병렬화된 고속 포아송 방정식의 예측모델에의 적용

송 창 근† · 이 상 덕††

요 약

본 연구에서는 격자점의 갯수나 경계 조건에 관계없이 포아송 방정식를 푸는 일반적인 프로그램을 개발하고, 수퍼 컴퓨터의 병렬 기능과 벡터 기능을 이용하여 이 프로그램을 고속화시켰다. 우리는 실제 현업에 사용되고 있는 바로트로픽 예측 모델을 이용하여 실제 태풍인 Elena의 궤도를 예측하여 보았고, 병렬화된 고속의 포아송 방정식을 사용하는 경우 상당한 시간이 절약됨을 알 수 있었다. 72시간 후의 허리케인의 궤도 예측을 시도하였다. 3000여개의 격자점 위에서 시간 간격을 16분으로 하여 실험하였는데, 8개 벡터 프로세서를 갖고 있는 Alliant FX/8에서 30초만에 이루어 졌고, 3.7의 계산 효율을 얻어냈다.

Application of a Fast Parallel Poisson Solver to Barotropic Prediction Model

Chang Geun Song† · Sang Duck Lee††

ABSTRACT

In this paper, we develop the code, called the fast parallel Poisson solver, which solves the Poisson's equation of arbitrary dimension and parallelize it. And we apply the fast parallel Poisson solver to the barotropic prediction model to explore the advantages of using it. In particular, we apply this model to the track forecasting of hurricane Elena(1985) and demonstrate that the fast parallel Poisson solver significantly reduces the computational time required to integrate the barotropic model. A 72-h track prediction was made by using time step of 16 minutes on a network of about 3000 grid points. The prediction took 30 seconds on the 8-processor Alliant FX/8 mini supercomputer. It was a speed-up of 3.7 when compared to the one-processor version.

1. 서 론

대기의 상태에 관한 예측은 대기의 운동을 설명하는 수치 모델인 미분 방정식을 적분하므로써 얻어진다. 이러한 수치 모델에 대한 적분을 수치 예보(numerical weather prediction)라고 부른다. 태풍이나 허리케인의 궤도를 추적하여 예측하는 것도 수치 예보에 해당되며, 예측 모델로서는 바로트로픽(barotropic) 모델이 오래 전부터 사용되어 왔다[10, 12]. 벡터 컴퓨터 혹은 병렬 컴퓨터로 대표되는 고성능의 수퍼 컴퓨터의 개발은 수치 예보를 통하여 미래의 상태에 대한 좀 더 빠르고 정확한 예측을 가능하게 하였다.

컴퓨터를 이용하여 미분 방정식을 푸는 방법으로는 유한 차분법(finite difference method)이 자주 사용된다[1]. 이 방법은 미분 방정식에 나타나는 미분항을 그에 대응하는 근사식으로 치환하여 대수적인 연립

† 정 회 원:한림대학교 컴퓨터공학과 교수
†† 준 회 원:한림대학교 컴퓨터공학과
논문접수:1996년 2월 29일, 심사완료:1997년 1월 29일

방정식을 얻는 것이다. 즉, 계산 영역 안에 격자점들이 설정되고, 이러한 점들 위에 종속 변수들이 정의된다. 근사식은 격자점 위에서 정의되는 종속 변수들로 정의된다. 초기 시점과 경계선 상에 있는 격자점에서의 값들은 주로 관측에 의하여 얻게 되는데, 미분방정식이 해를 갖기 위해서는 초기 조건(initial condition)이나 경계 조건(boundary condition)들을 근사하는 추가적인 수식이 필요하다.

포아송 방정식(Poisson's equation)은 과학이나 공학 분야의 수치 모델에 자주 사용되고 있으며, 본 연구에서 사용한 바로트로픽 모델에서도 역시 사용되고 있다. 포아송 방정식은 주어지는 경계 조건에 따라 Dirichlet, Neumann 그리고 Periodic 문제로 구별된다. 포아송 방정식을 유한 차분법을 이용하여 이산화하면 $Qx=b$꼴의 선형 연립 방정식를 얻게 된다. 이러한 선형 연립 방정식를 푸는 방법들은 SOR(successive over-relaxation)과 ADI(alternating direction implicit)방법과 같은 반복법(iterative method)과 BCR (block cyclic reduction)과 같은 직접법(direct method)으로 대별 된다[3, 17].

이산화된 포아송 방정식의 해를 구하는 방법은 포아송 방정식의 중요성 때문에 오래 전부터 꾸준히 연구가 이루어져 왔다[3, 4, 14, 15, 16, 17, 18]. 특히, Hockney에 의하여 최초로 제안된 BCR 방법은 수치적으로 안정되지 못 하였으나 그 후 곧 안정적인 방법으로 개선되었다[3]. 그리고, 초기에 제안된 BCR 방법은 계산 영역 안의 격자점의 갯수가 2^k-1 꼴의 특수한 경우에만 사용되었다. Sweet는 격자점의 갯수에 관계없는 일반적인 BCR 알고리즘을 개발하였다[15]. 그러나, BCR 알고리즘은 고유의 자료 종속성 때문에 병렬 컴퓨터에 효율적으로 구현되지 못한다. 따라서, 최근에는 포아송 방정식을 수퍼 컴퓨터의 벡터 기능과 병렬 기능을 이용하여 고속으로 해를 구하려는 시도가 이루어 지고 있다[4, 11, 16, 18].

본 연구에서는 포아송 방정식의 해를 구하는 BCR 알고리즘을 연구 분야에서 널리 사용되고 있는 미니 수퍼컴퓨터인 Alliant FX/8에 구현한다. 이 때, Alliant의 병렬처리 기법을 이용하여 방정식의 해를 고속으로 계산하고자 한다. 이 프로그램은 Dirichlet 문제와 Neumann 문제들을 격자점의 갯수가 2^k-1 꼴인 경우뿐만 아니라 임의의 갯수의 격자점 위에서 풀 수 있는 일반적인 것이다. 이렇게 개발된 고속의 병렬화된 포아송 방정식의 해를 태풍이나 허리케인과 같은 대규모 기상 현상에 대한 예측에 적용하여 그 이득을 실제로 확인하여 본다.

