並列計算機 Cenju 上の有限要素法による非線形変形解析

加納 健[†] 中田登志之[†] 奥村秀人[††]
大竹邦彦[††] 中村 孝[††] 福田正大[††]
小池誠彦[†]

衝撃解析等の非線形問題の有限要素法では，剛性行列の作成処理を Newton-Raphson ループの繰り返しごとに行わなければならないため，ベクトル計算機では高速に処理することが困難である．このような処理の大部分を占める，剛性行列の作成と，線形求解について，それぞれ，並列計算機を用いて高速に処理する手法を提案し，並列シミュレーションマシン Cenju を用いて評価を行った．剛性行列の作成では，バッファを用いることにより，要素剛性行列を集める処理でのプロセッサ間同期を取り除いた．その結果，64台のプロセッサで49倍の速度向上率を得た．線形求解では，反復法の一種である SCG (Scaled Conjugate Gradient) 法を並列化した．不必要なプロセッサ間通信を削減することで，38倍の速度向上率を得た．また，さらに性能を向上させるため，処理の局所性を増し，プロセッサ間通信を削減する，要素のプロセッサへの割り当てと節点自由度の番号付けの方法を提案した．その結果，剛性行列の作成処理では，64台で60倍の速度向上率が得られた．この方法を用いることにより，問題であったバッファ容量を削減することもできた．さらに，この割り当て方法は，並列 SCG 法でもプロセッサ間通信を削減できることを確認した．

Evaluation of Nonlinear Deformation Analysis by Finite Element Method on the Parallel Machine Cenju

YASUSHI KANOH,[†] TOSHIYUKI NAKATA,[†] HIDEHITO OKUMURA,[††] KUNIHIKO OHTAKE,[††]
TAKASHI NAKAMURA,[††] MASAHIRO FUKUDA[††] and NOBUHIKO KOIKE[†]

In the nonlinear deformation analysis, there are two main stages, namely, generation of a stiffness matrix and solution of a linear equation. In this paper, we present the methods of parallelizing the two stages, and their evaluation on the parallel machine *Cenju*. In generating the stiffness matrix, we parallelized its assembly process by using extra buffers for each element of the stiffness matrix in order to reduce the synchronization. We gained 49 times speedup using 64 PEs. In solving a linear equation, we tried to parallelize Scaled Conjugate Gradient (SCG) Method, and gained 38 speedup using 64 PEs by reducing amount of interprocessor communications. Moreover, we propose the strategy for assigning elements to processors and renumbering nodal point degrees of freedom which improves locality of computation and reduces interprocessor communications. Using this strategy, we attain 60 times speedup in generating the stiffness matrix. Moreover, this strategy can also reduce amount of interprocessor communication in parallel SCG Method.

1. はじめに

一般に有限要素法では，対象物の挙動をいくつかの要素の集合体の節点自由度の運動ととらえる．対象物の変位，速度，加速度は，要素の辺上にある節点の自由度方向に対して求められる．対象物に対して荷重ベクトル$\vec{L}$が加えられたときの各自由度の変位$\vec{D}$は，対象物の剛性行列Kによって関係付けられ，

$$K\vec{D}=\vec{L}$$

となる．(**図1**参照)

非線形問題では，K，$\vec{L}$は，$\vec{D}$の関数であり，近似解$\vec{D}$に対して，剛性行列Kと，荷重ベクトル$\vec{L}$を求め，変位$\vec{D}$が収束するまで，連立一次方程式の求解を繰り返す．衝撃解析等で問題となる有限要素法の非線形構造問題では，Newton-Raphson 法の逐次近似の毎ステップごとに剛性行列を作り直さないと収束が悪い場合がある[1]．1回の Newton-Raphson ループの中では，剛性行列の作成と線形求解が主な処理である(**図2**参照)．このような解析の場合は，線形解析の有限要素法と異なり，剛性行列の作成部分が計算時間の大きな割合を占める．従って，非線形問題を高速に処理するためには，剛性行列の作成部分と連立一次方程

† 日本電気(株) C&C システム研究所
C&C Systems Research Laboratories, NEC Corporation
†† 航空宇宙技術研究所
National Aerospace Laboratory

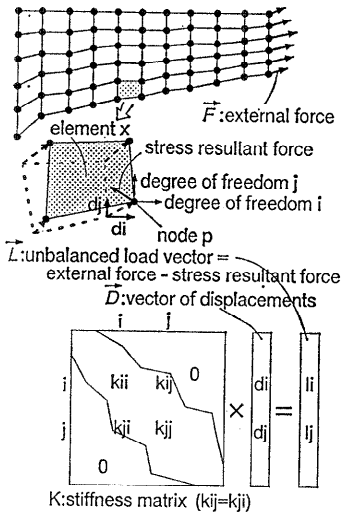


図 1 構造解析における有限要素法
Fig. 1 Finite element method for structural analysis.

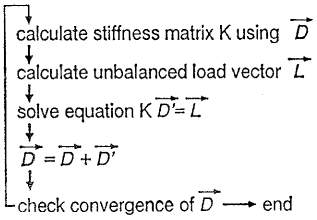


図 2 Newton-Raphson ループ
Fig. 2 Newton-Raphson loop.

式の求解部分を並列化する必要がある.

剛性行列の作成処理は，要素の種類によって異なった処理を行わなければならず，ベクトル化が難しい問題である．しかしながら，処理の前半の部分（4章**処理1**）は，各要素ごとに，完全に独立に計算でき，並列処理効果が得られやすい問題である．一方，連立一次方程式の求解処理では，行列が対称行列で，スパースなバンド行列という特徴を持っており，並列化するには工夫を要する．2章でも述べるように，従来から，有限要素法の並列化は試みられていたが，プロセッサ間通信のオーバヘッドやプロセッサ間の負荷の不均衡などの問題があった.

本論文では，米国カリフォルニア大学で開発された構造解析用プログラムである NONSAP[2]の剛性行列作成の部分と，反復法の一種である SCG (Scaled Conjugate Gradient) 法[3]を，それぞれ，並列化し，Cenju[4]を用いて評価を行った．また，プロセッサ間通信を削減するための，要素のプロセッサへの割り付けと，節点自由度の番号付けの方法を提案した．この方法を用いることにより，剛性行列作成時に並列処理のオーバヘッドとなっていたプロセッサ間通信を削減できた．さらに，このような節点自由度の番号付けは，線形求解での並列 SCG 法でもプロセッサ間通信を削減できることを示す.

