메모리가 있어야 하며, 제어 프로시져와 연산 프로시져 사이에 통신은 감소되겠지만 최대 병렬성은 프로세서의 수에 제한을 받는다.

한편 fine-grain으로 구현하는 경우 기본적인 단위는 scalar 연산이며 각 프로세서는 submatrix의 각 요소와 미지 벡타 U_{jj}와의 內積을 계산하고 기지 벡타 b_j의 각성분 b_{jj}와의 뺄셈을 한다. 각 프로세서는 submatrix의 각 요소와 미지 벡타 U_{jj}및 기지 벡타 b_{jj}를 기억하기 위한 매우 작은 메모리가 필요하다. 이와 같은 경우에 프로세서들 사이의 통신은 크게 증가 하지만 최대 병렬성은 matrix Q의 nonzero 요소들의 수까지 증가한다.

따라서 제안 하고자 하는 data granularity는 프로세서들 사이의 통신과 병렬성을 고려하여 large-grain으로 구현한다.

2. 알고리즘의 평가

앞에서 설명한 알고리즘에서 계산되는 산술연산은 matrix의 (i) 전달연산, (ii) 곱셈 (또는 나눗셈), (iii) 덧셈 (또는 뺄셈)등이며 이들 산술연산 중 전달 연산의 실행시간은 무시하고 나머지 산술연산은 모두 하나의 단위시간으로 계산한다.

(1) 병렬성과 실행시간

(i) 기존의 block cyclic reduction 알고리즘은 앞에서 설명한바에 따라 다음과 같이 된다.

begin
for r=1 to $\log_2 n - 1$ do
for all $2^r \leq j \leq n-2^r$ step 2^r do
$A^{(r)} = 2B^{(r-1)}C^{(r-1)} - (A^{(r-1)})^2$
$B^{(r)} = (B^{(r-1)})^2$
$C^{(r)} = (C^{(r-1)})^2$
$b_j^{(r)} = C^{(r-1)}b_{j-2^r}^{(r-1)} + B^{(r-1)}b_{j+2^r}^{(r-1)} - A^{(r-1)}b_j^{(r-1)}$
end for all j

end for r
$U_0=0$
$U_{\log_2 n}=0$
for $r=\log_2 n-1$ to 1 do
  for all $2^{(r-1)} \leq j \leq n-2^{(r-1)}$ step $2^{(r-1)}$ do
$U_j=(b_j^{(r-1)}-C^{(r-1)}U_{j-2^{(r-1)}}-B^{(r-1)}U_{j+2^{(r-1)}})/A^{(r-1)}$
  end for all j
 end for r
End

위의 알고리즘에서 매 reduction 단계에서 각 프로세서가 수행하는 산술연산의 수는 11이고 병렬성은 $n2^{-r}-1$이다. 이때 순차적으로 실행하는 시간 t_{sq1}은

$$t_{sq1}=11\sum_{r=1}^{\log_2 n-1}(n2^{-r}-1)$$
$$=11(n-\log_2 n+1)$$
$$=11(n-\log_2 n) \qquad (5\text{-}1)$$

이고 back substitution의 매 단계에서 각 프로세서가 수행하는 산술연산의 수는 5이며 병렬성은 $n2^{-r}$이다. 이때 순차 실행시간 t_{sq2}는

$$t_{sq2}=5\sum_{r=1}^{\log_2 n-1}(n2^{-r})$$
$$=5n \qquad (5\text{-}2)$$

이다. 따라서 기존의 block cyclic reduction 알고리즘을 순차적으로 실행 하는데 필요한 전체시간 t_{sq}는 다음에 비례한다.

$$t_{sq}=t_{sq1}+t_{sq2}\fallingdotseq 16n-11\log_2 n \qquad (5\text{-}3)$$

그러나 병렬로 실행 할때 reduction과 substitution 의 매 단계는 $n/2$개의 프로세서를 사용한다면 단위 시간으로 환산 할 수 있으므로 전체 병렬 실행시간 t_{par}은

$$t_{par}=16\ (\log_2 n-1) \qquad (5\text{-}4)$$

이다.