한편, 예측에는 신속한 처리 뿐만 아니라 정확함이 필수적이다. 예측이 정확하기 위해서는 예측 모델의 초기 조건의 정확성이 매우 중요하다. 초기 조건은 주로 관측에 의하여 얻어지는 데, 물리적인 한계로 초기 관측점이 부족한 경우와 관측시 발생하는 관측 에러 등의 이유 때문에 초기값들이 부정확하다. 최근까지 이러한 예측 모델의 초기 조건을 개선하려는 노력이 많이 있어 왔는데, 이 중에는 자료 동화 방법(data assimilation)이 있다[8]. 이 방법은 예측(forecasting)이 얼마간 진행된 후에 중간 시점에 얻어진 관측 자료(observed data)와 모델에서 생성된 과거의 예측 자료(forecast data)의 차이를 최소화하도록 초기 조건을 개선하여, 좀더 나은 예측을 얻으려는 시도이다. 그러나 이러한 자료 동화 방법은 매우 많은 계산량을 요구하므로, 이에 대한 고속의 처리가 필수적이다. 본 연구는 이러한 자료 동화 방법의 전 단계 연구라고 할 수 있다.

2장에서는 전통적인 BCR 알고리즘에 대하여 소개하며, 이 알고리즘을 수퍼 컴퓨터에 적합하도록 병렬화시키는 방법에 대하여 논의한다. 3장에서는 예측 모델인 바로트로픽 모델의 이산화와 안정화에 대하여 논의한다. 특별히 바로트로픽 모델에서는 비선형항(nonlinear term)이 있으므로 장시간의 수치적 적분은 비선형적 불안정성(numerical nonlinear instability)을 초래할 수 있다[2]. 이러한 수치적 불안정성으로 인하여 바로트로픽 모델의 장시간 사용이 제한을 받는다. 이에 대한 개선책으로 Arakawa의 Jacobian항을 사용하는 데, 이러한 Jacobian항의 사용 효과를 실험적으로 확인한다. 4장에서는 개발된 포아송 방정식에 대한 프로그램의 성능을 분석한다. 또한, 바로트로픽 모델을 이용하여 북미 지역에서 실제 발생한 허리케인인 Elena의 궤도 예측에 적용하여 그 결과를 분석한다. BCR 방법 뿐만 아니라 예측 프로그램도 Alliant의 벡터 기능과 병렬 기능을 이용하여 고속화한다. 마지막으로 5장에서는 결론과 향후 연구 과제를 기술한다.

2. 포아송 편미분 방정식의 고속해

2.1 포아송 편미분 방정식(Poisson's partial differential equation)

포아송 방정식을 매우 빠르고 정확하게 푸는 것은 바로트로픽 예측 모델의 실용성에 매우 중요하다. 그 이유는 예측 모델을 사용할 때, 매 시간 간격당 포아송 미분 방정식을 한 번씩 풀어야 하므로 장시간 후의 궤도를 예측할 경우 매우 많은 포아송 방정식을 풀어야 하기 때문이다. 물론 시간 간격을 늘리면, 풀어야 하는 포아송 방정식의 갯수를 줄일 수 있으나, 3장에서 볼 수 있듯이 수치적 안정성 때문에 시간 간격의 크기를 임의로 늘릴 수가 없다. 포아송 방정식은 다음과 같은 미분 방정식을 말한다.

$$\nabla^2 u(x, y) = f(x, y). \tag{2.1}$$

포아송 방정식의 해를 u(x, y)라고 하고, 함수 f(x, y)가 직사각형 모양의 2 차원 평면 상에 정의되어 있다고 하자. 이 때, 주어진 경계 조건에 따라 포아송 방정식은 분류된다. 경계선 위에서 함수값이 주어졌을 때, 포아송 방정식을 푸는 문제를 Dirichlet 유형의 문제라고 하고, 경계선 위에서 수직 방향의 1차 도함수가 주어지는 문제를 Neumann 유형의 문제라고 부른다. 한편, u(x, y)가 x축 방향이나, y축 방향 혹은 양방향으로 주기적이면, periodic 유형의 문제라고 부른다. 포아송 방정식을 격자선 위에다 유한 차분법으로 이산화시키면, Qx=b꼴의 선형 연립 방정식을 얻게 된다. 주어진 경계 조건에 따라 계수 행렬 Q의 모양이 달라진다. Dirichlet 유형의 문제의 경우를 예를 들어 설명하면 다음과 같다.

$$Qx = b \tag{2.2}$$

$$Q=\begin{bmatrix} A & -I & & & \\ -I & A & -I & & \\ & -I & A & -I & \\ & & \ddots & \ddots & \ddots \\ & & & -I & A \end{bmatrix} \text{ 이고, } A=\begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 4 \end{bmatrix} \tag{2.3}$$

이 때, Q의 각 원소 A와 I는 M×M 삼중 대각 행렬이고, Q 자체는 N×N의 차원을 갖는 블록 삼 중 대각 행렬(block tridiagonal matrix)이다. 처음으로 제안된 BCR 방법은 경계선 상의 격자점의 갯수에 해당되는 N이나, M이 2^k-1 꼴로 제한되는 것이었다. 최근에 Sweet에 의해 격자점의 갯수에 대한 제한이 해결되었다[15].

2.2 BCR 알고리즘.

설명을 간단히 하기 위해서 N이 2^k-1 꼴의 경우에 대하여 생각하며, 초기에 다음과 같은 선형 방정식을 갖는다고 가정하자.

$$\begin{bmatrix} A & -I & & & \\ -I & A & -I & & \\ & -I & A & -I & \\ & & \ddots & \ddots & \ddots \\ & & & -I & A \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_N \end{bmatrix} \tag{2.4}$$

여기서, 벡터 u_i와 f_i는 크기가 M인 벡터이다. BCR 알고리즘은 소거 과정(reduction phase)과 회복과정(back-substitution phase)의 두 과정으로 구성된다. 먼저 소거 과정을 설명한다. 소거 과정의 r 번째 단계(step)가 진행한 후에는 식(2.5)와 같은 선형 연립 방정식을 갖게 된다. A(r)은 r번째 소거 단계에서의 블록 행렬을 나타낸다.

$$\begin{bmatrix} A(r) & -I & & & \\ -I & A(r) & -I & & \\ & -I & A(r) & -I & \\ & & \ddots & \ddots & \ddots \\ & & & -I & A(r) \end{bmatrix} \begin{bmatrix} u_h \\ u_{2h} \\ u_{3h} \\ \vdots \\ u_{n_r} \end{bmatrix} = \begin{bmatrix} A(r)\,p_h(r) + q_h(r) \\ A(r)\,p_{2h}(r) + q_{2h}(r) \\ A(r)\,p_{3h}(r) + q_{3h}(r) \\ \vdots \\ A(r)\,p_{n_r-h}(r) + q_{n_r-h}(r) \\ A(r)\,p_{n_r}(r) + q_{n_r}(r) \end{bmatrix} \tag{2.5}$$

