◎ 논 문

비정렬 격자계에서 강건하고 효율적인 LU-SGS 기법 개발: Part II- 효율적인 적용

김 주 성*1, 권 오 준*2

Robust and Efficient LU-SGS Scheme on Unstructured Meshes: Part II- Efficient Implementation

Joo Sung Kim and Oh Joon Kwon

In the present study, an efficient implementation technique of the van Leer's implicit operator is suggested in accordance with the Roe's explicit operator. By using an efficient treatment of the off-diagonal terms, which occupy most of the memory requirement for the linear system of equations, it is shown that the improved scheme only requires less than 30% of memory and is approximately 10-20% faster than the baseline scheme.

Key Words: LU-SGS 기법, 내재적 연산자(Implicit Operator), van Leer's FVS 기법, 비정렬 격자(Unstructured Mesh)

1. 서 론

최근 들어 대형의 병렬 컴퓨터와 전산유체역학(CFD) 해석 기법의 발전과 함께 복잡한 물리적인 현상이나 삼차원의 복잡한 형상에 대한 공학적인 해석이 가능하게 되었다. 또한 복잡한 형상에 대한 해석 욕구와 이러한 형상에 대한 정렬 격자 생성의 어려움으로 인하여 비정렬 격자 기법에 관한 연구가 최근까지 매우 활발하게 진행되고 있다.

본 논문의 part I [1]에서는 강건하고 효율적인 해석 기법을 위해서는 내재적 연산자의 수치적인 소산 레벨이 외재적 연산자의 소산 레벨보다 커야 함을 보였다. 결과적으로 비점성 유동 뿐만 아니라 점성 유동에 대해서도 지속적으로 매우 정확한 결과를 제공하는 Roe의 FDS(flux difference splitting) 기법을 외재적 연산자로 사용하고자 할 때, Roe 기법에 기초한 내재적 연산자보다는 이 보다 수치적인 소산이 큰 기법을 내재적 연산자로 선택하는게 좋다. Part I 에서는 van Leer의 FVS(flux vector splitting) 기법에 기초한 내재적 연산자가 가장 적절한 것으로 보이고 있다. 본 연구에서는 이러한 van Leer의 FVS 내재적 연산자에 대해서 효율적으로 해석 코드에 적용하는 연구를 수행하고자 한다.

삼차원의 실제적인 형상을 해석하기 위해서는 일반적으로 수십만에서 수백만개의 격자가 필요하며, 격자 적응(mesh adaptation) 등을 사용하여 정확한 해석을 수행하기 위해서는 이보다 훨씬 더 많은 격자가 필요하다. 이러한 대형의 문제를 해석하기 위해서 부딪히는 가장 큰 장애 요인중에 하나는 계산 시간 증대와 함께 컴퓨터의 기억 용량의 제한이다. 따라서 매우 빠르고 강건한 기법이라 하여도 기억 용량 요구

* 2004년 5월 5일 접수
*1 학생회원, 한국과학기술원 항공우주공학과 대학원
*2 정회원, 한국과학기술원 항공우주공학과 교수

량이 크다면 비정렬 격자 유동 해석 기법으로는 적합하지 않게 된다. 또한 기억 용량의 요구량을 감소시키기 위해 계산 시간이 매우 크게 증대된다면 이 또한 효율적이지 못하게 된다.

Nakahashi 등[2]은 Jameson과 Yoon[3]에 의해 정렬 격자계에서 적용된 간단화된 내재적 연산자를 사용하는 LU-SGS(lower-upper symmetric Gauss-Seidel) 기법을 성공적으로 비정렬 격자계에 확장 적용하였다. 이 연구에서는 원래의 LU-SGS 기법에 존재하였던 자코비안 행렬의 계산과 기억 용량에의 저장을 적절한 근사화를 통해서 완전히 없앴다. 또한 Chen과 Wang[4]은 Roe의 FDS 기법에 기초한 내재적 연산자를 사용하는 LU-SGS 기법에 대해서 비대각항을 내부반복계산 동안에 반복적으로 계산하는 방법을 사용하여 기억 용량 요구량의 상당 부분 줄였다. GMRES 기법에서도 유한 차분 방법을 이용하여 내재적 연산자를 구성하여 저장하지 않는 matrix-free 방법이 많이 사용되고 있다. 또한 Luo 등[5]은 이러한 matrix-free GMRES 방법에 Nakahashi[2] 등에 의해 개발된 LU-SGS 예조건자(preconditioner)를 사용하는 방법을 채택하여 전체 시간 적분 기법을 matrix-free 기법으로 구성하는 방법을 제안하였다.

본 연구에서는 내재적 연산자로 van Leer의 FVS 기법을 사용하는 LU-SGS 기법에 대해서 기억 용량의 요구량이 매우 적으면서, 내재적 연산자를 저장하여 사용하는 기법에 비해 계산 시간의 증가가 없는 방법을 제안하고자 한다. 제안된 기법에 대해서 기존의 기법들과 수렴성, 계산시간, 기억 용량 요구량을 비교하여 효율성을 검증하고자 한다. 특히 비정렬 격자에서 효율적으로 사용되고 있는 Nakahashi 등에 의해 개발된 LU-SGS 기법과의 수렴성 및 계산 시간과 기억 용량 등을 비교하고자 한다. 이러한 검증을 위해 자유류 유동, NACA 0012, RAE 2822, Suddhoo-Hall 4요소 익형에 대한 해석을 수행하였다.

