기술기사

대형구조물을 위한 병렬 구조해석 및 설계

박 효 선*

1. 서 론

최근 공학 분야에서 다루어지고 있는 문제들은 크기의 대형화와 해석법 복잡화로 특징지어질 수 있다. 이러한 특징을 갖는 공학 문제들의 해결을 위한 모델링 및 해석 기법은 전산 공학 분야의 급속한 발달로 새로운 형식의 해석 기법으로 개발되거나 전산기 자체의 성능 향상에 의해 기존 해석법의 효율성이 높아지고 있다.

전산 공학 분야의 연구 결과 및 발달 중 전산기를 이용하는 학문 분야에 가장 큰 영향을 미치고 있는 것은 여러 개의 프로세스를 효율적으로 이용할 수 있는 병렬컴퓨터(parallel machine)의 개발이다. 이러한 컴퓨터를 이용한 병렬계산법(parallel computation or parallel processing)은 공학 분야에서 다양한 형식으로 개발되어 널리 사용되고 있으며, 특히 구조공학 분야에서는 종전의 한 개의 프로세스를 이용한 직렬해석 및 설계법(serial analysis and design)이 병렬해석 및 설계법(parallel analysis and design)으로 전환 개발되어 활발하게 이용되고 있다.[1-3]

2. 병렬 계산의 기본 개념

전산 공학의 급속한 발달에도 불구하고 한 개 프로세스의 기계적 성능 향상에 의한 계산 속도의 증가에는 한계가 있으므로 여러 개의 프로세스들을 같은 프로그램 수행시 동시에 사용함으로써 전체적인 계산 속도를 증가시키고자 하는 것이 병렬계산법의 기본 개념이다. 이러한 개념은 다음과 같은 프로그램의 일부분을 예로 들어 설명될 수 있다.

```
DO I = 1, N
    C(I) = B(I) + A(I)
ENDDO
```

한 개의 프로세스를 이용할 경우 위의 반복 계산에 필요한 시간이 T_1이었다면 N_p개의 프로세스를 사용하였을 경우 이론적으로는 그림 1에서와 같이 각 프로세스가 N/N_p개의 반복 계산을 동시에 수행함으로써 전체 계산 시간이 T_1/N_p로 줄어들게 된다.

* 영남대학교 건축공학과, 전임강사, 공학박사

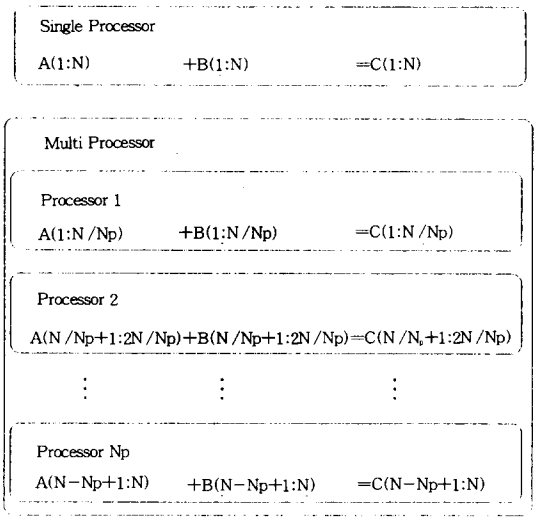

그림 1 병렬계산 기본 개념도

이러한 개념의 병렬계산법 효율성은 speedup으로 평가되며 이는 다음의 식으로 표현된다.

$$S(Np) = \frac{T_1}{T_{Np}} \tag{1}$$

T_1와 T_{Np}는 각각 1개와 Np개의 프로세스를 이용하여 프로그램을 수행하는데 걸리는 시간을 나타내며, speedup의 최대치는 Np가 된다. 그러나 프로그램 전체의 병렬계산은 불가능하고, 프로그램은 항상 병렬 및 직렬 계산 부분으로 구성되어져 있으므로 speedup은 직렬 계산 부분이 증가하면 감소하게 된다. 프로그램의 직렬 계산 부분의 비율이 f이고 Np개의 프로세스를 이용하는 경우 speedup은 다음과 같이 계산된다.