以下，2章では，従来の研究を概説する．3章では，実験に用いた並列シミュレーションマシン Cenju についてふれる．処理の大部分を占める剛性行列の作成処理と線形求解の並列化方法と実験結果については，4章と5章で示す．6章では，処理の局所性を高める要素のプロセッサへの割り付けと，節点自由度の番号付けの方法を示し，その効果について述べる.

2. 従来の研究

有限要素法の並列化の研究は，最初に並列計算機が開発されたころから，さまざまなところで行われてきた．有限要素法はその問題の性質上，並列処理に向いた問題と考えられてきた．しかしながら，効率の良い並列処理の障害となる，プロセッサ間通信や，プロセッサ間の負荷の不均衡などの問題は，他のアプリケーションと同じように存在する．本章では，従来の研究で，これらの問題がどのように扱われているかについて，簡単に紹介する.

2.1 隣接プロセッサ間通信

有限要素法のモデルと並列計算機の構成とを1対1にマッピングすることにより，並列処理を行う処理方式が提案されている．NASA の FEM (Finite Element Machine)[5]や，筑波大学の PAX-128[6]上での有限要素法がこれにあたる．これらの計算機では，高速な隣接プロセッサ間の通信によって，プロセッサ間通信のオーバヘッドを削減している.

FEM は2次元格子状に配置されたプロセッサがネットワークによって8近傍のプロセッサと接続されている．FEM 上での Newmark Beta 法による過渡解析では，8台プロセッサで 7.83 倍の速度向上率が報告されている.

PAX-128 は 128 台のプロセッサが2次元トーラス状に接続されている．PAX-128 上では，2次元の弾性問題が並列化されており，線形求解には CG 法が使われている．問題の規模が大きく，要素がプロセッサに均等に割り当てられた場合にはほぼ線形に近い速度向上率が報告されている.

しかしながら，これらの処理方式では，計算機の構

成に有限要素法のモデルがうまくマッピングできない場合には十分に性能が発揮できないと思われる．この原因としては，(1)プロセッサ間通信が隣接プロセッサとの規則的な通信に限定されなくなり，プロセッサ間通信のオーバヘッドが大きくなる，(2)プロセッサ間の負荷の不均衡，などが考えられる．

2.2 領域分割法

1つのプロセッサが行う処理の粒度を大きくとることにより，プロセッサ間通信を削減する方法として，領域分割法を用いた並列有限要素法が提案されている[7]．この方法は，問題を複数の領域に分割し，各領域を1つの部分問題ととらえ，その各々に対して有限要素法解析を行う．その結果得られた境界部分の拘束条件を解いて境界部分の解を求める．これを解が収束するまで繰り返す．そして，最後に得られた境界部分の解を用いて，全体の解を求めるというものである．この方法では，各部分問題を解く処理は，完全に独立に行えるので，この部分ではプロセッサ間通信はまったく起こらない．

この方法では，領域分割の質が性能を左右すると思われる．すなわち，(1)部分問題間の負荷の不均衡，(2)境界部分の処理量の増大，が問題となる．負荷の不均衡は，プロセッサの使用効率を下げ，部分問題の解析時間を大きくする．また，境界部分の求解は，基本的にオーバヘッドにあたる処理である．文献7)では，領域数を多くし，部分問題の処理時間を短くして，プロセッサに部分問題を動的に割り当てる方法をとって，(1)の問題を解決している．

2.3 プロセッサ間通信量の削減

並列処理のオーバヘッドとなるプロセッサ間通信を削減する工夫を行っている研究もある．

文献8)では，あるプロセッサに格納する剛性行列の成分を計算するために必要な要素剛性行列の成分だけをそのプロセッサで計算することにより，剛性行列作成時のプロセッサ間通信を無くす方法を提案している．しかしながら，要素剛性行列の作成処理は，要素内のすべての節点自由度の情報が必要な処理と，各節点自由度ごとに計算できる処理とからなっており，これらの比率は要素モデルによって異なるので，すべての場合に効率良い処理とは限らない．

文献9)では，1次元のプロセッサアレイで処理する場合の処理方式を提案している．行列のバンド幅を削減するアルゴリズム[10]を用いて，剛性行列のバンド幅が小さくなるように，節点自由度の番号付けを行うことにより，プロセッサ間通信を隣接のプロセッサ間だけに限定している．また，この結果をもとにプロセッサへの要素，節点自由度の割り付けを行い，負荷が均等になるようにしている．しかし，この割り付け方法は1次元のプロセッサアレイ用で，一般の計算機の構成には適用できないと思われる．

本研究も，この範疇に位置付けられる．提案する処理方式では，要素剛性行列を作成したプロセッサに，それを使って作られる剛性行列の成分を可能な限り格納するように，要素のプロセッサへの割り付けと，節点自由度の番号付けを行う．この割り付け方式は，文献9)の方法とは逆に，要素のプロセッサへの割り付けを最初に行い，その結果を使って，節点自由度の番号付けを行う．このようなアプローチを採ったのは，剛性行列の作成が処理時間の大部分を占めるため，要素のプロセッサへの割り付けを優先して行うためである．

節点自由度の番号付けの結果，剛性行列作成時のプロセッサ間通信を削減でき，64台で60倍の高い速度向上率が得られた．また，この節点自由度の番号付けは，線形求解で行う並列SCG法でもプロセッサ間通信を削減できることが確認できた．詳しくは，6章で述べる．

3. 並列シミュレーション・マシン Cenju

並列シミュレーションマシン Cenju は8個のクラスタで構成され，各クラスタ間はパケット交換の多段網で接続されている．各クラスタは，バスで接続された8台の要素プロセッサ (PE) で構成されている．(**図3**参照)

メモリ空間は各プロセッサに分割された分散共有メモリの構成を採っており，アドレスにより他のプロセッサのメモリをアクセスすることができる．

また，クラスタ内とクラスタ外では，他のプロセッサのメモリのアクセス時間が大きく異なるが，プログラマにはすべて同じ論理的なプロセッサとして見え，クラスタ内外の区別はない．

Cenju では，クラスタ間の read は，write の40倍の時間（レイテンシ）がかかるため，データの生産者プロセッサが，消費者プロセッサのメモリにデータを書き込むというプログラミングスタイルとなる．

Cenju では，並列ライブラリ関数を付加したCおよびFORTRANを使ってプログラムを記述する．並列ライブラリ関数には，複数のプロセッサに同じメモリ

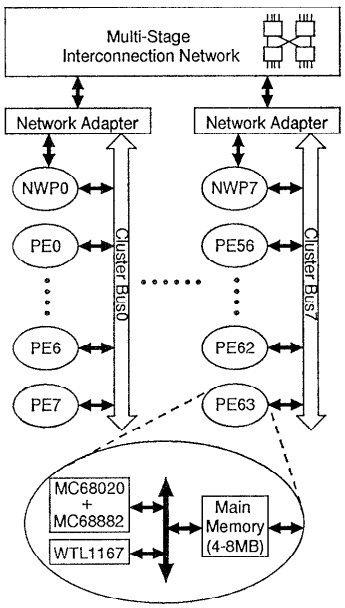


図 3 Cenju のブロック図
Fig. 3 Block diagram oi Cenju.