여기서, $h=2_r$이고, $N=n_rh$이다. 초기에는 ($r=0$의 경우) $A(0)=A$, $C(0)=I$, $n_0=N$, $p_j(0)=0$, $q_j(0)=f_j$ $j=1, 2, ..., N$ 이다. 다음의 $r+1$ 단계에서는 짝수인 $2jh$ 꼴의 첨자를 갖는 변수들을 가진 식에 $A(r)$를 곱한 후, 인접한 두 식들을 더함으로써, $2jh$꼴이 아닌 변수들 $u_{(2j-1)h}$와 $u_{(2j+1)h}$을 소거한다. 다시 말하면,

$$-u_{(2j-2)h}+((A(r)^2-2I)-u_{(2j+2)h}$$
$$=(A(r)^2p_{2jh}(r)+A(r)(q_{2jh}(r)+p_{(2j-1)h}(r)+p_{(2j+1)h}(r))$$
$$+q_{(2j-1)h}(r)+q_{(2h+1)h}(r)$$

이 과정을 수행하면, h의 짝수 배에 해당하는 식만을 남게 된다. $r+1$번째 소거 단계가 끝나고 나면, 그 구조는 (2.5)식과 같으나 $A(r)$, $p(r)$, $q(r)$ 대신에 $A(r+1)$, $p(r+1)$, $q(r+1)$로 각각 대치한 같은 형태의 선형 연립 방정식을 얻게 된다. 단, r 단계에서의 미지수 벡터의 갯수보다 $r+1$단계에서는 미지수 벡터의 갯수가 반으로 줄어 들게 된다. 위의 과정을 종합하여 보면,

$$A(r+1)=(A(r))^2-2I \text{ 이고,} \tag{2.6}$$

또한, $j=2h, 4h, ..., n_r$에 대해서

$$p_{2jh}(r+1)=p_{2jh}(r)+(A(r))^{-1}(q_{2jh}(r)+p_{(2j-1)h}(r)+p_{(2j+1)h}(r)) \tag{2.7}$$

$$q_{2jh}(r+1)=q_{(2j-1)h}(r)+q_{(2j+1)h}(r)+2p_{2jh}(r+1) \tag{2.8}$$

한편, $A(r)=2\,T_2^r(A/2)$임이 알려져 있다[15]. $T_n(x)$는 첫째 종류의 체비세프 다항식(Chebyshev Polynomial of the first kind)이다. 소거 단계를 $k-1$ 번째 수행하고 나면, 결국에는 1개의 방정식을 갖게 된다. 이 방정식은 LU 분해법으로 해를 구할 수 있다. 두 번째인 회복 과정 역시 소거 과정과 마찬가지로 여러 단계로 구성된다. 소거 단계에서 소거되었던 변수 u_{jh}의 값을 구하는 것으로, j값이 짝수인 첨자를 갖는 u 벡터가 이전 단계에서 구해졌다고 가정하고, j값이 홀수인 첨자를 갖는 u벡터를 구한다. 즉, $jh \le N$이고, 홀수값 j에 대하여 다음 식을 이용하여 벡터 u_{jh}를 구한다.

$$u_{jh}=p_{jh}(r)+(A(r))^{-1}(q_{jh}(r)+u_{(j-1)h}+u_{(j+1)} \tag{2.9}$$

위의 회복 과정은 h 값이 2^{k-1} 부터 1이 될 때까지 반복해서 수행된다. 따라서 BCR 알고리즘은 다음과 같이 요약된다.

[알고리즘 2.1]

1). $p_j(0)=0$, $q_j(0)=f_j$, $j=1, 2, ..., N$ 그리고 $h=1$, $r=0$로 초기화한다.

2). 소거 과정(Reduction phase)

2.1) 열들이 $q_{2jh}(r)+p_{(2j-1)h}(r)+p_{(2j+1)h}(r)$, $j=1, ..., \frac{N}{2h}$ 인 행렬 Y_r을 구성한다.

2.2) $A(r)X_r=Y_r$인 선형 연립 방정식를 푼다.

2.3)식(2.7)과 식(2.8)에 따라서 벡터 p와 q를 구한다.

2.4) $h \le N$이면 $h:=2h$, $r:=r+1$하고, 2.1) 단계로 제어를 이동한다.

3). $A(r)v=q_h(r)$을 풀고, $u_h:=p_h+v$.

4). 회복 과정(Backsubstitution phase)

$h \ge 1$ 일 때 다음의 과정을 반복한다.

4.1) $h:=h/2$, $r:=r-1$

4.2) 열들이 $q_{jh}(r)+u_{(j-1)h}(r)+u_{(j+1)h}(r)$, $j=1, 3, 5, ..., N/1$인 행렬 Y_r을 구성한다.

4.3) $A(r)U_r=Y_r$인 선형 연립 방정식를 푼다.

4.4)식(2.9)에 따라서 벡터 u_{jh}, $j=1, 3, ..., [\frac{N}{h}]$을 구한다.

2.3 BCR 방법의 병렬화를 통한 고속화

기존의 전통적인 방법인 BCR 법은 Serial Bottleneck이라고 불리우는 문제를 안고 있다[7]. 최근 이러한 문제를 해결하기 위한 방법들이 Sweet와 Gallopoulous에 의해 제안되었다[4, 16, 18]. Serial Bottlenck은 다음과 같은 경우를 말한다. BCR 방법은 [알고리즘 2.1]의 단계 2.2), 3) 그리고 4.3)에서 다음과 같은 형태의 선형 방정식을 풀어야 한다.

$$A(r)X=Y$$

여기서, $A(r)$은 차수가 $2r$인 다항식이라고 하자. X, Y는 벡터이다. 또, $A(r)$이 다음과 같이 인수 분해 된다고 가정하자.

$$A(r)=\prod_{i=1}^{2^r}(A-\lambda_i(r)) \tag{2.10}$$

기존의 전통적인 방법은 [알고리즘 2.2]를 이용하여 식 (2.10)을 풀게 된다.