$$T_1 = T_{serial} + T_{parallel} = fT_1 + (1-f)T_1 \quad (2$$

$$T_{Np} = fT_1 + \frac{(1-f)T_1}{N_p} \tag{3}$$

$$S(Np,f) = \frac{T_1}{T_{Np}} = \frac{fT_1+(1-f)T_1}{fT_1+\frac{(1-f)T_1}{Np}} \tag{4}$$

$$= \frac{1}{f+\frac{1-f}{Np}} = \frac{Np}{f\,Np+1-f}$$

Amdahl의 법칙으로 불리는 식(4)는 사용하는 프로세스의 수에 무관하게 speedup이 f의 역수보다 작음을 나타낸다. f가 0.1인 경우, 사용하는 프로세스의 수에 따른 speedup의 분포는 그림 2와 같이 프로세스의 수를 무한으로 증가시켜도 10이상을 넘지 못함을 알 수 있다.[4)]

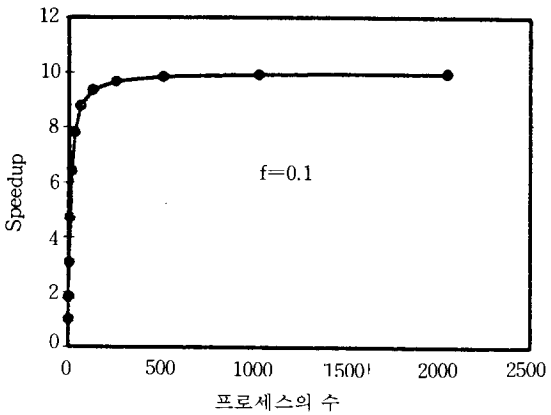

그림 2 프로세스의 수에 따른 Speedup의 분포

효율적인 병렬 계산은 사용하는 컴퓨터의 사양에도 관계가 있지만 알고리즘 자체의 병렬성(parallelism)에 의존하게 된다. 그러므로 기존의 직렬계산 프로그램을 단순히 병렬프로그래밍 언어로 전환해서는 기대하는 speedup을 얻을 수 없으므로 풀고자 하는 문제를 위한 병렬알고리즘(parallel algorithm)을 먼저 개발한 후 병렬계산을 해야 된다. 이에 따라 직렬계산에서는 효율적이던 알고리즘이 비효율적으로 될 수 있으며 반대의 경우도 종종 발생한다.

구조물의 해석시 가장 많은 계산 시간을 요하는 방정식해법의 경우 직렬계산에서는 가우스소거법 등의 직접법(direct method)이 널리 이용되지만 해석 시간의 단축은 사용하는 컴퓨터의 성능에 의존하고 있다. 그러나 병렬계산의 경우 반복해법(iterative method)을 이용한 병렬유한요소법이 효율적이며 가장 많이 사용되는 해법이 Conjugate-Gradient Method이다. 미국, 유럽에서는 자유도수가 많은 대형구조물의 해석에는 반복해법을 이용한 병렬유한요소법이 널리 이용되고 있다.

3. 병렬 컴퓨터의 분류

많은 종류의 병렬컴퓨터가 있지만 기종에 무관하게 프로그래밍의 관점에서 명령어 흐름(instruction stream)과 데이터 흐름(data stream)에 따라 4가지로 분류된다. 명령어 흐름은 컴퓨터에 의해서 수행되는 명령어의 순서이며 데이터 흐름은 하나의 명령어 수행을 위해 사용되는 데이터의 흐름이다.[5)]

3.1 SISD(Single Instruction Stream, Single Data Stream)

Von Neumann컴퓨터로 알려져 있고 가장 흔히 볼 수 있는 SISD는 직렬컴퓨터로서 모든 명령과 그에 따른 데이터의 이동이 순차적으로 수행되며 구성은 그림 3과 같다.