イメージをコピーし，並列処理を開始する fork や，すべてのプロセッサ間の同期をとる barrier などが用意されている．

4. 剛性行列の作成部分の並列化

4.1 剛性行列の作成

剛性行列Kは，各要素の要素剛性行列の足し合わせによって求められる．荷重ベクトル$\vec{L}$は，外力と，各要素でのひずみによる応力ベクトルとの和から計算される．要素剛性行列，応力ベクトルは，その要素の材質，形状等の情報，要素内節点自由度の変位から計算される．

剛性行列を作成する処理は大きく次の2つに分けられる．

処理1 要素の情報，要素内自由度の変位から要素剛性行列，応力ベクトルを計算する．

処理2 要素剛性行列，各要素の応力ベクトルを足し込むことにより，剛性行列，荷重ベクトルを計算する．

処理1は，各要素間で並列に計算できる．また，同じ要素特性を持つ要素は，ほぼ同じくらいの処理量となる．

処理2は，並列化が難しい問題で，並列化する場合のボトルネックとなる処理である．

実験に用いたデータは，2次元アイソパラメトリック要素の非線形問題である．平面応力のみを生ずるモデルで，要素数，節点自由度は，それぞれ，1266個，2743個である．

4.2 並列処理方式

処理1の部分は，要素ごとに独立して計算できるため，要素を単位として複数のプロセッサに処理を割り付ける．

作成する剛性行列は，列（節点自由度）ごとに各プロセッサに分散格納する．これは，**処理2**（部分剛性行列の足し込み）を並列に行うためである．また，このように格納すると，次に行う処理である線形求解を並列処理する場合に都合が良い．

分散格納した剛性行列の各成分には，それぞれ，**処理1**の結果を格納するためのバッファを用意する．このバッファは，**処理1**を行ったプロセッサからプロセッサ間通信によって転送された要素剛性行列の成分を格納するためのものである．このバッファの大きさは，**図4**のような要素構成の場合，最大で4個の成分を格納するだけの容量が必要となる．

以下に処理手順を示す．

すべてのプロセッサが次の処理を行う．

step 1 各プロセッサは要素の要素剛性行列，応力ベクトルを計算し，各成分の該当するバッファを格納しているプロセッサのメモリに書き込む．
割り当てられた要素すべてについてこの処理を繰り返す．（**図5**）

barrier 1 すべてのプロセッサが step 1 を終了したことを保証する．

step 2 各プロセッサは，バッファ内の値を加え，割り当てられた剛性行列，荷重ベクトルの成分を計算する．（**図6**）

図7に本方式での速度向上率のグラフを示す．プロ

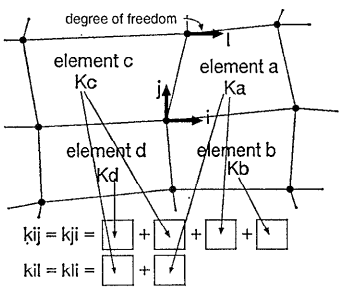


図 4 必要となるバッファの数
Fig. 4 Number of buffers needed.

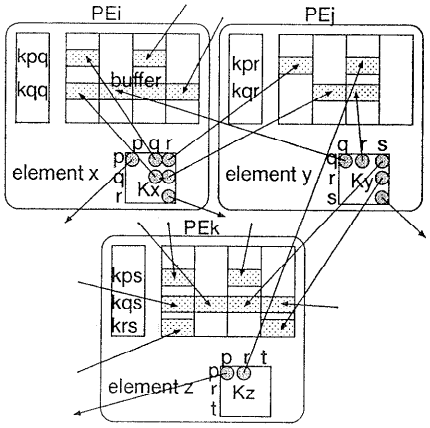


図 5 step 1 の処理
Fig. 5 Processing of step 1.

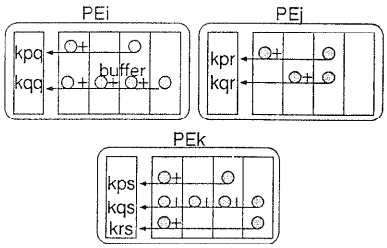


図 6 step 2 の処理
Fig. 6 Processing of step 2.

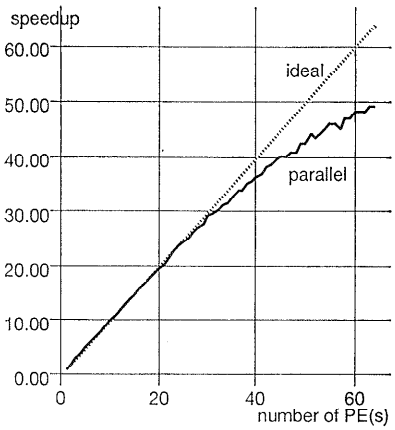


図 7 剛性行列作成処理の速度向上率
Fig. 7 Speedup ratio for generation of the stiffness matrix.

セッサ台数 64 台のときに 1 台のときの約 49 倍の速度向上が得られた.

バッファを用いることにより，**処理 2** で必要なプロセッサ間同期という時間的制約を取り外している．また，剛性行列を各プロセッサに分散格納することにより，加算を並列に処理している.

本方式では，バッファの容量が大きくなりすぎるという問題がある．今回評価したプログラムは，2 次元要素に関するものであるが，実用的な問題では，3 次元要素も扱われる．3 次元要素の場合には，1 つの節点を共有する要素数が増加し，8 節点からなる 6 面体の要素を考えると，8 個のバッファが必要となる．この問題の解決法は，6 章で述べる.

5. 線性求解の並列化

剛性行列が，対称行列で，スパースなバンド行列という特徴を持つため，NONSAP では，剛性行列の格納法としてスカイライン法を用い，線形求解の方法として LU 分解を行っている.

この解法をそのまま Cenju で並列化したところ，2 倍程度の速度向上率しか得られなかった.

LU 分解などの直接法は，どのような問題に対しても線形求解の解を求めることができるため，解法として用いられている.

