構造工学論文集 Vol.37 A (1991年3月) 土木学会

# 共役勾配法のスーパーコンピュータ適合アルゴリズム

# A SOLUTION ALGORITHM OF THE CONJUGATE GRADIENT METHOD SUITABLE TO SUPERCOMPUTER ARCHITECTURES

吉田 裕* 中川 昌弥** 田中 知足***
Yutaka YOSHIDA, Masaya NAKAGAW, Tomoyuki TANAKA

Because of several practical applications of the finite element analysis being computationally very demanding, supercomputer performance is desirable. The storage requirements and performance consequences of different implementations of the conjugate gradient methods for the solution of sparse systems arising from finite element structural analysis are reviewed.

A stiffness matrix storage strategy suitable to pipelining of instruction and arithmetic units of supercomputer architectures is presented. The results from several simulations show that the conjugate gradient method is competitive with direct methods.

## 1. はじめに

ここ数年の間で計算機技術はめざましい発達を遂げ、それにともない数値解析に基づいて実現象をシミュレートしようとする計算力学の分野も急速に進歩している。コンピュータの性能の向上により、大次元の非線形問題を対象とすることも稀ではなく、このような場合係数マトリックスを書き換えながら大次元の連立一次方程式を繰り返し解くということが必要となってくる。このため、大次元の問題を少ない容量でいかに解くかといった連立一次方程式解法の効率化が重要な課題である。

連立一次方程式は大別して、消去法の系統の直接法と、共役勾配法(以下ＣＧ法)に代表される反復法とに分けることができる。このうちＣＧ法は、マトリックスとベクトルの乗算だけで解き進めることができるので、有限要素法や差分法で対象とする疎な大次元行列に対して、記憶容量の面で圧倒的に有利であるが、対象とする問題の違いによって収束性が大きく異なるといった問題点を有している。

ＣＧ法の計算時間の効率化を計る１つの方策として、近年普及しつつあるスーパーコンピュータとの適合性を考慮することが重要である。[1)-4)] スーパーコンピュータの性能は、演算が連続的、並列的に行われて初めて発揮されるものであるから、その性能を生かすためにはベクトル化率を高くし、ベクトル長を長くする、などプログラムやアルゴリズムに相当の配慮を必要とする。

* 工博 東京工業大学教授 工学部土木工学科 (〒152 東京都目黒区大岡山2-12-1)
** 工修 ＪＲ東日本 (研究当時 東京工業大学大学院 修士課程学生)
*** 東京工業大学大学院 修士課程学生 (〒152 東京都目黒区大岡山2-12-1)

ＣＧ法のアルゴリズムのうち最も計算量の多い部分は、係数マトリックス**A**と修正ベクトル**p**の積**Ap**であり、この計算部分の効率化が解法全体の効率化に大きく影響する。積**Ap**はマトリックスとベクトルの乗算であり、データに依存関係がなくベクトル化に適しているように思えるが、通常とられるアルゴリズムではマトリックスの対称性やスパース性を考慮するために、ベクトル長を長くとりアルゴリズムをベクトル計算に適合させることと直結するわけではない。

ＣＧ法の記憶容量の面の有効性を犠牲にすることなく収束性を改善する方法として、前処理として不完全なコレスキー分解を介するＩＣＣＧ法がよく研究されており[5)-9)]、収束性の大幅な改善につながるのが普通である。しかし、ＩＣＣＧ法では反復計算ごとに前進代入および後退代入計算を要し、この前進代入および後退代入計算部は、データに依存関係があるためにベクトル計算機による効率化の方策を考えることは困難である。

本研究は、有限要素法で対象とすることになる疎な大次元連立一次方程式の効率的な解法のアルゴリズムを構築することを目的として、構造問題の有限要素方程式を対象として具体的な数値実験を行い、ＣＧ法を基礎とする種々の連立一次方程式のアルゴリズムの収束性、要する計算時間などを検討し、これを踏まえた上でスーパーコンピュータとの適合性を考慮した効率化アルゴリズムを具体化し、その有効性を確認したので、その内容を報告するものである。