2. Gauss-Seidel Relaxation 방법

본 연구에서는 내재항은 일차정확도를 사용하며 외재항에는 이차정확도를 사용하는 defect correction 방법에 대한 선형 방정식인 part I 의 식 (4)를 계산하기 위해 Gauss-Seidel relaxation 방법을 사용한다. Part I 의 식 (4)는 다음과 같다.

$$\left[\frac{V_i}{\Delta t_i} I + J_{low}(Q^n)\right]\Delta Q = -R_{high}(Q^n) \tag{1}$$

위 식의 선형 방정식을 대각항(diagonal element)과 비대각항(off-diagonal element)의 두 부분으로 나누어서 쓰면 다음과 같다.

$$D_i \Delta Q_i + \sum_{j \in nf(i)} O_{ij} \Delta Q_j = -R_i(Q) \tag{2}$$

여기서 $nf(i)$는 i 격자점에 붙어있는 총 격자면의 수를 나타내며, 인덱스 i와 j는 변수가 저장되는 격자점들을 표현한다. 잔류항 $R_i(Q)$ 는 Roe 기법을 사용하여 계산된다. 대각항과 비대각항을 각각 표현하는 D_i와 O_{ij} 는 다음과 같다.

$$D_i = \left\{\frac{V_i}{\Delta t_i} I + \sum_{j \in nf(i)} A_{ij}^{+} \Delta l_j\right\}, \quad O_{ij} = A_{ij}^{-} \Delta l_j \tag{3}$$

여기서 A_{ij}^{+}와 A_{ij}^{-}는 다음과 같다.

$$A_{ij}^{+} = \frac{\partial F_{ij}}{\partial Q_i}, \quad A_{ij}^{-} = \frac{\partial F_{ij}}{\partial Q_j} \tag{4}$$

여기서 F_{ij}는 저차 정확도로 구해지는 i 격자점에 대한 제어체적의 j 제어표면에서의 수치적인 비점성 플럭스를 나타내며, 저차정확도로 구해지기 때문에 Q_i와 Q_j 만의 함수로 표현된다. 또한 F_{ij}에는 잔류항을 구하는 방법과 다른 기법이 사용될 수 있다.

우선 잔류항과 같은 기법의 Roe의 FDS 기법에 대한 내재적 연산자는 다음과 같이 표현될 수 있다.

$$A_{ij}^{+} = \frac{1}{2}[A_{ij}(Q_i) + |\bar{A}|], \; A_{ij}^{-} = \frac{1}{2}[A_{ij}(Q_j) - |\bar{A}|] \tag{5}$$

여기서 간단화를 위해 $|\bar{A}|$ 행렬이 시간 변화에 대해서 일정하다는 가정이 사용되었다. FVS 기법은 Roe의 기법과 같은 FDS 기법과는 달리 플럭스 벡터의 인자분해가 주위의 값와 연관되지 않고 수행되는 point-by-point 인자분해 방법을 사용한다. 이러한 플럭스 식에 대해서 내재적 연산자는 다음과 같이 표현된다.

$$A_{ij}^{+}=\frac{\partial F_{ij}^{+}(Q_i)}{\partial Q_i}\quad,\quad A_{ij}^{-}=\frac{\partial F_{ij}^{-}(Q_j)}{\partial Q_j} \tag{6}$$

위의 미분들은 단순한 수식 전개를 통해서 어떠한 근사 없이도 쉽게 얻을 수 있다.

Gauss-Seidel 기법과 같은 relaxation 방법에서는 선형방정식의 대각우세 성질이 만족되어야 수렴하게 된다. 중심 차분 기법(central differencing scheme)에 대해서 내재적 방법을 사용하여 차분화하면 결과적인 선형 방정식은 매우 작은 시간 간격에 대해서만 대각 우세 성질을 유지할 수 있으며, 따라서 매우 낮은 수렴성을 보인다. Jameson과 Turkel[6]은 이러한 중심 차분법에 대해서 매우 큰 시간간격에 대해서도 항상 대각 우세(diagonal dominance) 성질을 유지할 수 있는 내재적 연산자를 개발하였다. 이는 실제로 내재항인공 소산항(implicit artificial dissipation)을 사용하는 방법과 같다:

$$\begin{aligned}A_{ij}^{+}&=\frac{1}{2}[A_{ij}(Q_i)+r_{A,ij}I],\\ A_{ij}^{-}&=\frac{1}{2}[A_{ij}(Q_j)-r_{A,ij}I]\end{aligned} \tag{7}$$

여기서 r_A에 대해서는 $r_A \geq \max(|\lambda_A|)$의 조건을 만족해야 한다. 여기서 $|\lambda_A|$는 플럭스 자코비안 A의 고유값이다. 실제 적용에서는 상수 k를 도입하여 $r_A = k\max(|\lambda_A|)$가 사용되며, k는 1보다 크거나 같아야 한다. 또한 k의 전형적인 값은 1이며, 본 연구에서는 이 값만을 사용한다.

내재적 연산자는 수치적인 플럭스를 미분하여 얻으므로, 식 (7)의 Jameson과 Turkel이 제안한 내재적 연산자에 해당하는 플럭스를 역으로 구할 수 있으며, 다음과 같다[5].

$$F_{ij}=\frac{1}{2}[f_{ij}(Q_i)+f_{ij}(Q_j)-\max(|\lambda_{ij}|)(Q_j-Q_i)] \tag{8}$$

여기서 $\max(|\lambda_A|)$는 시간 변화에 대해서 일정하다고 가정되었으며, 이러한 가정은 Roe의 FDS 기법에 대한 근사적인 내재적 연산자 계산에서와 같다. 식 (8)은 내재적 연산자가 자동적으로 대각우세 성질을 갖는 풍상차분 기법과 같은 형태로 표현되었으며, Roe의 FDS 기법에서 Roe-average 행렬 $|\bar{A}|$가 $\max(|\lambda_A|)$로 교체된 형태와 같다. 이러한 수치적인 비점성 플럭스는 많이 쓰이지는 않지만 HLL 기법의 특별한 형태로 표현될 수 있는 Rusanov의 플럭스 식을 나타낸다[5]. 따라서 Jameson과 Turkel이 제안한 대각 우세를 유지할 수 있는 내재적 연산자는 Rusanov의 플럭스를 미분하여 얻을 수 있는 내재적 연산자로 생각할 수 있다.

3. 기본 LU-SGS 기법

식 (2)를 전방 스윕(forward sweep)과 후방 스윕(backward sweep)을 번갈아 수행하는 LU-SGS 방법으로 계산한다면 다음과 같이 표현된다.