しかし，一般に剛性行列はバンド幅が狭いので，バンド幅が処理の並列度となる LU 分解では並列処理効果が期待できない．そのため，他の研究[6),9)]でも，反復法を採用しているものが少なくない.

本研究でも，反復法の一種である，SCG 法の並列化を検討することにした.

5.1 SCG 法の概要

SCG (Scaled Conjugate Gradient) 法[3)]は速水らによって提案された，対角項によるスケーリングを施した共役勾配法の一種である．その主な処理は以下に示すようになる.

1. D^{-1} を求める.

$$D^{-1}=\begin{bmatrix} 1/a_{11} & 0 & \cdots & 0 \\ 0 & 1/a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/a_{nn} \end{bmatrix}$$

2. $\boldsymbol{r}_1=\boldsymbol{b}-A\boldsymbol{x}_1$

   $\boldsymbol{p}_1=D^{-1}\boldsymbol{r}_1$

3. i に関して反復計算.

   - $\alpha_i=\dfrac{(\boldsymbol{r}_i\cdot D^{-1}\boldsymbol{r}_i)}{(\boldsymbol{p}_i\cdot A\boldsymbol{p}_i)}$
   - $\boldsymbol{x}_{i+1}=\boldsymbol{x}_i+\alpha_i\boldsymbol{p}_i$
   - $\boldsymbol{r}_{i+1}=\boldsymbol{r}_i-\alpha_i A\boldsymbol{p}_i$
   - $\beta_i=\dfrac{(\boldsymbol{r}_{i+1}\cdot D^{-1}\boldsymbol{r}_{i+1})}{(\boldsymbol{r}_i\cdot D^{-1}\boldsymbol{r}_i)}$

• $\boldsymbol{p}_{i+1}=D^{-1}\boldsymbol{r}_{i+1}+\beta_i\boldsymbol{p}_i$

ただし$A\boldsymbol{p}$, $D^{-1}\boldsymbol{r}$, 内積などは反復当たり1回計算すれば良い.

演算量の大半は$A\boldsymbol{p}$の, 行列×ベクトル→ベクトルの演算である.

5.2 SCG 法の並列化

SCG 法の並列化に当たっては, 従来, 対称行列の上三角形だけを格納していたものを上下三角形両者を格納するようにし, 剛性行列の行ごとに PE に割り当てることにより並列化した. このときの並列化アルゴリズムを図8に示す.

図8の並列処理方式でのプロセッサ間通信は以下の3点で発生する.

1. 行列とベクトルのかけ算 $A\boldsymbol{p}$ を行うのに必要な$\boldsymbol{p}$の転送. 各プロセッサでは$\boldsymbol{p}$のベクトルの要素のうち, 割り当てられた要素に対して新しい値を計算し, その値をすべてのプロセッサに転送しなければならない.
2. ベクトルの内積 $(\boldsymbol{p}\cdot A\boldsymbol{p})$ を演算するときの部分和の転送.
3. ベクトルの内積 $(\boldsymbol{r}\cdot D^{-1}\boldsymbol{r})$ を演算するときの部分和の転送.

5.3 実験結果1

実験には, 2次元要素の剛性行列 (8904×8904, 非零要素 184,776 個) のデータを用いて, 評価を行った. その結果, 10 台程度で速度向上率 (7倍) が飽和した. (図9)

各演算での処理時間を計測した結果, プロセッサ台数が増えると, $\boldsymbol{p}$の通信時間が, $A\boldsymbol{p}$ の計算時間より大きくなることがわかった. (図10)

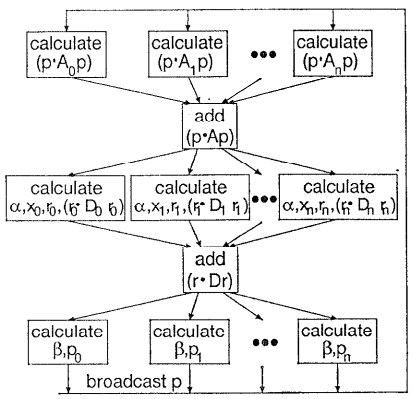


図 8 SCG 法の並列アルゴリズム
Fig. 8 Parallel algorithm for SCG method.

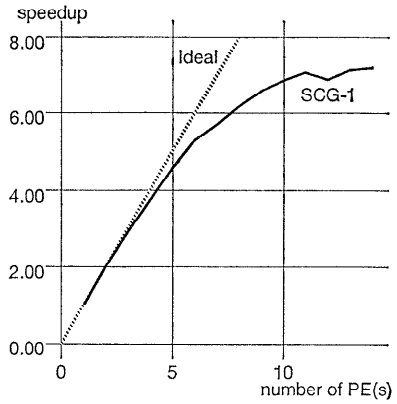


図 9 並列 SCG 法の速度向上率1
Fig. 9 Speedup ratio for parallel SCG method 1.

5.4 通信量の削減

前節の評価では$\boldsymbol{p}$の転送時間が速度向上を妨げることが分かった. ところで, ベクトル$\boldsymbol{p}$の各成分が, $A\boldsymbol{p}=\boldsymbol{s}$ に影響を及ぼす範囲は, 図11に示すように, Aの行列の形状によって定まる.

各 PE への行列Aの行の割り当てを帯状にすると, 行列Aがバンド行列であることから, ベクトル$\boldsymbol{p}$のある成分 $\boldsymbol{p}_i$ が影響を及ぼす$\boldsymbol{s}$の部分を受け持っている

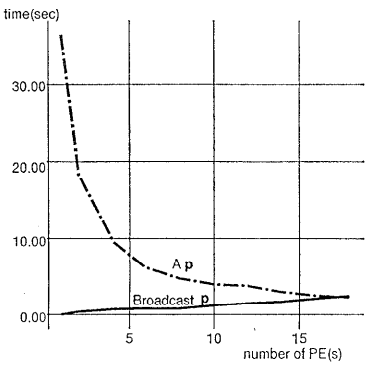


図 10 $\boldsymbol{p}$の通信時間と $A\times\boldsymbol{p}$ の処理時間
Fig. 10 Broadcasting time vs. multiplication time.

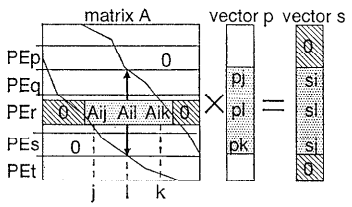


図 11 $\boldsymbol{p}$が影響を及ぼす範囲
Fig. 11 Region on which $\boldsymbol{p}$ effects.