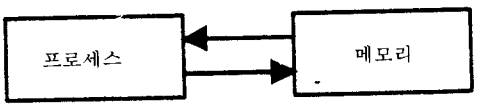

그림 3 SISD의 구성

3.2 SIMD(Single Instruction Stream, Multiple Data Stream)

SIMD에서는 여러 개의 프로세스가 하나의 콘트롤 프로세스에 의해 제어되므로 각 프로세스는 같은 명령을 하나의 콘트롤 프로세스에서 동시에 전달받아 자체 데이터 또는 공유 데이터를 독립적으로 이용한다. SIMD의 구성은 그림 4와 같다.

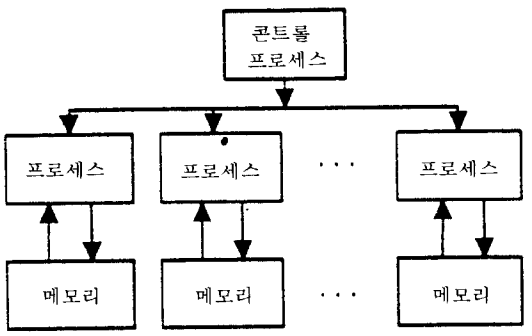

그림 4 SIMD의 구성

3.3 MISD(Multiple Instruction Stream, Single Data Stream)

MISD는 명령과 데이터의 단순한 조합에 의하여 만들어진 형식으로 병렬계산의 효율성이 떨어져 사용되지 않고 있다.

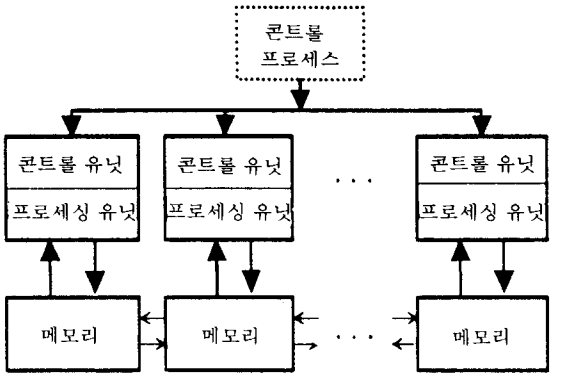

그림 5 MIMD의 구성

3.4 MIMD(Multiple Instruction Stream, Multiple Data Stream)

MIMD에서는 여러 개의 프로세스가 자체의 콘트롤 프로세스에 의해 제어되므로 각 프로세스는 동시에 독립적으로 다른 명령을 자체 데이터 또는 공유 데이터를 이용하여 수행한다. 그리고 전체적인 프로세스들을 콘트롤하는 프로세스의 유무는 기종에 따라 다르며 SIMD의 구성은 그림 5와 같다.

4. 병렬 구조해석을 위한 Preconditioned Conjugate Gradient Method

병렬 계산을 위한 프로그래밍 기법에는 Data parallel, Message passing 그리고 Work sharing 등이 있으며 참고문헌에 잘 나타나 있다. 본 기사에서는 병렬 구조해석 및 설계를 위한 병렬 선형 방정식의 해법으로서 Preconditioned conjugate gradient algorithm을 CRAY T3D work sharing 기법을 이용하여 소개한다.[6)]

4.1 CRAY T3D System

1993년에 출시된 CRAY T3D는 150 Mflops (million floating point operations per second)의 계산 성능과 8Mword(64 MB)의 메모리를 갖는 마이크로 프로세스 2n(n= 1, 2, ···, 11)개로 구성되어져 있는 병렬 계산기이다.[7] T3D system에서는 데이터가 두 가지 형식으로 분류되어 있으며 이는 shared data와 private data이다. private data의 경우, 각 프로세스는 프로그램 수행 중 초기에는 같은 값을 private data로써 갖지만 수행과정 중 각 프로세스가 다른 값을 가질 수 있다. 그러므로 같은 변수에 해당하는 data 값이 서로 다를 수 있다. 그러나 shared data는 전체 프로세스에 골고루 나누어져 저장되므로 각 프로세스는 shared data의 일부분을 나누어 가지게 되고 shared data로 표시된 변수는 같은 값을 가지게 된다.