전방 스윕:

$$\begin{aligned}D_i\Delta Q_i^{(*)} &= -R_i(Q)\\ &- \sum_{j\in low(i)} O_{ij}\Delta Q_j^{(*)} - \sum_{j\in up(i)} O_{ij}\Delta Q_j^{(k)}\end{aligned}$$

후방 스윕:

$$\begin{aligned}D_i\Delta Q_i^{(k+1)} &= -R_i(Q)\\ &- \sum_{j\in low(i)} O_{ij}\Delta Q_j^{(*)} - \sum_{j\in up(i)} O_{ij}\Delta Q_j^{(k+1)}\end{aligned} \tag{9}$$

여기서 위첨자 k는 내부반복계산의 레벨(level)을 나타내며, 위첨자 $*$는 전방 스윕에서 새로 계산되는 레벨을 나타낸다. 또한 $low(i)$와 $up(i)$는 i 격자점에 해당하는 비대각 행렬중에서 아랫 부분과 윗 부분을 각각 나타낸다.

식 (9)에서 Roe의 FDS와 van Leer의 FVS 기법에 대한 내재적 연산자뿐만 아니라 Jameson과 Turkel의 내재적 연산자에 대해서도 선형방정식은 행렬 요소

를 갖는다. 따라서 대각 행렬에 대해서는 $nnode \times neqn \times neqn$의 기억용량이 필요하며, 비대각 행렬에 대해서는 $2 \times nedge \times neqn \times neqn$의 기억용량이 필요하다. 여기서 $nnode$와 $nedge$는 각각 격자점과 격자면의 개수이다. 또한 $neqn$은 지배방정식의 수이며, 이차원에 대해서는 4이다. 이차원 삼각형 격자에 대해서 $nedge \approx 3\,nnode$이므로, 전체 내재적 연산자를 저장하는 기억용량에서 비대각항을 저장하는 부분이 전체의 약 86%를 차지하게 된다. 또한 삼차원에 대해서는 $nedge \approx 7\,nnode$ 이므로 비대각항을 저장하는 부분이 전체의 약 93%를 차지하게 된다. 전체적인 선형 방정식을 저장하기 위해서는 0이 아닌 요소만을 저장하는 Compressed Sparse Row(CSR) 저장 방법을 사용하였다[8].

4. 개선된 LU-SGS 기법

Jameson과 Turkel의 내재적 연산자에 대해서 대각행렬의 부분은 다음과 같이 스칼라 값으로 단순화되어 표현될 수 있다.

$$D_i \Delta Q_i = \left(\frac{V_i}{\Delta t_i} + \sum_{j \in nb(i)} \max(|\lambda_A|) \Delta l_j \right) \Delta Q_i \qquad (10)$$

또한 비대각 항의 부분에 대해서는 다음과 같이 근사적으로 표현될 수 있다.

$$O_{ij} \Delta Q_j = \frac{1}{2} [A_{ij}(Q_j) \Delta Q_j - \max(|\lambda_A|) \Delta Q_j] \Delta l_j$$
$$\approx \frac{1}{2} [f_{ij}(Q_j + \Delta Q_j) - f_{ij}(Q_j) - \max(|\lambda_A|) \Delta Q_j] \Delta l_j \qquad (11)$$

여기서 플럭스 자코비안 행렬과 해의 증분 벡터의 곱은 플럭스의 증분형태로 근사화되었다. 식 (10)과 (11)에서 자코비안 행렬에 해당하는 부분은 모두 사라지고 없으며, 이로 인해 LU-SGS 방법을 적용하는데 필요한 $neqn \times neqn$의 행렬 형태의 요소를 갖는 선형방정식을 계산하여 저장할 필요가 없게되며, 대각항에 해당하는 부분에 대해서 스칼라 값만을 저장하면 된다. 하지만 내부반복 계산동안에 간단한 형식을 갖는 f에 대한 계산이 반복적으로 필요하게 된다. 이 방법은 기본 기법(baseline scheme)과 비교하여 부가적인 계산시간의 소요가 거의 없이 기억 용량을 현저하게 줄일 수 있는 방법인 것으로 보고되었다[2, 5].

Jameson과 Turkel이 제안한 내재적 연산자에 대해서 전방 스윕과 후방 스윕을 여러번 수행하는 LU-SGS 기법으로 표현하면 다음과 같다.

전방 스윕:

$$D_i \Delta Q_i^{(*)} = -R_i(Q) + \frac{1}{2} \sum_{j \in nl(i)} f_{ij}(Q_j) \Delta l_j$$
$$-\frac{1}{2} \sum_{j \in low(i)} [f_{ij}(Q_j + \Delta Q_j^{(*)}) - \max(|\lambda_A|) \Delta Q_j^{(*)}] \Delta l_j$$
$$-\frac{1}{2} \sum_{j \in up(i)} [f_{ij}(Q_j + \Delta Q_j^{(k)}) - \max(|\lambda_A|) \Delta Q_j^{(k)}] \Delta l_j$$

후방 스윕:

$$D_i \Delta Q_i^{(k+1)} = -R_i(Q) + \frac{1}{2} \sum_{j \in nl(i)} f_{ij}(Q_j) \Delta l_j$$
$$-\frac{1}{2} \sum_{j \in low(i)} [f_{ij}(Q_j + \Delta Q_j^{(*)}) - \max(|\lambda_A|) \Delta Q_j^{(*)}] \Delta l_j$$
$$-\frac{1}{2} \sum_{j \in up(i)} [f_{ij}(Q_j + \Delta Q_j^{(k+1)}) - \max(|\lambda_A|) \Delta Q_j^{(k+1)}] \Delta l_j \qquad (12)$$

계산의 효율성을 높이기 위해서 오른편 항의 첫 번째 $\sum$은 내부반복계산이 시작되기 전에 잔류항에 더해진다. 또한 전방 스윕 단계에서는 아랫 부분에 해당하는 비대각항만 계산하면 되며, 윗 부분에 해당되는 부분은 후방 스윕 단계에서 계산된 값을 저장하여 사용한다. 후방 스윕에서는 반대이다. 이러한 방법은 LU-SGS 기법의 특성상 수렴성에 영향을 미치지 않는다. 또한 이러한 저장은 단지 $nnode \times neqn$의 저장 용량만을 필요로 하기 때문에 전체적으로 볼 때 매우 미미한 정도이다.