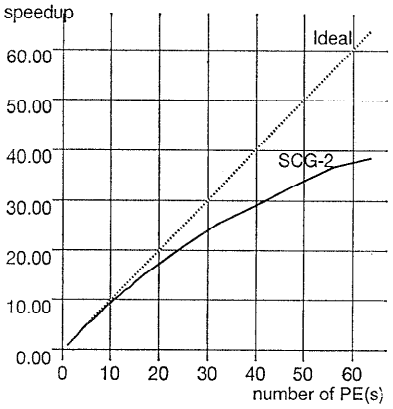


図 12 並列 SCG 法の速度向上率 2
Fig. 12 Speedup for Parallel SCG method 2.

プロセッサ数は少なくなり，$\boldsymbol{p}_i$ の転送時間も短くなる．図 11 の場合，$\boldsymbol{p}_i$ は，PE_q, PE_r, PE_s にだけ転送すればよい．

剛性行列はバンド幅を小さくするように作られているため，影響を及ぼす s の範囲は行列の大きさ N にくらべ十分小さい．

5.5 実験結果 2

$\boldsymbol{p}$ の転送時間を減らすために，影響を与えるプロセッサだけに $\boldsymbol{p}$ の成分を転送するようにした．このときの速度向上率を図 12 に示す．64 台時に 1 台の時の 38 倍の高い速度向上率が得られた．

6. 処理の局所性の活用

4 章の剛性行列の作成部分の並列化で示した並列処理方法では，要素剛性行列の成分が，他のプロセッサに格納された剛性行列の成分の作成に使われる場合，プロセッサ間通信が発生する．あるプロセッサで作成される要素剛性行列の成分から作成される剛性行列の成分が，同じプロセッサに格納される割合が高いほど，プロセッサ間通信は少なくなる．

5 章の SCG 法の並列化では，ベクトル $\boldsymbol{p}$ の転送が処理のオーバヘッドとなっている．節点自由度の番号付けを行って，$\boldsymbol{p}$ の成分を必要とする剛性行列の行を格納しているプロセッサ台数を削減できれば，プロセッサ間通信を削減できる．

このように，要素のプロセッサへの割り当てや，プロセッサが格納する剛性行列の成分の決定などを最適化することにより，処理の局所性が高まり，並列処理のオーバヘッドとなるプロセッサ間通信を削減することができる．

6.1 要素の割り付け・節点自由度の番号付け

4 章の実験では，要素，節点自由度の番号は，データの入力が簡単になるように 1 方向に対して連続に付けられている（図 13 参照）．また，各プロセッサへの要素の割り付けは，1 つのプロセッサが連続する番号の要素を受け持つように割り付けられるように行っている．

本節では，本論文で行った要素のプロセッサへの割り付け方法と，節点自由度の番号付けの方法について説明する．

まず，本研究では，剛性行列の作成時に使用するバッファ容量を削減することが，要素のプロセッサへの割り付けと節点自由度の番号付けを行う動機であった．バッファに格納する要素剛性行列の成分を同じプロセッサが作成した場合には，6.2 節の方法を用いることにより，バッファは不要になる．従って，あるプロセッサで作成した要素剛性行列が，同じプロセッサに格納した剛性行列の成分の作成に使用されるように，処理する要素と格納する節点自由度を決定すれば良い．

本方式は，3 つの処理によって実現されている．

1. 要素のプロセッサへの割り付け（図 14 参照）
   次のような基準で，各プロセッサに割り付ける要素集合を決定する．
   - 各要素集合の要素数が均等になるように選ぶ．
   - その要素を要素集合に加えたとき，集合内の要素間で共有される節点自由度が最も多くなるものを選択する．
   - その要素を要素集合に加えたとき，集合内の要素だけで共有する節点自由度が出現するものを選ぶ．

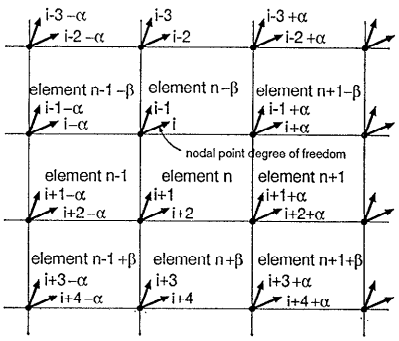


図 13 NONSAP での要素・節点自由度の番号付け
Fig. 13 Numbering elements and degrees of freedom in NONSAP.

- 集合の1つ目の成分となる要素を選ぶときには，その要素から空間的に連続した未割り付けの要素が十分存在するものを選ぶ．

2. プロセッサが格納する剛性行列の行（節点自由度に対応）の決定（**図 15** 参照）

次のような基準で節点自由度の集合を決定する．それに対応する剛性行列の行がプロセッサに格納される．

- 各プロセッサに割り付けられる節点自由度数が均等になるように選ぶ．
- 各プロセッサに割り付けられた要素集合が，各節点自由度と，どの程度関連しているかを評価関数として，各プロセッサに1個ずつ関連性の高い節点自由度を割り付けていく．
- 同じ節点上の自由度は，できるだけ同じプロセッサに割り付ける．
- もし，関連のある節点自由度が1つもないときは，最後に残った節点自由度を割り付ける．

3. 節点自由度の番号付け

各プロセッサに割り付ける節点自由度が，連続するように，節点自由度番号を付け直す．これにより，各プロセッサには剛性行列の行が帯状に格納される．

本方式により，4800 個の2次元要素を 32 台のプロセッサに割り付けたときの結果を**図 16** に示す．同じ文字で表されている要素が同じプロセッサに割り付けられている．

```
N：1つの要素集合あたりの要素数
P：要素集合数(プロセッサ台数)
最初の要素集合の1つ目の要素はユーザが指定．

for (p=0; p < P; p ++) {
  do {
    次の基準で要素を1個選び，要素集合pに加える．
    (1, 2, 3は選択条件の優先順位)
      1. その要素を要素集合pに加えた場合，集合内の要素間で共有
         される節点自由度が最も多くなる要素．
      2. その要素を要素集合pに加えた場合，集合内の要素だけで
         共有する節点自由度が多く生じる要素．
      3. その要素が共有する節点自由度の中で，すでに他の要素集合
         の成分となっている要素が共有する節点自由度が多い要素．
  } while (要素集合pの要素数 < N)

  if (p != P－1) {
    上の選択基準で要素を1つ選ぶ．
    if (その要素から空間的に連続した未割り付けの要素の数 ≧ N) {
      その要素を次の要素集合の1つ目の要素とする．
    } else {
      未割り付けの要素の中で，その要素から空間的に連続した
      未割り付けの要素数がN以上または最大の要素を次の要素集合
      p+1の1つ目の要素とする．
    }
  }
}
```