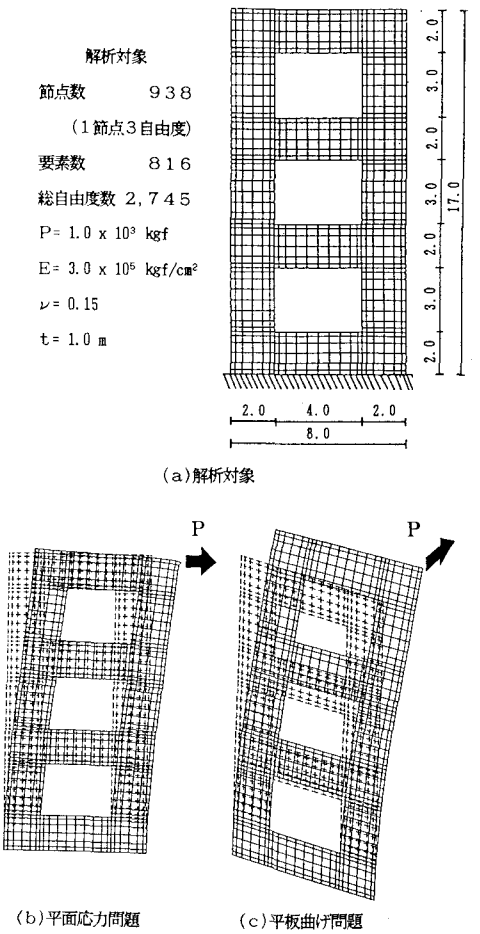
## ２．ＣＧ法について

ＣＧ法は、各反復計算ごとに残差ベクトルが直交するように解を修正しながら解き進めて行く解法である。解くべき連立一次方程式を$\mathbf{A}\mathbf{x}=\mathbf{b}$とすると、ＣＧ法のアルゴリズムは以下のように表される。

$$
\left.\begin{aligned}
&1)\ \ \mathbf{r}_0=\mathbf{b}-\mathbf{A}\mathbf{x}_0\ ,\ \ \mathbf{p}_0=\mathbf{r}_0\\
&\quad\ \text{ただし}\mathbf{x}_0\text{は任意の初期ベクトル}\\
&2)\ \ \alpha_k=(\mathbf{p}_k,\mathbf{r}_k)/(\mathbf{p}_k,\mathbf{A}\mathbf{p}_k)\\
&3)\ \ \mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k\mathbf{p}_k\\
&4)\ \ \mathbf{r}_{k+1}=\mathbf{r}_k-\alpha_k\mathbf{A}\mathbf{p}_k\\
&5)\ \ \beta_k=-(\mathbf{r}_{k+1},\mathbf{A}\mathbf{p}_k)/(\mathbf{p}_k,\mathbf{A}\mathbf{p}_k)\\
&6)\ \ \mathbf{p}_{k+1}=\mathbf{r}_{k+1}+\beta_k\mathbf{p}_k\\
&7)\ \ 2)\text{へ}
\end{aligned}\right\}\ (1)
$$

ここに、(,) はベクトルの内積である。このアルゴリズムからもわかるように、係数マトリックスとベクトルの乗算だけで解き進めて行くことができるので、係数マトリックスの非零要素だけを記憶しておけばよい。また、アルゴリズム中、最も演算量の多いのは積**Ap**の計算であるが、反復計算１回につき2)においてもとめた値を4),5)で用いる。

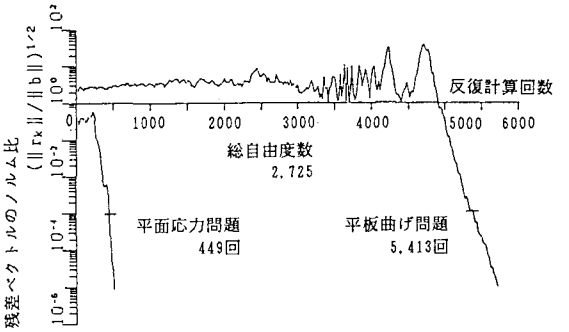
理論的には有限回の反復計算を行なえば解が得られるはずであるが、桁落ち等が大きく影響し、決められた反復計算回数内では収束しないことも多い。また、その収束性は係数マトリックスの性質に大きく依存し、対象とする問題によって大きな違いが生じることも多い。ここで解析の対象とする、図－１に示すような平面応力問題と平板曲げ問題についてみても、それぞれの問題に対して係数マトリックスを対角要素




図－１　解析対象問題

でスケーリングしてからＣＧ法を適用した場合の収束状況を図－２に示すが、同じ自由度数の問題であるにも関わらず収束状況は大きく違っている。このことは解析に要する計算時間にも反映され、同じ問題をスカイライン法で解いた場合の計算時間と比較すると（表－１）、対象とする問題によって大きな差がありＣＧ法の適用には十分な注意が必要なことがわかる。しかし、記憶容量の面では圧倒的に有利であることも示されている。以下、ここで挙げた平面応力問題と平板曲げ問題を収束の良し悪しを代表する問題として取り上げる。