(ii) 변형된 block cyclic reduction 알고리즘은 앞에서 설명한 바에 따라 다음과 같이 된다.

초기 조건은 $j\leq 0$와 $j\geq n$에 대해서

$B_j^{(r)}=C_j^{(r)}=b_j^{(r)}=0$
$A_j^{(r)}=1$

이다.

begin
for r =1 to $\log_2 n$ do
  for all $1\leq j\leq n-1$ do
$A_j^{(r)}=2B_j^{(r-1)}C_j^{(r-1)}-(A_j^{(r-1)})^2$
$B_j^{(r)}=(B_j^{(r-1)})^2$
$C_j^{(r)}=(C_j^{(r-1)})^2$
$b_j^{(r)}=C_j^{(r-1)}b_{j-2^{r-1}}^{(r-1)}+B_j^{(r-1)}b_{j+2^{r-1}}^{(r-1)}-A_j^{(r-1)}b_j^{(r-1)}$

    end for all j
  end for r
End

위의 알고리즘에서 reduction의 매 단계에서 각 프로세서가 실행하는 산술연산의 수는 11이고 병렬성은 $n-1$이다. 이때 순차적인 실행시간 t_{sq}는

$$t_{sq}=11\sum_{r=1}^{\log_2 n}(n-1)$$
$$=11(n-1)\log_2 n \qquad (5\text{-}5)$$

이다. 그러나 변형된 block cyclic reduction 알고리즘을 병렬로 실행 할 때 전체 병렬 실행시간 t_{par}은 $n-1$개의 프로세서를 사용 한다면 reduction 의 매 단계는 마찬가지로 단위 시간으로 환산 할수 있으므로 다음과 같이 된다.

$$t_{par}=11\log_2 n \qquad (5\text{-}6)$$

그림 1은 reduction 변수와 병렬성의 관계를 나타낸 것으로서 k=10의 경우이다. 그림에서

기존의 block cyclic redcution 알고리즘의 경우는 reduction 과정에서 reduction 변수가 증가 함에 따라 병렬성이 감소하고 있다. 그러나 변형된 block cyclic reduction 알고리즘의 경우에는 reduction 변수의 증가와 관계 없이 일정하다.

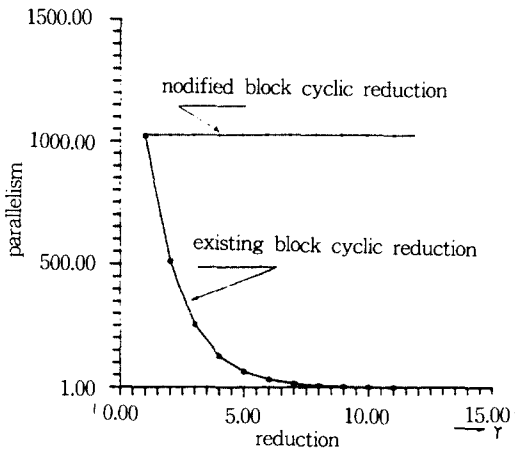


그림 1. reduction 수와 병렬성과의 관계
illustration of parailelism for k=10

그림 2는 기존의 block cyclic reduction 알고리즘의 경우 back substitution 과정에서 substitutuion 변수가 증가 함에 따른 병렬성을 나타낸 것인데 그림에서와 같이 reduction 과정과는 반대로 매 substitution 마다 병렬성이 증가 하고 있다.