図 14 プロセッサへの要素の割り付けアルゴリズム
Fig. 14 Algorithm for assignment of elements to processors.

```
M：1つのプロセッサに格納する剛性行列の行数(節点自由度数)
P：要素集合数(プロセッサ台数)

各節点自由度の各要素集合との関連性を評価する．
これは，節点自由度がどの要素集合に含まれている要素によって
共有されているかから求める．

for (m = 0; m < M; m ++) {
  for (p = 0; p < P; p ++) {
    if (next[p] > 0) {
      next[p] －－;
    } else {
      if (要素集合pに関連性のある節点自由度が存在する) {
        その中で最大の関連性を持つ節点自由度を要素集合pを
        処理するプロセッサに格納する．
        その節点自由度と同じ節点にある節点自由度(f個)は
        すべてこのプロセッサに格納する．
        next[p] = f;
      }
    }
  }
}

for (p = 0; p < P; p ++) {
  if (要素集合pを処理するプロセッサに格納した節点自由度数 < M) {
    未割り付けの節点自由度をそのプロセッサに格納する．
  }
}
```

図 15 プロセッサに格納する剛性行列の行の割り付けアルゴリズム
Fig. 15 Algorithm for assignment of rows of a stiffness matrix to processors.

6.2 バッファ容量の削減

本方式を用いて，あるプロセッサが**処理 1** で作成した要素剛性行列の成分を格納するバッファが，そのプロセッサ自身に割り付けられている場合が多く起こるように，節点自由度の番号付けを行い，以下のように処理することで，バッファの容量を削減することがで

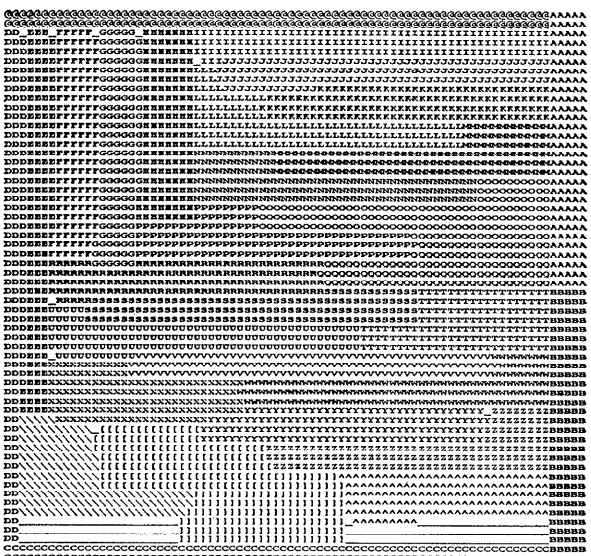


図 16 要素のプロセッサへの割り付け例
Fig. 16 Example of assignment of elements to processors.

きる.

1. 作成した要素剛性行列の成分を格納するバッファが，同じプロセッサに割り付けられている（すなわち，要素剛性行列を足し込んで作成する剛性行列の成分を同じプロセッサに格納する）場合は，この要素剛性行列の成分を直接剛性行列の成分に足し込む．この場合バッファは不要となる．また，バッファに書き込むときに行っていたプロセッサ間通信も不要となる．(**図 17** 参照)
2. それ以外の場合は，従来どおりプロセッサ間通信を行って，他のプロセッサにあるバッファに書き込む．そして，**step 2** でバッファの中の値を剛性行列の成分に足し込む．

2次元の要素 1200 個（30×40, 2542 節点自由度）と3次元の要素 1000 個（10×10×10, 3993 節点自由度），2次元の要素 4800 個（60×80, 9882 節点自由度）と3次元要素 3000 個（10×15×20, 11088 節点自由度）に対して本方式を適応した場合に，同じプロセッサに格納される要素剛性行列の成分の割合とプロセッサ台数の関係を，それぞれ，**図 18**, **図 19** に示す．

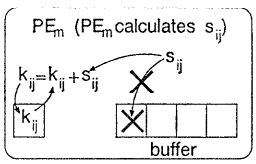


図 17 バッファが削減できる場合の処理
Fig. 17 Processing in case of reducing the size of buffers.

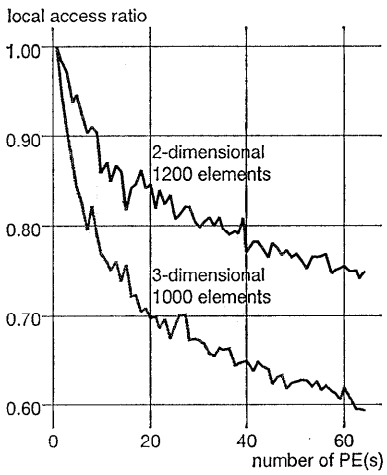


図 18 同じプロセッサへの書き込みの割合1
Fig. 18 Local access ratio 1.

3次元要素は，2次元要素に比べ，1つの自由度を共有している要素の数が多いため，同じプロセッサへの書き込みの割合が低くなっている．また，同じプロセッサへの書き込みの割合は，1つのプロセッサに割り付けられる要素数で決まることがわかる．従って，大規模な問題でも，プロセッサ台数を増やせば，処理の局所性は保たれることになる．

6.3 剛性行列作成における効果

4章で実験に使ったデータ（要素数 1266 個，節点自由度数 2743 個）を用い，プロセッサへの要素の割り付けと節点自由度の番号付けを行った場合の，剛性行列の作成処理の速度向上率を**図 20** に示す．64 台の

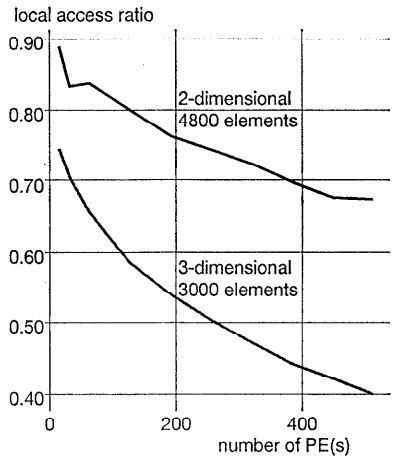


図 19 同じプロセッサへの書き込みの割合2
Fig. 19 Local access ratio 2.