[알고리즘 2.2] (old method)
1). $Z_0=Y$
2). FOR j=1 TO 2^r STEP 1 DO
    $(A-\lambda_j(r)\ I)\ Z_j=Z_{j-1}$
    ENDDO
3). $X=Z_{2^r}$

[알고리즘 2.2]의 FOR 루프에서는 자료의 종속성이 존재하므로, 여러 개의 프로세서들이 사용되는 경우에도 수행 시간이 단축되지 않는다. 이것은 〈표 4.1〉에서 확인된다. 한편 식 (2.10)에 의하면,

$(A(r))^{-1}=\sum_{i=1}^{2^r}\frac{\alpha_i}{(A-\lambda_i)}$ 이다.

Partial Fraction expansion에 기초하여 병렬 컴퓨터에 알맞게 개선된 새로운 알고리즘은 다음과 같다.

[알고리즘 2.3] (new method)
1). FOR j=1 TO 2^r STEP 1 DO in parallel
    $(A-\lambda_j(r)\ I)\ Z_j=Y$
    ENDDO
2). $X=\sum_{i=1}^{2^r}\alpha_i Z_i$

여기서 $\alpha_i=\frac{(-1)^{i-1}}{2^r}\sin\frac{(2i-1)\pi}{2^r+1}$ 이다[4]. [알고리즘 2.3]에서는 계수 α_i를 구하는 것이 매우 중요한데, Dirichlet 문제의 경우는 Gallopoulous와 Saad[4]에 의하여 분석되어 있고, Neumann 문제의 경우는 저자에 의해 잘 분석되어 있다[18].

3. 바로트로픽 예측 모델(Barotropic Prediction Model)

바로트로픽 예측 모델은 다음과 같이 기술할 수 있다.

$$\frac{\partial\zeta}{\partial t}+\beta\frac{\partial\psi}{\partial x}-\frac{\partial\psi}{\partial y}\frac{\partial\zeta}{\partial x}+\frac{\partial\psi}{\partial x}\frac{\partial\zeta}{\partial y}=\varepsilon\nabla^2 \tag{3.1}$$

그리고,

$$\nabla^2\psi=\zeta \tag{3.2}$$

이다. 여기서, ψ는 streamfunction이고, ζ는 와도(vorticity)를 의미하며, ε는 eddy-viscosity 계수로서 frictional dissipation을 나타낸다. 또한 β는 Coriolis 매개변수를 의미한다. 바람의 X 축 방향, u와 Y축 방향 v는 다음과 같이 주어진다.

$u=-\frac{\partial\psi}{\partial y}$ 그리고 $v=\frac{\partial\psi}{\partial x}$.

태풍의 궤도를 예측하는 과정은 [알고리즘 2.4]와 같다.

[알고리즘 2.4]
1). 초기 와도, $\zeta^{(0)}$를 구한다
2). 포아송 방정식인 (3.2)식을 풀어서 초기 streamfunction $\psi^{(0)}$을 구한다.
3). (3.1)식을 풀어서 다음 단계의 와도 $\zeta^{(i)}$을 구한다.
4). (3.2)식을 풀어서 $\psi^{(i)}$을 구한다.
5). 단계 3)과 4)를 예측 기간까지 반복한다.

바로트로픽 예측에서는 경계선에서 함수값이 주어지는 Dirichlet 유형만을 취급하고자 한다. 2)와 4)단계에서는 포아송 방정식을 풀어야 하는 데, 장기간의 예측인 경우 매우 많은 포아송 방정식을 풀어야 한다. (예를 들어 72시간 예측을 하는 경우, 시간 간격을 4분으로 할 때, 1080번의 포아송 방정식을 풀어야 한다.) 전체 시간을 줄이기 위해서는 단계 2)와 4)에 해당하는 포아송 방정식의 해법을 고속화시킬 필요가 있다.

(3.1)식을 이산화할 때, 시간에 대해서는 2차 Adams-Bashforth 방법을 적용하고, 공간에 대해서는 Centered differencing 방법을 사용하였다. 따라서 이산화하여 얻어진 식은 다음과 같다.

$\zeta^1_{i,j}=\zeta^0_{i,j}+\Delta_t F^0_i$, 와

$$\zeta^n_{i,j}=\zeta^{n-1}_{i,j}+\frac{3\Delta_t}{2}F^n_{i,j}-\frac{\Delta_t}{2}F^{n-1}_{i,j} \quad \text{for } n\geq 1 \tag{3.3}$$

여기서

$$F^n_{i,j}=-\frac{\beta}{2\Delta_x}(\psi^n_{i+1,j}-\psi^n_{i-1,j})-J^n_{i,j}(\psi,\zeta)-D^n_{i,j} \tag{3.4}$$

$$D^n_{i,j}=\frac{\varepsilon}{\Delta_x^2}(\zeta^n_{i+1,j}-2\zeta^n_{i,j}+\zeta^n_{i-1,j})$$
$$-\frac{\varepsilon}{\Delta_y^2}(\zeta^n_{i,j+1}-2\zeta^n_{i,j}+\zeta^n_{i,j-1}) \tag{3.5}$$

$$J^n(\psi,\zeta)=-\frac{1}{4\Delta_x\Delta_y}\{(\psi^n_{i,j+1}-\psi^n_{i,j-1})(\zeta^n_{i+1,j}-\zeta^n_{i-1,j})$$
$$-(\psi^n_{i+1,j}-\psi^n_{i-1,j})(\zeta^n_{i,j+1}-\zeta^n_{i,j-1})\} \tag{3.6}$$

DeMaria[9]의 분석에 의하면, Adams-Bashforth 방법은 아래와 같은 CFL 조건을 만족하는 범위 내에서는 수치적으로 안정되게 사용할 수 있다.

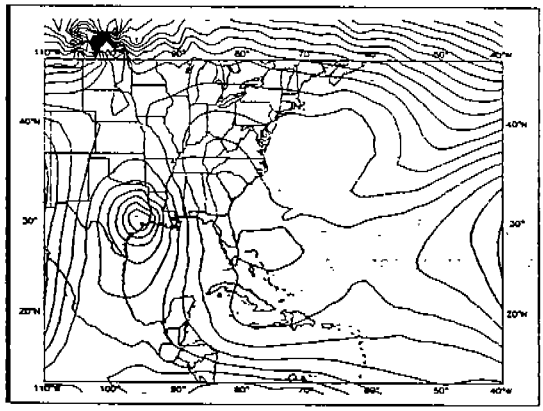
$$\Delta_t\leq\frac{\Delta_x}{2\sqrt{2}\,V_{max}}. \tag{3.7}$$

여기서 V_{max}는 최대의 풍속이다. 한편 식 (3.1)에서 $-\frac{\partial\psi}{\partial y}\frac{\partial\zeta}{\partial x}+\frac{\partial\psi}{\partial x}\frac{\partial\zeta}{\partial y}$ 항은 비선형항(nonlinear)으로서 Jacobian항이라고도 불리운다. 이것은 보통 식 (3.6)과 같이 이산화된다. 그러나, 태풍의 진로를 장시간 예측하게 되면, Jacobian 항으로 인한 비선형성 수치적 불안정성이 우려된다. (3.6)의 이산화 항을 사용하여 [알고리즘 2.4]를 1650번 반복한 경우(약 110 시간에 해당) (그림 3.1)의 좌측 상단에서 볼 수 있듯이 수치적으로 불안정한 결과를 주게 된다. 실제로 113 시간 예측 이후에는 overflow error로 프로그램이 종료된다. 그러나, Jacobian 항은 다음과 같이 3가지 서로 다른 방식으로 정의될 수 있다.