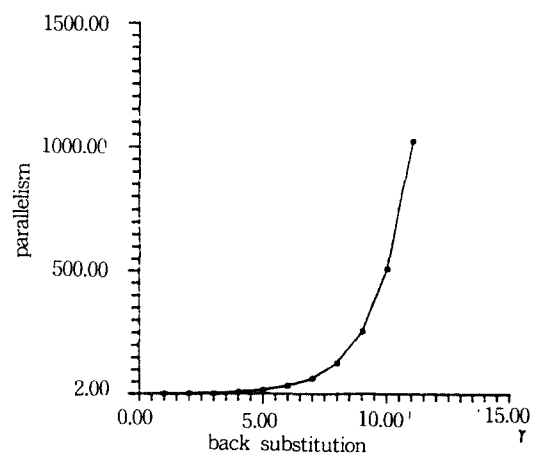


그림 2. 기존의 block cyclic reduction의 back substitution 에 대한 병렬성
iliustration of parallelism on the back substitution of existing block cycic reduction

그림 3은 프로세서 수 p와 실행시간 t와의 관계를 나타낸 것이며 그림에서 보는 바와 같이 순차적으로 실행 했을 때는 변형된 block cyclic reduction 알고리즘이 기존의 block cyclic reduction 알고리즘보다 실행 시간이많이 걸리며 프로세서 수가 증가함에 따라 그 차가 현저하게 커지는 것을 볼 수 있다. 그러나 병렬로 실행했을 때는 기존의 block cyclic reductuion 알고리즘이 변형된 block cyclic reduction 알고리즘 보다 실행시간 이 더 많이 걸리며 역시 프로세서 수가 많아지면 그 차가 현저하게 커지는 것을 알 수 있다.

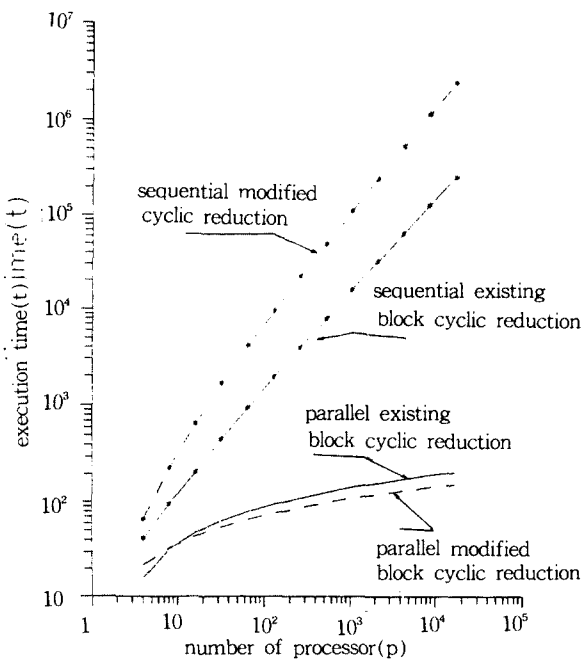


그림 3. 프로세서수와 실행시간과의 관계
illustration of execution time

(2) speed-up과 프로세서의 사용률

(i) 기존의 block cyclic reduction 알고리즘의 speed-up은 다음과 같다.

$$S_{p_1}=\frac{t_{sq_1}}{t_{par_2}}=\frac{16n-11\log_2 n}{11(\log_2 n-1)} \qquad (5\text{-}7)$$

여기서 t_{sq_1}는 기존의 block cyclic reduction 알고리즘의 순차적인 실행시간이고 t_{par_1}은 기존

의 block cyclic reduction 알고리즘의 병렬 실행 시간이다.

(ii) 변형된 block cyclic reduction 알고리즘의 speed-up은 다음과 같다.

$$sp_2=\frac{t_{sq_2}}{t_{par_2}}=\frac{11(n-1)\log_2 n}{11\log_2 n}=n-1 \qquad (5\text{-}8)$$

여기서 t_{sq_2}는 변형된 block cyclic reduction 알고리즘의 순차적인 실행시간이고 t_{par_2}은 변형된 block cyclic reduction 알고리즘의 병렬 실행 시간이다.