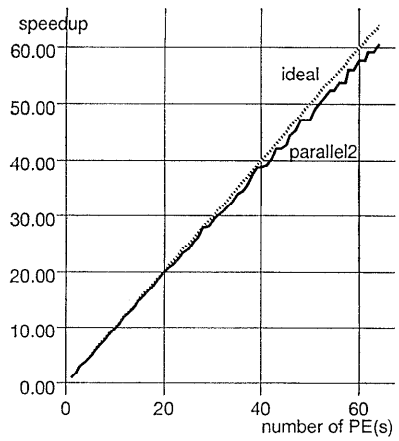


図 20 節点自由度の番号付けを行った場合の剛性行列作成処理の速度向上率
Fig. 20 Speedup ratio for generation of stiffness matrix using renumbering method.

表 1 並列 SCG 法で起こるプロセッサ間通信
Table 1 Interprocessor communications in palallel SCG method.

プロセッサ台数	通信量（自由度数）			通信回数 (PE→PE)		
	処理前	バンド化	本方式	処理前	バンド化	本方式
8	2302	1946	2132	14	14	46
16	4926	4206	3810	30	30	124
24	7750	6318	5004	46	46	232
32	10194	8568	6330	62	62	296
40	12830	10682	6234	46	78	406
48	15422	12906	6232	94	94	448
56	18066	14904	7346	110	124	558
64	20258	16590	7460	250	160	630

プロセッサで約 60 倍の高い速度向上率が得られた．また，並列処理のオーバヘッドとなるプロセッサ間通信が削減されるため，PE 台数が増加しても速度向上率が鈍化していない．

このデータでは，プロセッサ台数が 64 台のとき，82% が同じプロセッサ内の処理（前節の 1 の場合）となっており，82% のバッファが削減できたことになる．

また，未処理の場合には，1.8% が同じプロセッサへの書き込みとなっており，本方式により，プロセッサ間通信が 80% 削減されたことになる．

6.4 並列 SCG 法における効果

並列 SCG 法では，ベクトル $\boldsymbol{p}$ の成分 $\boldsymbol{p}_i$ を必要とするプロセッサ台数が少ないほど，プロセッサ間通信量は少なくなる．一般に，剛性行列のバンド幅を小さくすると，$\boldsymbol{p}_i$ を転送するプロセッサ台数が少なくなることが予想される．

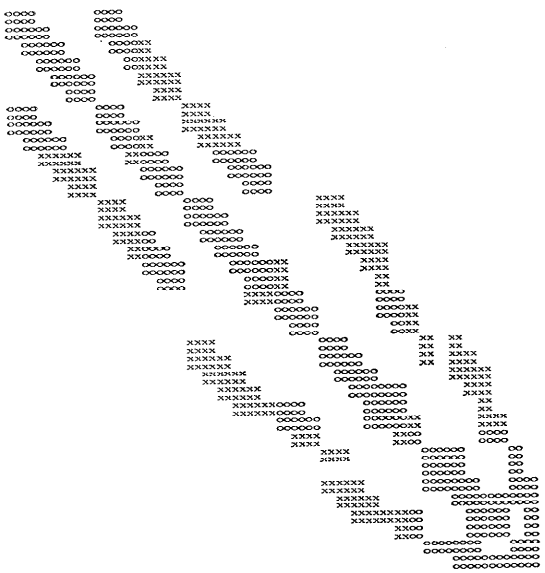


図 21 Gibbs のアルゴリズムによる節点自由度の番号付け
Fig. 21 Renumbering by Gibbs's Algorithm.

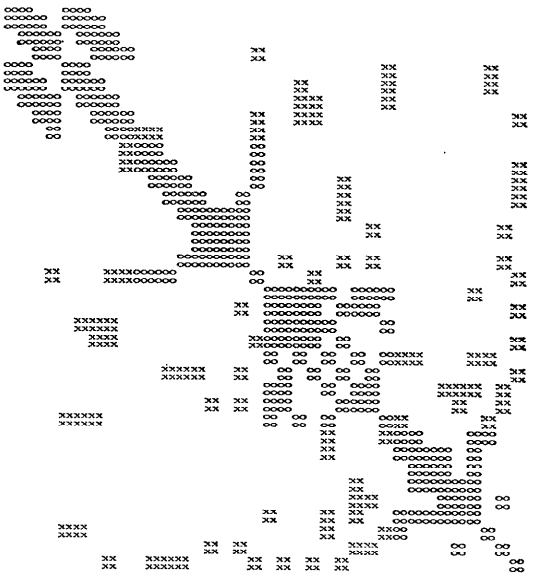


図 22 本方式の節点自由度の番号付け
Fig. 22 Renumbering by our method.

文献10)の行列のバンド幅を減少させるアルゴリズムを用い，バンド幅を減少させた場合の $\boldsymbol{p}$ の成分の転送量と，本方式の節点自由度の番号付けを行った場合の $\boldsymbol{p}$ の成分の転送量とを比較した．(**表 1** 参照)

本方式を用いた方が，バンド幅を減少する操作を行った場合よりも，$\boldsymbol{p}_i$ の転送量は少なくなっている．これは，バンド幅を減少するという制限によって，同じプロセッサの $\boldsymbol{p}_i$ を使用できる成分（O）の数が減り，他のプロセッサの $\boldsymbol{p}_i$ が必要となる成分（X）が増加するためである．他のプロセッサの $\boldsymbol{p}_i$ を必要とする成分には，バンド幅を減少するという制約があるため，対角成分に近い部分に集まる．従って，$\boldsymbol{p}_i$ の転送元のプロセッサ数は，本方式に比べ少なくなっている．(**図 21**, **図 22** 参照)

以上述べたように，本方式により並列 SCG 法でも $\boldsymbol{p}$ の転送に必要なプロセッサ間通信が削減できる．

7. おわりに

並列シミュレーションマシン Cenju の応用の1つとして，有限要素法による非線形変形解析のプログラムを並列化し，その評価結果について述べた．

剛性行列作成部分に関しては，バッファを用いて**処理2**（部分剛性行列の足し込み）を並列に行う方法により，64 台で約 49 倍の高い速度向上が得られた．

線形求解の部分は，LU 分解を直接並列化した場合は，余り高い速度向上が得られなかったのに対して，共役勾配法の一種である SCG 法を用いると，プロセッサ間通信を削減した結果，64 台で1台のときの 38 倍の速度向上が得られた．

要素のプロセッサへの割り付けと節点自由度の番号付けによるプロセッサ間通信の削減法を提案した．その結果，剛性行列の作成で，64 台で 60 倍もの高い速度向上率を得ることができ，この方法の有効性を示した．また，この方法による節点自由度の番号付けは，並列 SCG 法のプロセッサ間通信の削減にも寄与することがわかった．