내재적 연산자의 저장에 필요한 기억 용량의 대부분은 비대각항의 저장을 위해 요구된다. 참고문헌 [4]에서는 Roe의 FDS 기법에 기초한 내재적 연산자에 대해서 대각항은 미리 계산하여 저장하여 사용하고, 비대각 항은 기억 용량 요구량을 줄이기 위해 내부반복계산 동안에 필요한 단계에서 계산하여 사용하는 방법을 제안하였다. 비대각항은 다음과 같이 근사화될 수 있다.

$$O_{ij} \Delta Q_j \approx \frac{1}{2} [F_{ij}(Q_i, Q_j + \Delta Q_j) - F_{ij}(Q_i, Q_j)] \qquad (13)$$
$$\approx \frac{1}{2} [f_{ij}(Q_j + \Delta Q_j) + f_{ij}(Q_j) - |\bar{A}| \Delta Q_j]$$

식 (13)을 사용하여 전방 스윕과 후방 스윕을 여러번 수행하는 LU-SGS 기법으로 표현하면, Jameson과 Turkel의 내재적 연산자를 사용하는 식 (12)와 비교하여 $\max(|\lambda_A|)$가 $|\overline{A}|$로 대치된 형태로 표현된다. $|\overline{A}|$ 행렬은 $\max(|\lambda_A|)$보다 훨씬 복잡하므로 내부반복계산 동안에 계속해서 구하는 것은 Jameson과 Turkel의 기법보다 계산시간이 훨씬 많이 소요되게 된다. 또한 이 방법은 Roe 기법의 내재적 연산자를 사용하는 기본 기법보다 한번의 전방 스윕과 후방 스윕의 조합당 저차정확도 Roe 기법에 대한 플럭스 계산 시간만큼 근사적으로 더 필요하게 된다.

FVS 기법의 내재적 연산자에 대해서 비대각 항은 다음과 같이 쓰여질 수 있다.

$$\begin{aligned} & O_{ij}\Delta Q_j \\ \approx & \left[F_{ij}^{+}(Q_i)+F_{ij}^{-}(Q_j+\Delta Q_j)\right]-\left[F_{ij}^{+}(Q_i)+F_{ij}^{-}(Q_j)\right] \\ = & F_{ij}^{-}(Q_j+\Delta Q_j)-F_{ij}^{-}(Q_j) \end{aligned} \tag{14}$$

여기서 전방 플럭스인 F^{+}는 FVS 기법의 point-by-point 인자분해 특성에 의해 소거되었다. 식 (14)를 사용하여 전방 스윕에 대한 LU-SGS 기법을 표현하면 다음과 같다.

전방 스윕:

$$\begin{aligned} D_i\Delta Q_i^{(*)} = & -R_i(Q)+\sum_{j\in nb(i)} F_{ij}^{-}(Q_j)\Delta l_j \\ & -\sum_{j\in low(i)} F_{ij}^{-}(Q_j+\Delta Q_j^{(*)})\Delta l_j \\ & -\sum_{j\in up(i)} F_{ij}^{-}(Q_j+\Delta Q_j^{(k)})\Delta l_j \end{aligned} \tag{15}$$

위 기법은 van Leer 기법의 내재적 연산자를 사용하는 기본 기법보다 한번의 전방 스윕과 후방 스윕의 조합당 저차정확도 잔류항의 F^{-} 부분만을 구하는 계산량 만큼 근사적으로 더 필요하게 된다. 따라서 내부반복계산을 많이 수행할수록 기본 기법보다 계산량이 많아지게 된다. 하지만 van Leer나 Steger-Warming의 FVS 기법은 Roe나 Osher의 FDS 기법과 비교하여 상대적으로 플럭스 구하는 방법이 단순하다(하지만 플럭스 쟈코비안의 계산은 거의 비슷하다). 또한 비대각항이 FVS 기법의 특성에 의해 단순화 되었기 때문에 기본 기법과 비교하여 계산시간 증대가 적정 수준이 될 것으로 판단된다.

개선된 LU-SGS 기법에서는 기본 기법에서와 달리 선형방정식을 저장하기 위해 CSR 저장 방법과 같은 특별한 방법이 필요치 않으며, 단지 해당 격자점에 대해서 $neqn \times neqn$의 크기의 행렬만을 저장하고 있으면 된다. 이러한 특성에 의해 개선된 LU-SGS 기법은 기억 용량의 요구량을 줄이는 장점외에 캐쉬 기억 용량(cache memory)을 효율적으로 사용하는 장점을 갖고 있다. 예로 격자면 자료 구조를 사용하여 격자점에 저장되는 값을 계산할 경우 캐쉬 손실(cache miss)을 최소화 하기 위해서는, 격자점에 의해 요구되는 자료의 위치와 격자면에 의해 요구되는 자료 위치가 되도록이면 가까워야 된다[9]. 이러한 점에 비추어 기본 기법에서 CSR 저장방법을 사용하여 저장된 선형 방정식의 자료 위치와 격자점이나 격자면의 자료 위치와는 많은 차이가 나므로, 캐쉬 손실이 많이 발생된다. 하지만 개선된 기법에서는 단순히 대각항만 격자점의 위치에 저장되기 때문에 상대적으로 캐쉬 손실이 적게 발생된다.

5. 계산 결과

여러 가지 검증 경우에 대해서 Jameson과 Turkel의 내재적 연산자와 van Leer FVS 내재적 연산자를 사용하는 기본 LU-SGS 기법과 개선된 LU-SGS 기법의 수렴된 해를 위한 반복 계산 회수 및 계산 시간에 관해서 비교하고자 한다. 표현의 간단화를 위해 Jameson과 Turkel의 내재적 연산자는 JT로 van Leer의 내재적 연산자는 VL의 약자로 표현되었다. 모든 계산은 Linux OS를 사용하는 Pentium Ⅳ 2.4MHz CPU(central processing unit)와 256 Mbyte RAM을 사용하는 PC에서 수행되었으며, 변수들의 저장과 계산은 double precision으로 하였다.