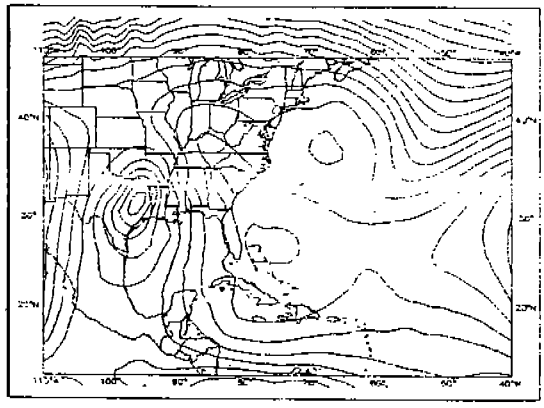
$$J(\psi,\zeta)=-\frac{\partial\psi}{\partial y}\frac{\partial\zeta}{\partial x}+\frac{\partial\psi}{\partial x}\frac{\partial\zeta}{\partial y}$$
$$=-\frac{\partial}{\partial x}(\zeta\frac{\partial\psi}{\partial y})+\frac{\partial}{\partial y}(\zeta\frac{\partial\psi}{\partial x})$$
$$=-\frac{\partial}{\partial y}(\psi\frac{\partial\zeta}{\partial x})+\frac{\partial}{\partial x}(\psi\frac{\partial\zeta}{\partial y})$$

이러한 3가지 정의들 각각을 이산화한 후, 그들의 평균값으로 Jacobian 항을 재정의하면, 식(3.8)와 같다. 이것은 Arakawa[2]가 제안한 방법과 같음을 알 수 있다. 이렇게 새롭게 정의된 Jacobian 항을 도입하여 사용하면, 수치적으로 매우 안정된 결과를 얻게 된다. 이것은 (그림 3.2)에서 알 수 있듯이 같은 110시간 후의 예측에서 식(3.8)의 이산화 항을 사용하는 경우에는 매우 안정적인 결과를 준다.

$$J^n_{i,j}(\psi,\zeta)=\frac{1}{12\Delta_x\Delta_y}\{(\psi^n_{i,j+1}-\psi^n_{i,j-1})(\zeta^n_{i+1,j}-\zeta^n_{i-1,j})$$
$$-(\psi^n_{i+1,j}-\psi^n_{i-1,j})(\zeta^n_{i,j+1}-\zeta^n_{i,j-1})$$
$$+\zeta^n_{i+1,j}(\psi^n_{i+1,j+1}-\psi^n_{i+1,j-1})-\zeta^n_{i-1,j}(\psi^n_{i-1,j+1}-\psi^n_{i-1,j-1})$$
$$-\zeta^n_{i,j+1}(\psi^n_{i+1,j+1}-\psi^n_{i-1,j+1})-\zeta^n_{i,j-1}(\psi^n_{i+1,j-1}-\psi^n_{i-1,j-1})$$



(그림 3.2) 식(3.8)을 이용한 113시간 후의 예측
(Fig. 3.2) The forecast after 113 hours using equation (3.8)



(그림 3.1) 식(3.6)을 이용한 113시간 후의 예측
(Fig. 3.1) The forecast after 113 hours using equation (3.6)

$$+\psi^n_{i,j+1}(\zeta^n_{i+1,j+1}-\zeta^n_{i-1,j+1})-\psi^n_{i,j-1}(\zeta^n_{i+1,j-1}-\zeta^n_{i-1,j-1})$$

$$+\psi^n_{i+1,j}(\zeta^n_{i+1,j+1}-\zeta^n_{i+1,j-1})-\psi^n_{i-1,j}(\zeta^n_{i-1,j+1}-\zeta^n_{i-1,j-1})\}$$

(3.8)

4. 실험 및 결과.

4.1 고속 포아송 방정식의 구현

[알고리즘 2.1]을 [알고리즘 2.2]와 [알고리즘 2.3]을 이용하는 두 가지 경우에 대하여 실험하였다. 가로 혹은 세로 방향의 격자점의 갯수가 같은 경우 즉, M×M인 경우에 대하여 실험한다. 프로세서의 벡터 기능이나 병렬 기능을 사용하거나, 혹은 두 가지 기능들을 모두 사용하면서 포아송 방정식을 풀어 본 결과가 〈표 4.1〉와 〈표 4.2〉에 요약되어 있다. 〈표 4.1〉과 〈표 4.2〉에서는 Dirichlet 문제를 M이 31, 63, 127인 세 가지 경우에 대하여 계산 시간, 계산 효율과 잔차(residual)를 보여 준다. 격자점의 갯수가 다른 경우에도 매우 유사한 결과를 얻는다. 또한, Neumann문제의 경우에도 매우 유사한 결과를 얻는다. 〈표 4.1〉에서 Serial은 1개의 프로세서를 사용하되 그 프로세서의 벡터 연산 기능을 이용하지 않고 BCR 알고리즘을 수행한 것이고, 〈표 4.1〉에서 프로세서의 개수가 1인 경우는 그 프로세서의 벡터 연산 기능을 사용하여 BCR 알고리즘을 수행한 차이를 갖는다.