本研究の今後の課題としては，

- SCG 法の収束性の検討
- 並列 SCG 法での，節点自由度の番号付けの効果の評価
- 有限要素法の処理全体での並列処理効果の評価
- 領域分割法との性能の比較
- VR 3000 を要素プロセッサにした Cenju 2[11]での本手法の評価

などがあげられる．

並列計算機の大規模化やプロセッサの高速化に伴い，ますますプロセッサ間通信のオーバヘッドが大きく見えてくるであろう．このような場合に，並列処理を効率良く行うためには，本論文で提案したような，処理の局所性を活かしてプロセッサ間通信を削減する並列処理手法が必要となってくる．

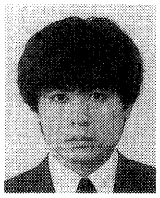
なお，本研究の一部は，科学技術庁官民特定共同研究の一環として行った．

謝辞 本研究の機会を与えて頂き，また有益な示唆を頂いた日本電気(株)研究開発グループ 石黒支配人，同 C & C システム研究所 山本所長，SCG 法の並列化に貴重なコメントを頂きました同 C & C 情報基礎研 速水課長，Cenju のハードウェア製作に尽力された同 C & C システム研究所 三野輪一氏に深謝いたします．

参考文献

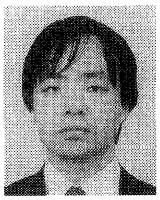
1) 大竹，福島，安永，山本：大型衛星フェアリングの分離挙動の数値シミュレーションについて，日本航空宇宙学会第 31 回構造強度講演集，pp. 304-307 (1989)．
2) Bathe, K., Wilson, E. L. and Iding, R. H.: NONSAP, Structural Engineering Laboratory, University of California Berkley, UCSESM 74-3 (1974)．
3) 速水，原田：対角項スケーリングを施した共役勾配法のベクトル計算機における有効性について，情報処理学会論文誌，Vol. 30, No. 11, pp. 1364-1375 (1989)．
4) 中田ほか：並列回路シミュレーションマシン Cenju，情報処理，Vol. 31, No. 5, pp. 593-601 (1990)．
5) Storaasli, O. and Ransom, J.: Structural Dynamic Analysis on a Parallel Computer: The Finite Element Machine, *Computers & Structures*, Vol. 26, No. 4, pp. 551-559 (1987)．
6) 佐藤，上村，星野：並列計算機 PAX による2次元弾性問題の有限要素解析，情報処理学会論文誌，Vol. 26, No. 4, pp. 584-590 (1985)．
7) 吉岡，矢川，吉村，曽根田：大規模・超高速計算力学のためのネットワーク・コンピューティング手法の開発，日本機械学会論文集（A編），Vol. 57, No. 541, pp. 22-30 (1991)．
8) Zois, D.: Parallel Processing Techniques for FE Analysis: Stiffness, Loads and Stresses Evaluation, *Computers & Structures*, Vol. 28, No. 2, pp. 247-260 (1988)．
9) 土肥，小山：有限要素法の並列処理手法，電子通信学会論文誌，Vol. J 65-D, No. 4, pp. 464-470 (1982)．
10) Gibbs, N. E., Poole, W. G. and Stockmeyer, P. K.: An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix, *SIAM J. Num. Anal.*, Vol. 13, No. 2, pp. 236-250 (1976)．
11) 松下，山内，中田，小池：並列マシン Cenju 2 のアーキテクチャ，情報処理学会研究報告，Vol. 92, No. 64, pp. 17-23 (1992)．

（平成4年 9 月 14 日受付）
（平成4年12月 10 日採録）



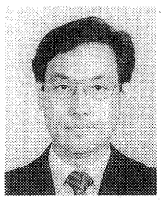
加納　健（正会員）

1962 年生．1987 年京都大学工学部情報工学科卒業．1989 年京都大学大学院工学研究科修士課程情報工学専攻修了．同年日本電気(株)入社．現在，C&C システム研究所コンピュータシステム研究部にて，並列計算機システム，並列処理の研究に従事．



中田登志之（正会員）

昭和 32 年生．昭和 57 年京都大学大学院工学研究科修士課程修了．昭和 60 年同大学院博士後期課程単位取得退学．同年日本電気(株)入社．京都大学工学博士．現在同社 C&C システム研究所コンピュータシステム研究部研究課長．並列計算機システムの研究に従事．昭和 61 年度本学会論文賞受賞．30 周年記念論文入賞．電子情報通信学会会員．



奥村　秀人（正会員）

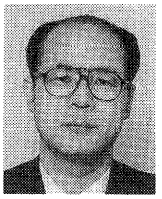
昭和 26 年生．昭和 51 年電気通信大学大学院修士課程修了．同年東京大学生産技術研究所入所．昭和 59 年航空宇宙技術研究所入所，現在に至る．工学博士．日本航空宇宙学会，日本機械学会，日本複合材料学会各会員．

大竹　邦彦

1942 年生．1966 年東京大学航空学科卒業．同年航空宇宙技術研究所入所．1979 年テキサス大学 Ph. D. 航空機構造力学，計算力学の研究に従事．日本航空宇宙学会，機械学会，材料学会各会員．

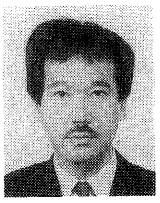
中村　孝（正会員）

昭和 24 年生．昭和 52 年東京電機大学通信工学科卒業．昭和 44 年航空宇宙技術研究所入所．現在 CFD および並列計算機システムの研究に従事．



福田　正大

1947 年生．1972 年京都大学理学部卒業．同年航空宇宙技術研究所入所．線形計算法，内部流数値シミュレーション，並列計算法の研究に従事．機械学会，応用数理学各会員．



小池　誠彦（正会員）

昭和 22 年生．昭和 45 年東京大学工学部電気工学科卒業．昭和 47 年同大学院修士課程修了．同年日本電気(株)に入社．以来並列計算機システム，論理シミュレーションエンジン (HAL)，並列回路シミュレーションマシン (Cenju) などの研究開発に従事．最近では，並列計算機のアーキテクチャ，CAD マシン，AI システム，ニューラルネットワークなどの研究に興味を持つ．現在，同社 C&C システム研究所コンピュータシステム研究部長．工学博士．著書「CAD マシン」(電子情報通信学会編，オーム社)．電子情報通信学会会員．昭和 59 年度情報処理学会論文賞受賞．平成 2 年本学会創立30 周年記念論文入選．