T3D system에서 프로그램은 세 가지 형식으로 수행되며 각각은 parallel region, serial region 그리고 work sharing region이다. 기본적으로 프로그램은 사용자가 지정하지 않으면 모든 프로세스가 같은 명령을 수행하는 parallel region으로 분류되며 serial region에서는 사용자가 지정한 하나의 프로세스만이 명령을 수행하게 된다. work sharing region에서는 iteration들이 모든 프로세스에 나누어져 수행되게 되며 각 프로세스는 사용자에 의해 할당된 iteration들을 수행하게 된다. 그러므로 효율적인 병렬알고리즘의 개발은 각 프로세스 간의 일의 균등한 분배(load balancing)와 각 프로세스에 할당된 명령 및 iteration에 상응하는 Data를 해당 프로세스에 분포시키는 것(data locality)에 좌우되게 된다.

4.2 Work Sharing Preconditioned Conjugate Gradient Algorithm

직접법과는 달리 반복해법인 conjugate gradient method은 주 계산이 벡터와 벡터의 곱, 벡터와 매트릭스의 곱, 그리고 매트릭스와 매트릭스의 곱이므로 구조물을 구성하는 부재와 절점을 각 프로세스에 균등히 할당하여 계산할 수 있는 병렬성(parallelism)을 가지고 있다. 구조물 전체의 부재수가 M이고 프로세스의 수가 Np이면 각 프로세스에는 Nm=M/Np개의 부재가 할당된다. 그리고 부재의 단면 성능을 해당 프로세스에 저장함으로써 각 프로세스는 Nm개의 부재 강성 매트릭스를 계산하게 된다. 그리고 conjugate gradient method에서는 계산된 부재 강성 매트릭스는 전체 강성 매트릭스(global stiffness matrix)로 조합될 필요가 없으므로 효율적인 병렬 알고리즘의 개발이 가능하다.

최적화 이론에 근거한 conjugate gradient method의 수렴성을 증가시키기 위하여 다양한 형식의 preconditioner가 적용되고 있으나 여기서는 전체 구조물의 자유도수(Nd)와 같은 개수의 요소를 갖는 diagonal preconditioner를 각 부재 강성 매트릭스의 대각선 요소를 이용하여 구성하고 각 프로세스에 Ndp=Nd/Np 개수의 요소를 할당하여 저장한다. conjugate gradient method에 필요한 다른 모든 벡터들 또한 preconditioner와 같은 방식으로 각 프로세스에 할당되며 그러한 벡터는 하중, 변위, residual 그리고 search direction 벡터 등이 있다. 그러므로 그러한 벡터들에 관련된 작업은 모든 프로세스들에서 동시에 가능하게 된다.

전체 프로세스들에 골고루 할당되어 분포된 두 벡터 간의 곱은 세 단계로 이루어진다. 먼저, 각 프로세스에 할당된 요소들 간의 곱이 각 프로세스에서 이루어진다. 두 벡터 요소들의 곱을 합한 부분합들은 각 프로세스 내에서 이루어진다. 마지막으로 전체의 프로세스들간의 부분 합이 더해져서 두 벡터의 곱이 계산된다.