〈표 4.1〉 [알고리즘 2.2]와 [알고리즘 2.3]을 이용한 BCR의 수행시간

〈Table 4.1〉 The excution times of BCR using [algorithm 2.2] and [algorithm 2.3]

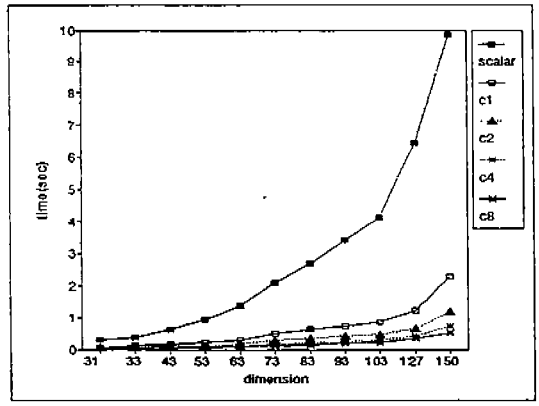
차원	프로세서의 개수	알고리즘 2.2 시간 (초)	계산 효율	알고리즘 2.3 시간 (초)	계산 효율
31	Serial	0.1947498		0.3004103	
	1	8.1600189E-02	1.0	7.9500198E-02	1.0
	8	8.4280014E-02	0.97	3.6911001E-02	2.2
63	Serial	0.9004517		1.398163	
	1	0.3334732	1.0	0.2992401	1.0
	8	0.3349609	1.0	9.7190857E-02	3.1
127	Serial	4.083286		6.443832	
	1	1.390976	1.0	1.225998	1.0
	8	1.364838	1.0	0.3482666	3.5

〈표 4.2〉 [알고리즘 2.2]와 [알고리즘 2.3]을 이용한 경우의 ∞-Norm

〈Table 4.2〉 The ∞-Norm of BCR using [algorithm 2.2] and [algorithm 2.3]

차원	Alg. 2.2의 잔차의 노름	Alg. 2.3의 잔차의 노름
31	1.023181539494544E-012	1.080024958355352E-012
63	6.139089236967266E-012	5.002220859751105E-012
127	3.637978807091713E-011	5.366018740460277E-011

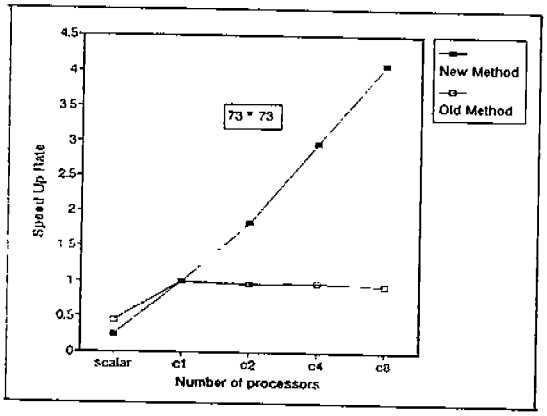
격자점의 수인 M을 31부터 151까지 증가시키면서 포아송 방정식을 푸는 데 걸리는 시간을 측정하였다. 그 실험한 결과가 (그림 4.1)에 요약되어 있다. (그림 4.1)에서는 우선 벡터화를 하지 않은 경우(Scalar)와 벡터 처리 후 프로세서의 갯수를 증가시키면서 측정한 결과를 나타낸다. 포아송 방정식을 Qx=y꼴의 선형 방정식으로 생각할 때, Q는 블록 삼중 대각 행렬이므로, 그 크기가 매우 큰 행렬이다. 예를 들어서 M이 31인 경우 Q의 차수는 312×312이다. 〈표 4.1〉에서 알 수 있듯이 전통적인 방법인 [알고리즘 2.2]를 사용하는 경우 비록 8개의 프로세서들을 사용하여도 시간이 줄어들지 않는다. 그러나 [알고리즘 2.3]을 사용하는 경우 격자점의 갯수에 관계없이 프로세서를 8개 사용하는 경우에 포아송 방정식을 푸는 시간이 현저히 줄어들었다. 구체적으로는 8개의 프로세서들을 사용하는 경우 Q의 크기에 따라 다소 다르지만 약 3배의 계산 효율을 얻는다. 특별히 Alliant의 벡터 기능을 사용하지 않는 경우에 비해서는 그 계산 속도가 매우 빠르다. 한편 (그림 4.2)는 M이 73인 경우, 1개의 벡



(그림 4.1) M×M 격자점에서의 포아송 방정식

(Fig. 4.1) The poisson equation discretized into M×M grid

터 프로세서를 사용하는 경우를 기준으로 하여 계산 효율(speed-up)을 보여 준다. 본 실험을 통해서 M이 31부터 150에 해당되는 경우에 M값에 상관없이 (그림 4.2)와 같은 형태의 계산 효율을 얻을 수 있었다.



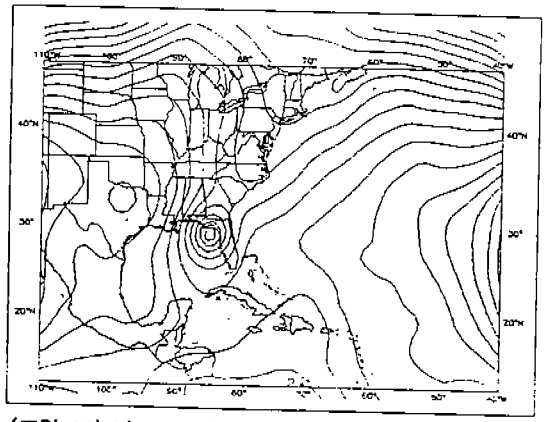
(그림 4.2) 73×73의 경우 계산 효율
(Fig. 4.2) The speed-up ratio in case M = 73

4.2 바로트로픽 예측 모델의 실험

실험을 위하여 실제로 1985년도 미국 플로리다 주에서 발생한 허리케인인 Elena의 자료를 사용하였다. 궤도의 예측은 1985년도 8월 31일 0시를 기준으로 하여 이루어 졌다. 8월 31일 0시에 관측된 바람의 속력 값을 측정하고, 식 (4.1)을 사용하여 vorticity 값을 계산한다.

$$\zeta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{4.1}$$

여기서 x와 y는 동서 방향과 남북 방향의 축을 나타내며, u와 v는 관측된 바람의 수평 방향과 수직 방향의 속력을 나타낸다. 식(3.2)을 이용하여 streamfunction을 구했다. 이렇게 분석된 초기 상태인 streamfunction 값이 (그림 4.3)에 주어져 있다. (그림 4.3)에서 Florida 반도의 서쪽 해안가에 위치한 집중된 등고선의 중심점이 허리케인의 위치를 나타내며, 정확하게는 북위 28. 8도와 서경 84.4도이다. 계산에 사용되는 구간은, 경도로는 서경 40도에서 110도까지이고, 위도로는 북위 10도에서, 50도까지인 직사각형 구간이다. 격자점 사이의 간격은 1도씩으로 약 100km이다. 계산의 편의상 격자선의 간격을 등간격으로 하기 위하여 71×41개로 주어진 바람 자료를 IMSL 루틴을 사용하여 보간하여 63×42개로 보간하였다. 따라서, 계산에 사용된 격자점의 갯수는 M이 63이고, N이 42이다.