그림 4는 프로세서 수와 speed-up의 관계를 나타낸 것이며 speed-up은 그림에서 보는 바와 같이 변형된 block cyclic reduction 알고리즘이 기존의 block cyclic reduction 알고리즘 보다 크며 프로세서 수가 증가 할수록 현저하게 커지는 것을 알수 있다.

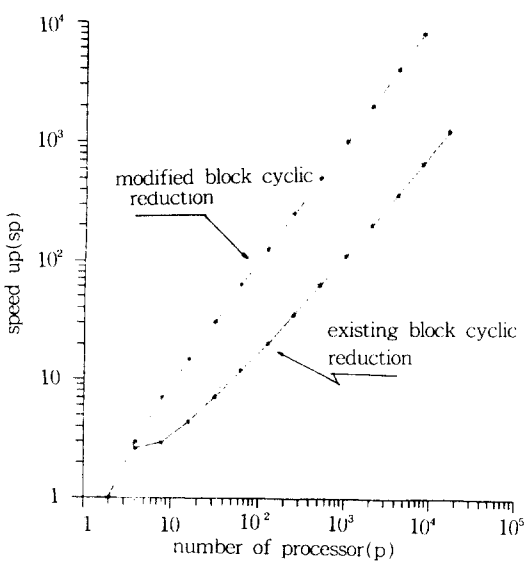


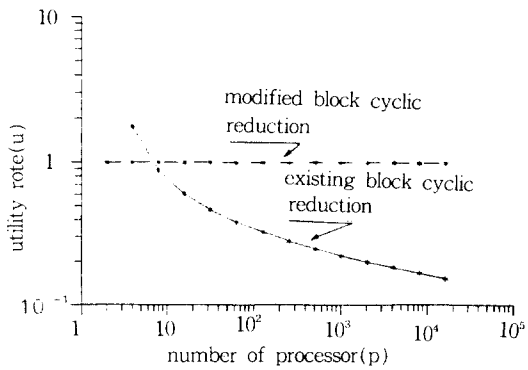
그림 4. 프로세서수와 speed-up과의 관계
illustration of speed-up

(iii) block cyclic reduction 알고리즘의 사용률은 다음과 같다.

$$U=\sum_{i=1}^{p} t_i \frac{\sum_{i=1}^{p} t_i}{t_{max}*P}=\frac{(2n-\log_2 n+1)}{n(\log_2 n-1)} \qquad (5\text{-}9)$$

여기서 t_i는 각 프로세서가 실행하는 시간이고 t_{max}는 t_i중에서 최대가 되는 시간이다. 그리고 p는 병렬로 수행하는 프로세서의 수이다.

그림 5은 사용률과 프로세서 수와의 관계를 나타낸 것이며 그림에서 기존의 block cyclic reduction 알고리즘은 프로세서 수가 증가 할수록 점점 감소하여 0에 가까와 지는데 반해 변형된 block cyclic reduction 알고리즘은 사용률이 1로서 일정하다.



(3) 기존의 parallel method들과의 비교

어떤 문제에 대하여 가장 좋은 알고리즘을 선택하여 컴퓨터를 실행하는 것은 컴퓨터를 효율적으로 이용하는데 있어서 가장 바람직한 일이다. 순차적인 컴퓨터에서는 산술연산의 수가 가장 적은 것이 고려 되지만 병렬 컴퓨터에서는 가용의 많은 option들이 있으며 가장 좋은 방법을 선택하는 것이 그렇게 쉽지는 않다.

미지수가 n인 n개의 방정식으로 된 m개의 tridiagonal system에 대해서 알고리즘을 선택하는데는 2가지 방법이 고려될 수 있다.

첫째는 가장 좋은 알고리즘을 찾아서 그들을 m개의 시스템에다 병렬로 응용하는 방법이고, 둘째는 하나의 단일 시스템을 해석하는데 가장 좋은 병렬 알고리즘을 찾는 방법이다.