Conjugate gradient method에서 search direction 및 Residual 벡터의 계산을 위해 필요한 부재 강성 매트릭스(K)와 Search direction 벡터(p)의 곱은 벡터 p를 $2N_2 \times M$의 direction 매트릭스로 변환하여 각 프로세스에 Nm 수의 부재 강성 매트릭스와 Nm 수의 direction 매트릭스를 할당함으로써 모든 프로세스들이 동시에 Nm 수의 매트릭스와 매트릭스의 곱을 수행하게 된다. 여기서 N_2는 절점당 자유도 수이며 K는 $2N_2 \times 2N_2 \times M$의 요소를 가진다. 마지막으로 계산된 direction 매트릭

스는 Nd 요소를 갖는 direction 벡터 q로 전환되어 나머지 벡터들 간의 계산에 사용되어진다. 각 프로세스에서 수행되는 이러한 과정은 그림 6에 나타나 있다.

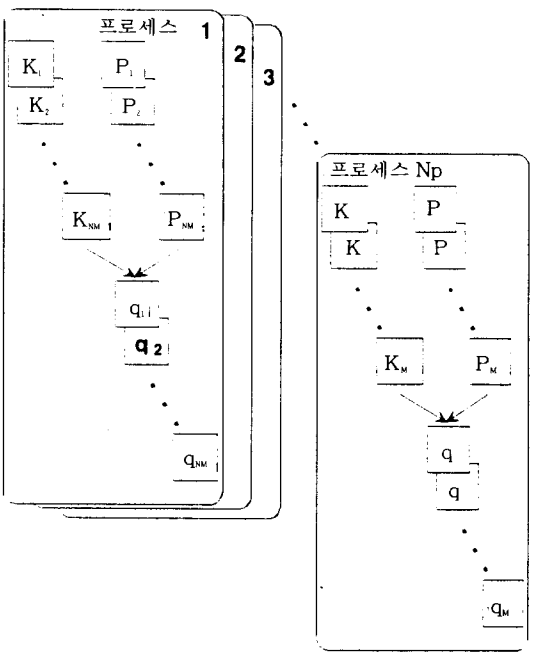

그림 6 Work sharing direction 벡터의 계산

부재별 그리고 자유도별 parallelism을 이용한 병렬 preconditioned conjugate gradient algorithm은 다음과 같이 주어진다.

Step 1. 부재 강성 매트릭스 $K(2N_2, 2N_2, M)$의 구성 및 Connectivity 매트릭스 $IDJ(2N_2, M)$를 이용하여 Diagonal Preconditioner, $D(Nd)$의 구성. 각 프로세스는 $2N_2 \times 2N_2 \times Nm$, Ndp 요소의 K와 D를 계산한다.

Step 2. 초기 변위 벡터 $u_0=0$를 이용하여 초기 residual, search direction, temporary 벡터들, 계수 r_0, p_0, z_0, ρ를 계산한다. 각 프로세스는 동시에 Ndp 요소의 각 벡터를 계산한다.

$$r_0(1:Nd)=P(1:Nd)-K(2N_2,2N_2,M)\ u_0(1:Nd)$$

$$p_0(1:Nd)=r_0(1:Nd)$$

$$z_0(1:Nd)=D^{-1}(1:Nd)\ r_0(1:Nd)$$

$$\rho= z_0^T(1:Nd)\ r_0(1:Nd)$$

Step 3. Residual 벡터의 norm을 이용하여 preconditioned conjugate gradient 알고리즘의 수렴도를 검토한다.

if $\|r\| \le \varepsilon$, then u가 수렴된 절점 변위 벡터, 아니면 나머지 step들 수행. ε는 stopping tolerance.

Step 4. Direction 벡터를 connectivity 매트릭스 $IDJ(2N_2, M)$를 이용하여 이차원 임시 매트릭스ptemp로 치환한다.

$$ptemp(2N_2, M) = p(IDJ(2N_2, M))$$

Step 5. 부재 단위의 병렬 매트릭스 · 매트릭스 곱을 전체 프로세스에서 동시에 계산한다.

$$vtemp(2N_2, M)=K(2N_2, 2N_2, M)\ ptemp(2N_2, M)$$

Step 6. ptemp를 connectivity 매트릭스 $IDJ(2N_2, M)$를 이용하여 벡터 v(Nd)로 치환한다.