5.1 자유류 유동

본 검증 경우는 길이가 20이며, 높이가 10인 단순한 직사각형 계산 영역에 대해서 마하수 0.8에 대한 자유류의 유동이며, 어떠한 교란도 존재하지 않는다. 본 검증 경우의 목적은 내부반복계산과 격자수와 같은 수치적인 파라메터에 대해서 기본 LU-SGS 기법

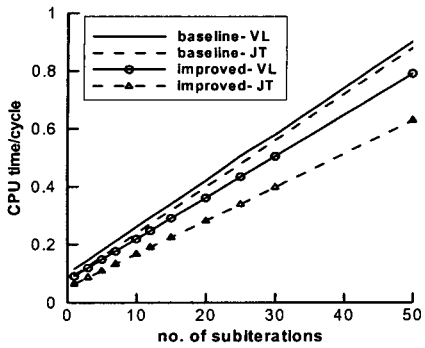

Fig. 1 Comparison of CPU time/cycle between baseline and improved LU-SGS schemes as a function of number of subiterations.

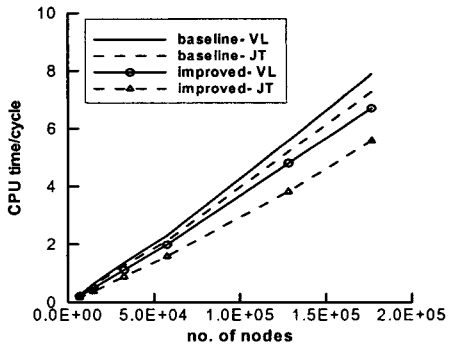

Fig. 2 Comparison of CPU time/cycle between baseline and improved LU-SGS schemes as a function of number of nodes.

과 개선된 LU-SGS 기법의 단순한 계산량의 비교를 하는데 있다. 계산에 사용된 격자는 Advancing Front Method(AFM) 기법을 사용하여 생성하였으며, 각각의 격자 요소는 등방성 삼각형 모양을 갖는다.

개선된 LU-SGS 기법은 내부반복계산 동안에 비대각항에 해당하는 내재적 연산자를 반복적으로 계산하므로, 내부반복계산의 수에 대한 기본 기법과 개선된 기법의 성능에 관한 비교는 매우 중요하다. Fig. 1은 기본 LU-SGS 기법과 개선된 기법의 내부반복계산의 수에 따른 반복계산당 계산시간을 보이고 있다. 계산에 사용된 격자는 6,830개의 격자점, 13,263개의 격자, 20,092개의 격자면으로 구성되었으며, 경계면에 395개의 격자점이 존재한다. JT 내재적 연산자를 사용하는 개선된 기법은 기본 기법에 비해 모든 내부반복계산에 대해서 약 30% 정도의 계산 시간이 적게 소요되는 것을 볼 수 있다. 또한 VL 내재적 연산자에 대해서는 개선된 기법이 약 15% 정도의 계산시간이 적게 소요되는 것을 볼 수 있다. 따라서 개선된 기법의 CPU floating point operation의 증가가 실제로 매우 미미하며, 캐쉬 기억용량의 효율성의 증대와 indirect addressing의 감소에 의한 호의적인 효과가 지배적인 것으로 판단된다. 이 경우의 격자에 대해서는 JT 내재적 연산자를 사용하는 개선된 기법은 기본 기법에 비해 선형방정식을 저장하는데 약 1%의 기억 용량을 사용하며, VL 내재적 연산자를 사용하는 개선된 기법은 약 14.5% 기억 용량을 사용한다.

개선된 기법의 호의적인 효과에 기여하는 캐쉬 기억용량의 효율적인 사용과 indirect addressing의 감소는 궁극적으로 격자수의 크기에 따라 달라질 수 있다. 또한 개선된 기법은 많은 격자수에 대한 계산을 수행하고자 개발된 기법이므로 격자수의 증감에 따른 계산 성능은 중요한 특성이 된다. Fig. 2는 격자점의 수에 따른 반복 계산당 계산시간을 보이고 있다. 격자점의 수는 6,830개에서 176,541개 까지 변화되었다. 격자수의 크기에 상관없이 JT 내재적 연산자를 사용하는 개선된 기법은 약 30%, VL 내재적 연산자를 사용하는 개선된 기법은 약 15% 정도의 계산시간이 적게 소요되는 것을 볼 수 있다. 따라서 격자수에 무관하게 개선된 기법은 기본 기법에 비해 계산시간이 적게 소요되는 것을 알 수 있다.

Fig. 2의 가장 큰 격자수에 대해서 기본 기법은 약 220Mbyte 정도의 기억용량을 요구하며, 이는 본 연구에서 사용하는 PC에서 계산할 수 있는 최대 격자 크기에 근접한다. 하지만 JT 내재적 연산자를 사용하는 개선된 기법은 약 60 Mbyte, VL 내재적 연산자를 사용하는 개선된 기법은 약 80 Mbyte 정도의 기억용량만을 필요로 한다. 따라서 개선된 기법은 기본 기법에 비해 약 30% 정도의 기억 용량만을 필요로 한다.

5.2 NACA 0012 익형

본 검증 경우는 자유류 마하수가 0.8이며, 받음각이 1.25도인 NACA 0012 익형 주위의 천음속 유동이다. 본 예제의 목적은 해석 프로그램의 검증에 많이 사용되는 표준적인 유동 조건에서의 개선된 LU-SGS 기

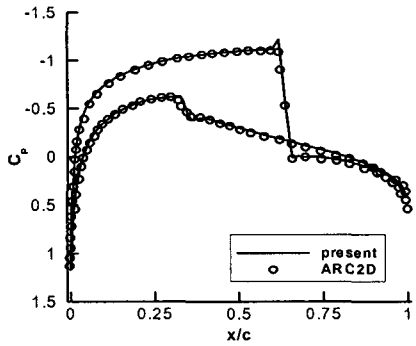

Fig. 3 Pressure coefficient distributions for a NACA 0012 airfoil at M_∞=0.8, α=1.25°.