그리고 알고리즘을 비교하는 방법에는 여러가지가 있겠지만 일반적으로 performance를 가지고 비교하는 경우가 많다.

performance $\propto t^{-1}$의 관계를 고려하여 위에서 후자의 방법을 선택한다면 식 5-3, 5-6 그리고 recursive doubling 기법에 대한 알고리즘 수행시간 t_{rd}($t_{rd}=24n\log_2 n$)[9][14][16]로부터 기존의 block cyclic reduction 기법(ECR)과 변형된 block cyclic reduction 기법(MCR) 그리고 recursive doubling 기법(RD)에 대하여 performance를 계산하면

(i) $P_{RD}\propto[24n\log_2 n]^{-1}$
(ii) $P_{ECR}\propto[16n-11\log_2 n]^{-1}$
(iii) $P_{MCR}\propto[11\log_2 n]^{-1}$

로 되고 이들을 비교해보면 recursive doubling 기법은 block cyclic reduction의 어느 경우보다도 performance가 떨어지며, 기존의 block cyclic reduction기법보다는 변형된 block cyclic reduction 기법의 performance가 더 높은 것을 알 수 있다.

일반적으로 방정식의 수 n보다 프로세서의 수 p가 아주 작은 경우 ($p\ll n$경우) partition 알고리즘이 사용되면 프로세서는 최대의 효율로 동작하고[9], recursive doubling과 cyclic reduction 기법은 $n=2^q$(여기서 $q=\log_2 n$)개의 방정식을 해석할때 가장 효율적이다.[9] 그리고 order n의 tridiagonal system을 cyclic reduction 기법으로 해석하면 $\log_2 n$ step이 걸리며 i번째 step에서 vector operand는 기억장치 내에서 2^i words 만큼 떨어져 있게 된다. 이로 인하여 data rate는 bank conflict 때문에 감소하고 vector operation 에서 potential 지연이 생긴다.

그러나 partition mehtod는 bank conflict가 생기지 않도록 항상 block의 수를 조정할 수 있다. 따라서 block cyclic reduction의 경우에도 matrix을 각 submatrices로 분할할때 block의 수를 조정하면 vector 연산에서 potential 지연을 줄일수 있을 것이다.

Ⅵ. 결 론

선형2계 편미분 방정식의 일반식을 유한차분법에 의하여 선형 방정식으로 변환 하였을때 계수 matrix Q는 대단히 sparse한 시스템으로 되고 이를 submatrix로 분할하면 tridiagonal system으로서 diagonally dominant system이 된다. 그리고 이 tridiagonal system을 block cyclic reduction 알고리즘에 의하여 reduction하면 $\log_2 n-1$ step 후에는 하나의 block으로 된다. reduction 과정에서 병렬성과 산술 연산의 수는 매 reduction 때마다 2^r(여기서 r은 reduction 수)에 비례하여 감소한다. 그러나 back substitutuion 과정에서는 이들은 2^r(여기서 r은 substition 수)에 비례하여 증가한다. 이에 반하여 변형된 block cyclic reduction 알고리즘은 병렬성과 산술 연산수는 reduction에 관계 없이 일정하며 back substitution 과정이 필요가 없고 reduction 의 마지막 단계에서 모든 미지수가 동시에 구해진다.

따라서 n−1개의 프로세서를 사용하여 병렬로 실행시킨다면 실행시간도 변형된 block cyclic reduction 알고리즘이 기존의 block cyclic reduction 알고리즘에 비하여 훨씬 단축 될 수 있다.

일반적으로 병렬 컴퓨터에는 산술 연산 수보다는 병렬성이 큰 알고리즘이 바람직 하므로 병렬성이 작고 변하는 기존의 block cyclic reduction 알고리즘 보다는 병렬성이 크고 일정하며 실행시간이 더 단축되는 변형된 block cyclic reduction 알고리즘이 선형2계 편미분 방정식의 解 알고리즘으로 효율적인 알고리즘이 될 수 있다.