$$v(IDJ(2N_2, M))=v(IDJ(2N_2, M))+ptemp(2N_2, M)$$

Step 7. Search direction의 step size $\gamma=\frac{\rho}{\alpha}$를 계산한다.

$$\alpha = p^T(Nd) \cdot v(Nd)$$

Step 8. 개선된 변위 벡터를 계산한다. 각 프로세스는 동시에 Ndp 요소의 각 벡터를 계산한다.

$$u(1:Nd) = u(1:Nd) + \gamma p(1:Nd)$$

Step 9. 새로운 residual 벡터 r를 계산한다. 각 프로세스는 동시에 Ndp 요소의 각 벡터를 계산한다.

$$r(1:Nd) = r(1:Nd) + \gamma p(1:Nd)$$

Step 10. 새로운 search direction 벡터를 계산한다. 각 프로세스는 동시에 Ndp 요소의 각 벡터를 계산한다.

$$p(1:Nd) = z(1:Nd) + \beta\, p(1:Nd)$$
$$z(1:Nd) = M^{-1}(1:Nd)\, r(1:Nd)$$
$$\rho = z(1:Nd)\, r(1:Nd)$$
$$\beta = \rho_i / \rho_{i-1}$$

step 3으로 돌아간다.

5. 예 제

개발된 병렬해석법과 병렬구조 최적화 알고리즘을 이용하여 그림 7과 같은 147층 초고층 철골 트러스 구조물의 최소 중량설계에 적용하였다.[6,8] 예제 구조물의 자유도수는 1801이며 8904개의 부재를 갖는 envelop structure로 세장비는 7.2를 갖는다. 본 예제에 사용된 컴퓨터 시스템은 최소 4개에서 최대 32개의 프로세스로 구성될 수 있는 T3D system과 최소 32개에서 최대 256개의 프로세스로 구성될 수 있는 TMC(Thinking Machine Corporation) CM-5 시스템을 사용하였다.[9] 각 경우 Speedup의 분포는 그림 8과 같다.

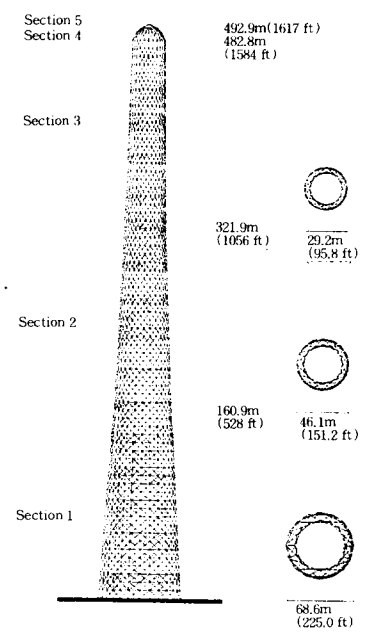

그림 7 147층 트러스 Envelop structure

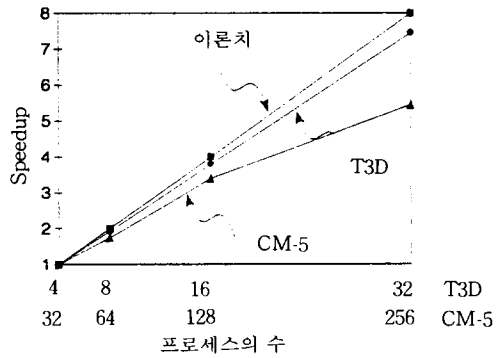

그림 8 147층 트러스 구조물의 Speedup

CM-5를 사용한 경우 구조물의 해석에 필요한 CPU시간은 1.78초가 소요되었으며 전체적인 최소 중량 설계는 임의의 초기 설계에서 시작하여 미국의 설계 규준인 AISC ASD 그리고 LRFD 규준에 적합한 최종 설계를 수행하는데 8.9분 그리고 9.2분이 소요되었다.