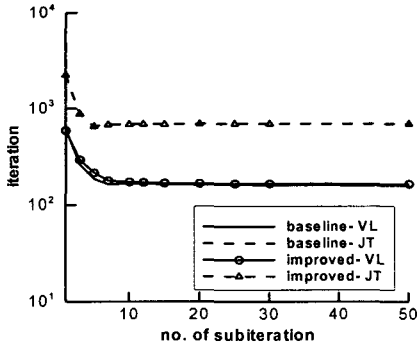

Fig. 4 Number of iterations required for the tenth order residual reduction as a function of subiteration.

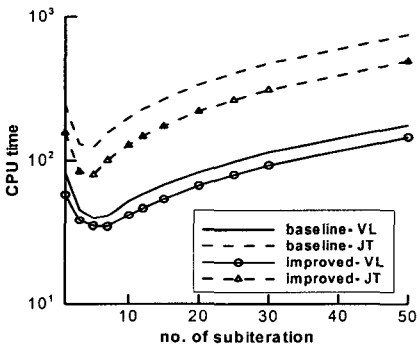

Fig. 5 CPU time required for the tenth order residual reduction as a function of subiteration.

법과 기본 기법의 성능을 평가하는데 있다. 계산에 사용된 격자는 7,355개의 격자점과 14,431개의 격자, 21,786개의 격자면으로 구성되었으며, 익형의 표면에는 279개의 격자점이 분포되어 있다. 이러한 격자수는 통상적인 계산에 사용되는 범위이다. Fig. 3에서는 익형 표면에서의 압력 분포를 중심 차분 기법을 사용하는 ARC2D 프로그램[10]의 결과와 비교하고 있다. 충격파의 위치와 강도 뿐만 아니라 익형의 모든 위치에서 ARC2D 결과와 잘 일치하는 것을 볼 수 있다. 하지만 Venkatakrishnan의 제한자의 사용으로 인해 윗면의 충격파 주위에서 어느 정도의 해의 진동이 발생된 것을 볼 수 있다. 이 경우 제한자를 사용하지 않으면 기계오차 까지 수렴된 해를 얻을 수 없다.

유동 조건과 격자가 정의되면 해석 프로그램의 수렴성은 CFL(Courant-Friedrichs-Lewy) 수와 내부반복계산의 수에 의해 달라지게 된다. JT 내재적 연산자는 매우 큰 CFL 수에서도 안정하며, 본 연구에서는 모든 경우에 대해서 매우 큰 값인 10^7의 값이 사용되었으며, 이러한 큰 값에서 항상 가장 좋은 수렴성을 보인다. VL 연산자는 JT 연산자 보다 CFL 수의 변화에 대해서 민감하며, 너무 큰 값에 대해서는 발산한다. 본 연구에서는 10^3의 값이 사용되었다. 이 값이 모든 조건에 대해서 최적은 아니지만, 대체적으로 좋은 결과를 얻을 수 있는 값이다.

Fig. 4는 내부반복계산수의 함수관계로 잔류치 10 order를 감소시키는데 필요한 반복 계산 회수를 보이고 있다. 같은 내재적 연산자를 사용하는 경우 개선된 기법과 기본 기법의 반복계산당 수렴 성은 같은 것을 볼 수 있다. 또한 JT 연산자와 VL 연산자 모두에 대해서 어느 정도 이상의 내부반복계산 이상에서는 수렴성의 증가없이 일정하게 된다. 가장 좋은 수렴성에 대해서 VL 연산자가 JT 연산자 보다 약 5배 정도의 빠른 반복계산당 수렴성을 보인다.

Fig. 5는 내부반복계산의 수에 따른 실제로 반복계산 회수 보다 더 중요한 CPU 계산 시간을 보이고 있다. 5.1 절의 자유류 유동의 경우에서 본 것처럼 JT 연산자에 대해서는 개선된 기법이 약 30%, VL 연산자에 대해서는 약 15% 정도의 계산 시간 절약이 있는 것을 볼 수 있다. 또한 JT 연산자에 대해서는 5번 정도의 내부반복계산에 대해서, VL 연산자에 대해서는 7번 정도의 내부반복계산에 대해서 가장 작은 계산 시간이 소요된다. VL 연산자를 사용하는 개선된 기법은 JT 연산자를 사용하는 경우보다 계산 시간에서 약 2배 정도 빠르다. 따라서 본 연구에서 제안된

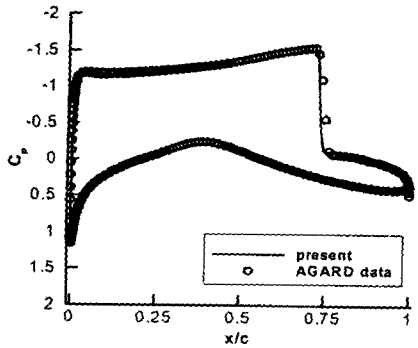

Fig. 6 Pressure coefficient distributions for the RAE 2822 airfoil at M_∞=0.75, α=3°.

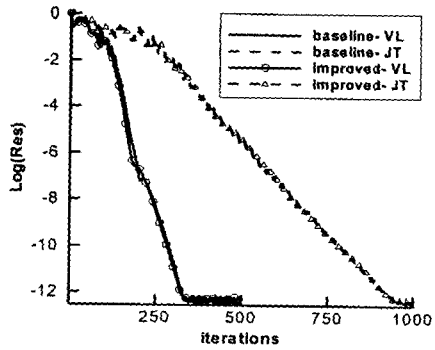

Fig. 7 Convergence rates vs iterations for the RAE 2822 airfoil flow.