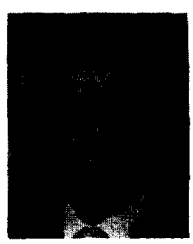
참 고 문 헌

1. B.L. Buzbee, G.H. Golub, and C.W. Nielson, "On direct methods for solving poisson's equation", SIAM J.Numer. Anal, Vol.7, No.4, pp.627-656, DEC.1970.

2. H.S. Stone "An efficient parallel algorithm for the solution of a tridiagonal linear system of equations", Journal of the association for computing machinery,

Vol.20, No.1, pp.27-37, Jan.1973.
3. G.R. Gao, "A pipelined solution method of tridiagonal linear equation system", Proc. of Int'l. conf. parallel processing, pp.84-91, AUG.1986.
4. O.Wing, and J.W. Huang, "A computation model of parallel solution of linear equations", IEEE Trans, Comput, Vol.c-29, No.7, pp.632-638, JULY. 1980.
5. C.P. Arnold, M.I. Parr, and M.B. Dewe, "An efficient parallel algorithm for the solution of large sparse linear matrix equations", IEEE Trans, Comput, Vol.C-32, No.3, pp.265-276, March. 1983.
6. Don Heller, "A survey of parallel algorithms in numerical linear algebra", SIAM REV, Vol.20, pp.740-777, OCT.1978.
7. J.W. Ortega, and R.G.Voigt, "Solution of partial differential equations on vector and parallel computers", SIAM REW., Vol.27, No.2, pp.149-240, JUNE. 1985.
8. Y.Wallach, and V.Konrad, "On block-parallel methods for solving linear equations", IEEE Trans, Comput, Vol.C-29, No.5, pp.354-359, May.1980.
9. H.H. Wang, "A parallel method for tridiagonal equations", ACM Trans. on Math, Soft, Vol.7, No.2, June. 1981.
10. Done Heller, "Some aspects of the cyclic reduction algorithm for block tridiagonal linear system", SIAM J. Numer, Anal, Vol.13, No.4, pp.484-496, Sept.1976.
11. Alan George, "On block elimination for sparse linear system", SIAM J.Numer. Anal, Vol.11 No.3 pp.585-603, June.1974.
12. S.C. Chen, D.J.Kuck, and A.H. Sameh, "Practical parallel band triangular system solvers", ACM Trans, on Math, Soft, Vol.4, No.3, pp.270-277, Sept.1978.
13. D.J. Evans, and M.Hatzopoulos, "A parallel linear system solvor", Intern.J. Computer Math, Vol.7, pp. 227-238, 1979.
14. D.J. Evans, and S.O. Okolie, "A recursive decoupling algorithm for solving banded linear systems", Int. JOUR.Comp. Math. Vol.10, pp.139-152, 1981.
15. D.P. Bertsekas and J.N. Tsitsiklis, Parallel and Distributed Computatuion, Printice-Hall, inc.
16. L.Lapidus and G.F. Prinder, Numerical solution of partial Differential equations in science and Engineering, John Wiley & sons, inc. 99.35-48. 1982.

李 秉 洪(Byeong Hong LEE) 正會員
1946年12月6日生
1970年2月：韓國航空大學通信工學科卒業(學士)
1982年2月：東國大學校大學院電子工學科 卒業(碩士)
1988年9月～現在：韓國航空大學大學院電子工學科 博士課程
1974年11月～1979年2月：麗水水産專門大學 專任講師
1979年3月～現在：烏山專門大學 教授



金 正 善(Jeong Sun KIM) 正會員
1941年5月5日生
1965年～1990年 現在：航空大學 教授
1988年～1990年：電子工學科 學科長
1990年2月～1990年6月：電子計算所長