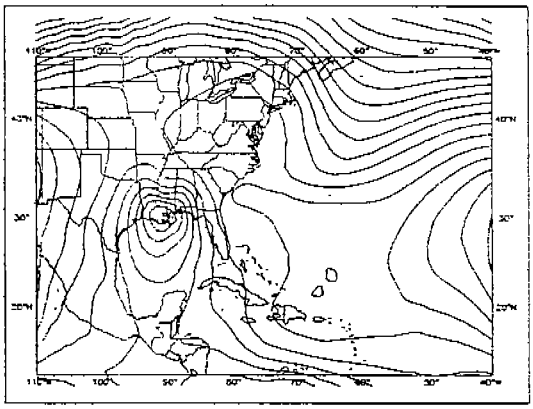
(그림 4.3) 바로트로픽 모델의 초기 조건 streamfunction, Aug. 31, 1985 0 UTC
(Fig. 4.3) The initial streamfunction field for barotropic vorticity model on Aug. 31, 1985 0 UTC

이 실험은 8개의 프로세서들을 갖고 있는 ALLIANT FX/8에서 실험되었고, FX/FORTRAN으로 작성된 프로그램은 Alliant의 벡터화와 병렬화 기능을 잘 이용할 수 있도록 최적화 되어 있다. 〈표 4.3〉에서 알 수 있듯이 실제 허리케인인 Elena를 72시간 예측하는 데 약 30초 정도에 가능하다. 만약 우리가 벡터 기능이 없는 한개의 프로세서를 사용하는 경우라면, 216초 정도가 걸린다. 이것은 병렬 벡터 프로세서를 사용하므로써 얻을 수 있는 이득으로 말할 수 있다. 〈표 4.3〉의 비율 1이란 scalar 프로세서를 사용하여 실험한 시간을 기준으로 계산효율을 계산한 것이며, 비율 2란 1개의 벡터 프로세서를 사용하여 실험한 시간을 기준으로 계산 효율을 계산한 것이다. 8개의 프로세서를

〈표 4.3〉 72시간 예측의 계산 비율 (Δ_t = 16분)
〈Table 4.3〉 The speed-up ratio after 72 hours forecast (Δ_t = 16 min)

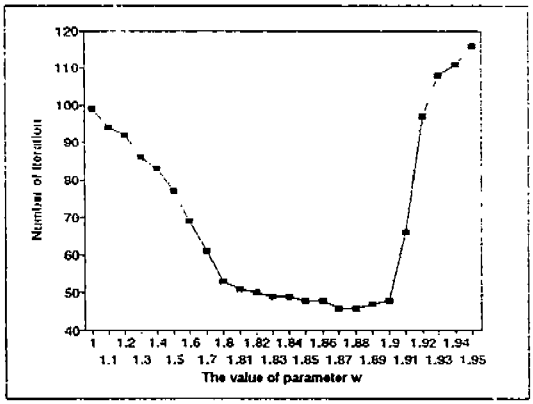
프로세서의 갯수	경과 시간(초)	비율 1	비율 2
Scalar	216.18	1.0	0.51
1 Vector	109.47	1.97	1.0
2 Vector	61.03	3.54	1.79
4 Vector	38.82	5.57	2.82
8 Vector	29.53	7.32	3.71

사용하는 경우, 벡터 기능이 없는 일반 프로세서를 사용한 경우와 벡터 기능을 사용한 8개의 프로세서들을 사용한 경우는 7배 이상의 계산 효율(Speed-up)을 얻었으며, 1개의 벡터 프로세서를 사용하는 경우에 비해서는 3.7 배 정도의 계산 효율을 얻을 수 있다. 또한 이러한 계산 효율은 시간 간격인 Δt를 다르게 하여도 비슷한 결과를 얻는다.



(그림 4.4) 72시간 후의 예측 궤도
(Fig. 4.4) The forecast path after 72 hours forecast

한편, 실제로는 실험에 사용되지는 않았지만, 반복법인 SOR 방법을 구현하여, BCR 방법과 비교하였다. 이 때, SOR 방법에서의 매개 변수인 ω의 최적치를 실험적으로 구하였다. (그림 4.5)에서 알 수 있듯이, M이 63이고, N이 42인 경우에 ω의 최적치는 1.87이다. 일반적으로 반복법은 넓은 범위의 문제들에 적



(그림 4.5) SOR 사용시 ω의 최적치
(Fig. 4.5) The optimum value of ω of SOR algorithm

용될 수 있는 장점이 있는 반면, 직접법은 반복법보다 빠르게 해를 구할 수 있다는 장점이 있다. 예를 들면, N×N의 격자점위에서 Poisson 방정식을 풀기 위한 시간 복잡도는 SOR 방법은 $O(N^3 \log N)$이다. 이에 비해 BCR 방법은 $O(N^2 \log N)$이다. 따라서 본 연구에서는 SOR 방법을 고속화하기보다는 BCR 방법을 고속화하였다.

5. 결론 및 앞으로의 연구 방향

본 연구에서는 바로트로픽 예측 모델에서 풀어야 하는 포아송 방정식의 해법인 BCR 방법을 실제 병렬 컴퓨터를 통하여 실현하였다. 격자점의 갯수에 상관없이 Dirichlet 경계 조건이나 Neumann 경계 조건을 갖는 다양한 형태의 포아송 방정식의 일반적인 해법들을 병렬화하였고, 이를 실제 사용되는 병렬 컴퓨터인 ALLIANT FX/8라는 미니급 수퍼 컴퓨터에 FX/FORTRAN으로 구현하였다. 또한, 실제로 발생하는 태풍이나 허리케인의 궤도를 추적하는데 사용되는 바로트로픽 예측 모델을 수치적으로 안정화시키며, 동시에 신속히 계산할 수 있는 고속화에 대하여 연구하였다.