VL 연산자의 개선된 기법은 현재 많은 연구자들에 의해 사용되는 JT 연산자의 개선된 기법에 비해 작은 기억 용량의 증가만을 요구하면서 두배 정도의 빠른 수렴성을 얻을 수 있는 기법으로 판단할 수 있다.

5.3 RAE 2822 익형

본 검증 경우는 참고문헌 [11]에서 제시된 비점성 유동 검증 경우 중에서 6번에 해당한다. 자유류 마하수는 0.75이며, 받음각은 3도인 RAE 2822 익형주위의 천음속 유동이다. 사용된 격자는 5,159개의 격자점과 10,050개의 격자, 15,209개의 격자면으로 구성되었으며, 익형의 표면에는 268개의 격자점이 분포되어 있다. Fig. 8에서는 익형 표면에서의 압력 분포를 중심 차분 기법을 사용하여 얻은 참고문헌 [11]의 결과와 비교하고 있으며, 서로 잘 일치하고 있는 것을 볼 수 있다. 본 검증 경우에도 Venkatakrishnan의 제한자가 사용되었으며, 제한자를 사용하지 않으면 기계오차 까지 수렴된 해를 얻을 수 없다.

Fig. 7과 8은 반복 계산과 CPU 시간에 대한 수렴성을 각각 보이고 있다. CFL 수와 내부반복계산의의 수는 실제 계산에서 중요한 CPU 시간이 가장 적게 소요되는 값으로 조절되었다. CFL 수는 JT 내재적 연산자에 대해서는 10^7의 값이 사용되었으며, VL 연산자에 대해서는 10^3의 값이 사용되었다. 내부반복계산의 수는 JT와 VL 연산자 모두에 대해서 7의 값이 사용되었다. Fig. 7에서 같은 내재적 연산자를 사용하는 경우 개선된 기법과 기본 기법의 반복 계산

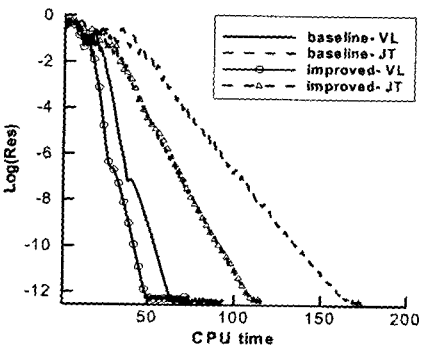

Fig. 8 Convergence rates vs CPU time for the RAE 2822 airfoil flow.

에 대한 수렴성은 같은 것을 볼 수 있다. 또한 VL 연산자가 JT 연산자 보다 약 3배 정도의 빠른 수렴성을 보이고 있다. Fig. 8에서 JT 연산자에 대해서는 개선된 기법이 기본 기법에 비해서 약 30%, VL 연산자에 대해서는 약 20% 정도의 계산 시간이 적게 소요되는 것을 볼 수 있다. 또한 VL 연산자를 사용하는 개선된 기법은 JT 연산자를 사용하는 경우보다 계산 시간에서 약 2배 정도 빠른 것을 볼 수 있다.

5.4 Suddhoo-Hall의 4요소 익형

마지막 검증 경우로 비압축성 포텐셜 유동에 대한 해석해가 있는 4개의 요소로 구성된 익형에 대한 해석을 수행하였으며, 이 경우 받음각은 0도 이다[12]. 해석해는 비압축성에 해당하지만, 현재의 압축성 유동 해석 프로그램의 수렴성 알아보기

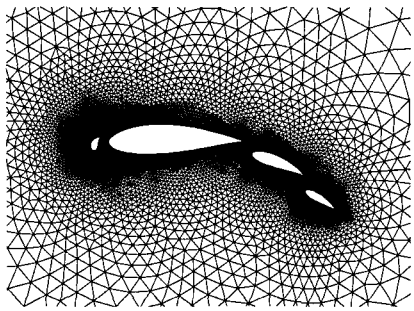

Fig. 9 Triangular mesh for four-element airfoil of Suddhoo and Hall.

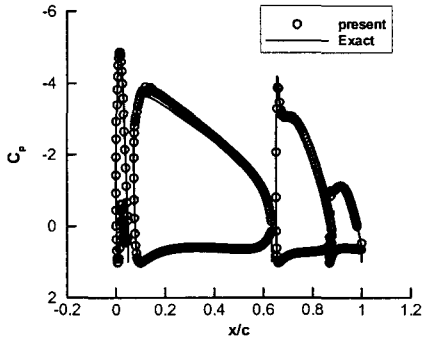

Fig. 10 Pressure coefficient distributions for four-element airfoil of Suddhoo and Hall.

위해 마하수로는 0.3을 사용하였다. Fig. 9는 계산에 사용된 격자 형상을 보이고 있으며, 9,123개의 격자점과 17,687개의 격자, 26,813개의 격자면으로 구성되었으며, 4개의 익형의 표면에는 489개의 격자점이 분포되어 있다. Fig. 10에서는 익형 표면에서의 압력 분포를 해석해와 비교하고 있으며, 두 번째 요소의 suction 부분을 제외하고 4개의 요소 모두에 대해서 해석해와 잘 일치하고 있는 것을 볼 수 있다. 본 예제에서 사용하는 마하수에 대해서 압축성 효과는 Prandtl-Glauert 식인 $1/\sqrt{1-M_\infty^2}-1$을 사용하면, 약 5% 정도가 되며, 해석해와의 오차는 이러한 압축성 효과에서 기인한 것으로 판단된다.

Fig. 11과 12는 반복 계산과 CPU 시간에 대한 수렴성을 각각 보이고 있다. CFL 수와 내부반복계산의 수는 앞절의 예제처럼 실제 계산에서 중요한 CPU 시간이 가장 적게 소요되는 값으로 조절되었다. CFL 수는 JT 연산자에 대해서는 10^7의 값이 사용되었으

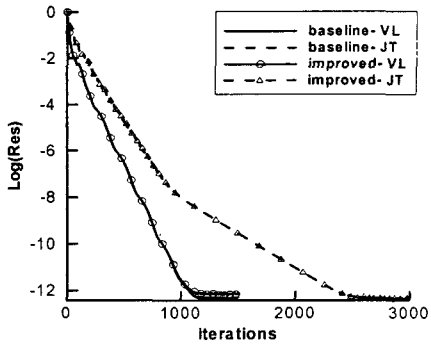

Fig. 11 Convergence rates vs iterations for four-element airfoil of Suddhoo and Hall.