6. 결 론

공학 전반에 걸쳐 다양한 형식으로 개발되어 사용되고 있는 병렬계산법의 기본 개념과 병렬계산기의 분류에 대하여 소개하였으며, 구조해석시 가장 많은 시간을 요하는 방정식해법을 preconditioned conjugate gradient를 이용하여 병렬화하는 과정과 병렬알고리즘을 소개하였다. 그리고 소개된 병렬방정식해법을 대형구조물의 해석 및 설계에 적용하여 병렬계산의 효율성을 speedup을 이용하여 도표화하였다.

특히 초고층 건물의 구조해석 및 설계는 작용하는 하중 및 역학적 특성이 중·저층 구조물과는 다르다는 점을 제외하고도 구조물을 구성하는 부재 및 자유도수가 많다는 점에서 기존의 중·저층 구조물의 해석 및 설계와는 다른 문제점들을 내포하고 있다. 즉, 기존의 구조해석용 알고리즘 및 소프트웨어를 단순하게 초고층 구조시스템해석에의

적용시에는 수행시간 및 필요로 하는 메모리의 관리 측면에서 어려움이 있으며, 특히 실무자들이 피부로 느끼게 되는 지나치게 긴 수행 시간은 반복적 해석을 필요로 하는 구조설계 작업의 효율을 제한하는 큰 장애가 된다.

병렬해석 및 설계의 필요조건들은 앞에서 언급한 하드웨어 및 프로그래밍 기법의 선정 그리고 병렬알고리즘의 개발을 들 수 있다. 국내의 실정을 고려하면 병렬컴퓨터의 확보 및 이의 이용에는 여건상 어려움이 많으므로 PVM(parallel virtual machine) 또는 WPVM을 통하여 여러 대의 Workstation들 또는 여러 대의 PC들을 묶어서 슈퍼컴퓨터 및 병렬컴퓨터의 효과를 낼 수 있는 분산해석법(distributed structural analysis and design)의 개발은 프로젝트의 규모를 고려하여 시급히 개발, 실용화되어야 한다.[10]

참 고 문 헌

1. Adeli, H. and Kamal, O., Parallel Processing in Structural Engineering, Elsevier Applied Science, New York, 1993.
2. Johnsson, S. L. and Marthur, K. A., "Data Structures and Algorithms for the Finite Element Method on a Data Parallel Supercomputers", International Journal for Numerical Methods in Engineering, Vol. 29, pp. 881-908, 1990.
3. Kumar, S. and Adeli, H., "Distributed Finite-Element Analysis on Network of Workstations-Implementation and Applications", Journal of Structural Engineering, ASCE, Vol. 121, No. 10, pp. 1456-1462, 1995.
4. Robert, Y. The Impact of Vector and Parallel Architecture on the Gaussian Elimination Algorithm, Manchester University Press, UK, 1990.
5. Adeli, H., Parallel Processing in Computational Mechanics, Marcel and Dekker, New York, 1992.
6. Park, H. S. and Adeli, H., "Distributed Neural Dynamics Algorithms for Optimization of Large Steel Structures", Journal of Structural Engineering, ASCE, accepted for publication.
7. Cray, CRAY T3D System Architecture Overview Manual, Cray Research, Inc, Eagan, MN, 1993.
8. Adeli, H., and Park, H. S., "Hybrid CPN-Neural Dynamics Model for Discrete Optimization of Steel Structures", Microcomputers in Civil Engineering, Vol. 11, pp. 355-366, 1996.
9. Thinking Machine Corporation, CM-5 CM Fortran Programming Guide, Ver. 2.1, TMC, Cambridge, MA, 1994.
10. Geist, G. A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V. S., PVM User's Guide and Reference Manual, Tech. Rep. ORNL/TM-12187, Engrg. Phys. and Math. Div. Oak Ridge Nat. Lab., Oak Ridge, Tenn, 1993.