바로트로픽 예측 모델을 장시간 적분하려면, 매 시간 간격당 포아송 방정식을 풀어야 하기 때문에 많은 양의 포아송 방정식을 풀어야 한다. 따라서, 포아송 방정식을 빠르게 푸는 것은 바로트로픽 예측 모델의 고속화에 절대적으로 필요하다. 물론 일정 시간 동안의 적분에서 시간 간격을 크게 하면, 풀어야 하는 포아송 방정식의 숫자도 줄어들게 된다. 그러나, 수치적 안정성 때문에 시간 간격을 임의적으로 크게 할 수 없는 제약이 있으므로, 예측 모델을 병렬화 하고, 특별히 포아송 방정식의 병렬화를 통하여, 전체 적분 시간을 줄이는 것은 매우 의미 있는 일이다. 그리고, 태풍의 궤도 예측과 같은 실제적인 문제에서는 대부분 공간 간격이 고정되어 실험자가 자유로이 크기를 조절할 수 없는 경우가 대부분이다. 격자점의 갯수를 임의로 조절할 수 없는 상황에서는 임의의 갯수에 대하여 포아송 방정식을 풀 수 있어야 한다.

바로트로픽 모델에 나타나는 비선형항인 Jacobian 항에 대한 이산화 방법을 기존의 방법이 장시간 예측을 수행할 때, 수치적으로 불안정함을 보였고, 이에

대한 개선책으로 Arakawa에 의해서 제안된 방법이 소개되었는데, 이 Arakawa의 Jacobian 항을 새로운 방식으로 유도해 내었다.

바로트로픽 예측 모델의 정확성을 보장하는 것은 초기 초기 조건의 정확성이다. 그러나 태풍이 발생하는 지점이 보통 해양이므로 관측 자료가 부족하다. 그러나 태풍이 발생하여 얼마간의 시간이 경과한 후에는 진행되어온 과거의 경로와 관측된 결과 및 과거에 예측된 결과들을 활용할 수 있다. 이러한 자료들을 더욱 정확한 초기 조건을 구하는 데 사용하여 앞으로의 새로운 예측의 정확성을 향상시키는 데에 사용 가능할 것이다. 이러한 과거의 자료를 통하여 더욱 좋은 초기 조건을 구하는 과정 중에 하나가 자료 동화 방법이다. 이 방법은 예측 모델의 적분에 걸리는 시간 보다 30-50배 이상의 시간을 요구하는 대규모 계산이다. 이러한 계산 과정에 병렬 처리 기법이 활용되는 것은 꼭 필요한 일이며, 본 연구는 이러한 연구의 전 단계 작업으로 매우 필요한 것이다.

참 고 문 헌

[1] D. Anderson, J. Tannehill, and R. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation, 1984.

[2] A. Arakawa, "Computational Design for long-Term Numerical Integration of the Equations of Fluid Motion: Two-Dimensional Incompressible Flow. Part I," Journal of Computational Physics, Vol. 1, pp. 119-143, 1966.

[3] B. L. Buzbee, G. H. Golub and C. W. Nielson, "On direct methods for solving Poisson's equation," SIAM Journal on Numerical Analysis, Vol. 7, pp. 627-655, 1970.

[4] E. Gallopoulos and Y. Saad, "A parallel block cyclic reduction algorithm for the fast solution of elliptic equations," Parallel Computing, Vol. 10, North-Holland, pp. 143-159, 1989.

[5] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 2nd Ed. Johns Hopkins University Press, 1989.

[6] G.J. Haltiner and R. T. Williams, *Numerical Prediction and Dynamic Meteorology*, 2nd ed., John Wiley & Sons, 1980.

[7] S. Lakshmivarahan and S. K. Dhall, *Analysis and Design of Parallel Algorithm*, McGraw Hill, New York, 1990.

[8] J. M. Lewis, and J. C. Derber, "The use of adjoint equations to solve a variational adjustment problem with advective constraints," Tellus 37A, 309-322, 1985.

[9] J. Lambiotte Jr. and R. Voigt, "The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer," ACM Trans. on Math. Software, Vol. 1, No. 4, pp. 308-329, 1975.

[10] M. DeMaria, "Tropical Cyclon Track Prediction with a Barotropic Spectral Model," Monthly Weather Review, Vol. 115, No. 10, pp. 2346-2357, 1987.

[11] J. M. Ortega, *Inroduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.

[12] F. Sanders, R. W. Burpee, "Experiments in Barotropic Hurricane Track Forcasting," Journal of Applied Meteorology, Vol. 7., No. 3, pp. 313-323, 1968.

[13] C. G. Song, J.S. Jwo, S. Lakshmivarahan, S.K. Dhall, J. Lewis, C.S. Velden, "An Experiment in hurricane track prediction using parallel computing methods," Parallel Algorithms and Applications, Vol. 2, pp. 315-332, 1994.

[14] P. A. Swarztrauber, "The method of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle," SIAM Review, Vol. 19, No. 3, pp. 490-501 July, 1977.

[15] R. A. Sweet, "A cyclic reduction algorithm for solving block tridiagonal systems of arbitrary dimension," SIAM Journal on Nuemrical Analysis, Vol. 143, pp. 706-720, 1977.

[16] R. A. Sweet, "A Parallel apnd vector variant of the cyclic reduction algorithm" SIAM Journal on Scientific and Statistical Computing, Vol. 9, pp.

761-765, 1988.

[17] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

[18] 송 창근, "노이만 경계 조건을 갖는 포아송 방정식의 개선된 직접 병렬 해법," 한국정보과학회논문지, 제 21권 제 4호, pp. 706-714.

[19] 이 동규, 권영철, 위 태권, "한반도에 접근하는 태풍, 1960-1989 제 2 부:중규모 모델에서의 예측실험," 한국기상학회지, Vol. 28, No. 2, pp. 149-163, 1992.

[20] FX/FORTRAN, *Language Manual Volume 1:Guidelines*, Alliant Computer System Corporation, 1986.

[21] FX/FORTRAN, *Programmer's Handbook*, Alliant Computer System Corporation, 1986.

[22] PV/WAVE Advantage, PV-WAVE Command Language, User's Guide, 1993.

송 창 근

1981년 서울대학교 계산통계학과, 이학사.

1983년 한국과학기술원 대학원 전산학과(공학석사).

1992년 University of Oklahoma (전산학 Ph.D.)

1984년~1987년 한림대학교 전자계산학과, 전임강사.

1992년~현재 한림대학교 컴퓨터공학과, 조교수.

관심분야:과학계산, 수치해석, 병렬알고리즘.

이 상 덕

1994년 한림대학교 전자계산학과 졸업(이학사)

1994년~1996년 한림대학교 대학원 컴퓨터공학과(공학석사)

1996년~현재 한림대학교 대학원 박사과정 재학중

관심분야:과학계산.