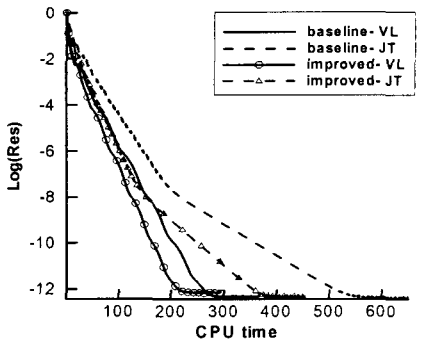

Fig. 12 Convergence rates vs CPU time for four-element airfoil of Suddhoo and Hall.

며, VL 연산자에 대해서는 10^3의 값이 사용되었다. 내부반복계산의 수는 JT와 VL 연산자 모두에 대해서 3의 값이 사용되었다. Fig. 11에서 VL 연산자가 JT 연산자 보다 기계오차 까지 수렴하는데 약 2배 정도의 빠른 것을 볼 수 있다. 하지만 이러한 성능 향상은 NACA 0012와 RAE 2822의 경우 보다는 적다. Fig. 12에서 JT 연산자에 대해서는 개선된 기법이 기본 기법에 비해서 약 30%, VL 연산자에 대해서는 약 20% 정도의 계산 시간이 적게 소요되는 것을 볼 수 있다. 또한 VL 연산자를 사용하는 개선된 기법은 JT 연산자를 사용하는 경우보다 기계오차까지 수렴하는데 계산 시간 면에서 약 2배 정도 빠른 것을 볼 수 있다. 하지만 대부분의 CFD 문제에서는 잔류치가 4개에서 6개 정도 감소하면 충분히 수렴된 해를 얻을 수 있으며, 이 정도의 잔류치에서는 JT 연산자와 VL 연산자의 수렴성이 거의 비슷하다.

하지만 대부분의 문제에서 VL 연산자를 사용하는 기법이 JT 연산자 보다 낮은 수렴성을 보이진 않는다.

6. 결 론

본 연구에서는 Part I에서 제안된 강건하고 효율적인 van Leer의 FVS 내재적 연산자에 대한 효율적인 수치적 프로그램화에 관한 연구를 수행하였다.

비대각 행렬을 미리 계산하여 저장하지 않고 필요한 순간에 계산하여 이용하는 방법을 사용하여 비대각 행렬을 저장하기 위한 기억 용량 요구량을 제거하였다. 이러한 접근을 통해서 본 연구에서 제안된 van Leer의 FVS 내재적 연산자를 사용하는 개선된 기법은 기본 기법의 약 30% 정도의 기억 용량만을 요구하며, 10~20% 정도의 계산 시간이 절약된다. 또한 본 연구에서 제안된 van Leer의 FVS 내재적 연산자를 사용하는 개선된 기법은 비정렬 격자계에서 많이 사용되는 Jameson과 Turkel의 내재적 연산자를 사용하는 개선된 기법의 강건성을 유지하면서 표준적인 검증 경우에 대해서 약 2배 정도 빠르다.

후기

본 연구는 국방과학연구소 수중운동체 기술 특화 연구센터의 '전산유체역학을 이용한 수중운동체의 선체-제어판-추진기 상호작용연구(SM-21)' 과제의 지원에 의한 결과의 일부이며, 이에 감사드립니다.

참고문헌

[1]김주성, 권오준, "비정렬 격자계에서 강건하고 효율적인 LU-SGS 기법 개발: Part I - 내재적 연산자," 한국전산유체공학회지, 제9권, 제3호, (2004).

[2]Sharov, D. and Nakahashi, K., "Reordering of 3-D Hybrid Unstructured Grids for Vectorized LU-SGS Navier-Stokes Computations," *AIAA Paper* 97-2102, (1997).

[3]Jameson, A. and Yoon, S.K., "Lower-Upper Implicit Schemes with Multiple Grids for the Euler Equations," *AIAA J.*, Vol.25, No.7, (1987), p.929-935.

[4]Chen, R.F. and Wang, Z.J., "Fast, Block Lower-Upper Symmertric Gauss-Seidel Scheme for Arbitrary Grids," *AIAA J.*, Vol.38, No.12, (2000), p.2238-2245.

[5]Luo, H., Baum, J.D., and Lohner, R., "A Fast, Matrix-free Implicit Method for Compressible Flows on Unstructured Grids," *J. Computational Physics*, Vol.146, (1998), p. 664-690.

[6]Jameson, A. and Trukel, E., "Implicit Schemes and LU Decompositions," *Mathematics of Computation*, Vol.37, No.156, (1981), p.385-397.

[7]Toro, E.F., "Riemann Solvers and Numerical Methods for Fluid Dynamics," Springer, 2nd ed., (1999).

[8]Saad, Y., "Iterative Methods for Sparse Linear Systems," PWS publishing company, (1995).

[9]Lohner, R. and Galle, M., "Minimization of Indirect Addressing for Edge-Based Field Solvers," *AIAA Paper* 2002-0967, (2002).

[10]Pulliam, T.H., "Euler and Thin Layer Navier-Stokes Codes: ARC2D, ARC3D," *Notes for Comp. Fluid Dynamics User's Workshop*, Univ. of Tennessee Space Institute, (1984).

[11]AGARD Subcommittee C., "Test Cases for Inviscid Flow Field Methods," *AGRAD Advisory Report* 211, (1986).

[12]Suddhoo, A. and Hall, I.M., "Test Cases for the Plane Potential Flow Past Multi-Element Aerofoils," *Aeronautical Journal*, Vol.89, (1985), p.403-414.