なお、以下の計算において収束判定は
$(\|r_k\|/\|b\|)^{1/2} < 10^{-3}$としたが、収束状況がどのように変化するか調べるため計算は$10^{-5}$まで続けた。



図－２　ＣＧ法の収束状況

表－１　ＣＧ法とスカイライン法の比較
（ETA10によるCPU-time）

| 解法の種類 | 平面応力問題 | 平板曲げ問題 | 要する容量 |
|---|---|---|---|
| ＣＧ法 | 53.4 (s) | 475.8 (s) | 447 KB |
| スカイライン法 | 13.1 (s) | 10.4 (s) | 1.44 MB |

## 3．スケーリングの方法によるＣＧ法の収束状況の違い

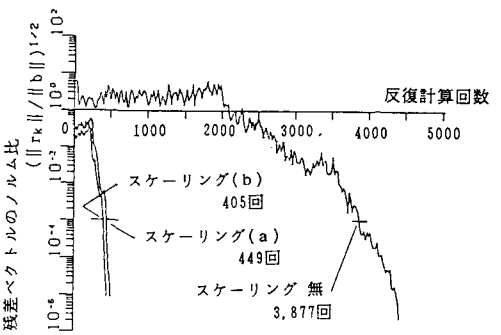
### (1)スケーリングの方法について

ＣＧ法は、係数マトリックスをスケーリングすることで数値誤差の影響が少なくなり、その収束性が改善されることが知られている。[10] ここでは、係数マトリックスをスケーリングするに際して２通りの方法を試みた。適用した方法は係数マトリックスの対称性をくずさないような以下に示すやり方である。

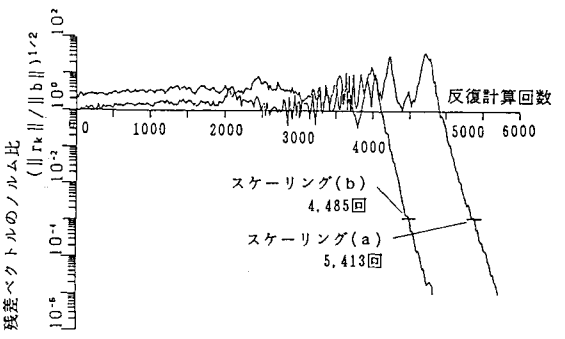
(a)　係数マトリックスの対角要素を１とする通常のスケーリング。与えられた連立一次方程式を$\mathbf{A}\,\mathbf{x} = \mathbf{b}$、係数マトリックス$\mathbf{A}$の対角要素だけを残したマトリックスを$\mathbf{D}$として下記のように表される。

(b)　係数マトリックスの対角部分に相当する、節点の自由度を単位とするブロックでスケーリングする方法。(a)と同様に、対角部分のブロックだけを残したマトリックスを$\mathbf{D}$として、次のように表される。

$$\left.\begin{aligned} \tilde{\mathbf{A}} &= \mathbf{D}^{-1/2}\,\mathbf{A}\,\mathbf{D}^{-1/2} \\ \tilde{\mathbf{x}} &= \mathbf{D}^{1/2}\,\mathbf{x} \\ \tilde{\mathbf{b}} &= \mathbf{D}^{-1/2}\,\mathbf{b} \end{aligned}\right\} \quad (2)$$



図－３　スケーリングによる収束状況の違い（平面応力問題）



図－４　スケーリングによる収束状況の違い（平板曲げ問題）

### (2)スケーリングの方法が収束状況に及ぼす影響

前述の平面応力問題と平板曲げ問題に対し、スケーリングをしない場合、(a)の方法、(b)の方法のそれぞれによる収束状況の違いを図－３、４に示す。なお、平板曲げ問題についてはスケーリングを施さなかった場合には１万回の反復計算をしても収束しなかった。スケーリングをするだけで

収束性が大幅に改善されているのがわかる。また、スケーリングの方法を変えるだけでも収束性に改善がみられている。これらのことは、ＣＧ法の収束性が数値誤差に大きく影響されることを端的に示している。

## 4．不完全コレスキー分解を施すＣＧ法(ＩＣＣＧ法)について

### (1)ＩＣＣＧ法のアルゴリズム

ＩＣＣＧ法は、前処理として元の係数マトリックスを不完全コレスキー分解したマトリックスを用い、係数マトリックスの性質を改善することにより、ＣＧ法の収束を速めようとするものである。アルゴリズムは不完全コレスキー分解

$$\mathbf{A} = \tilde{\mathbf{U}}^{T}\tilde{\mathbf{D}}\tilde{\mathbf{U}} + \mathbf{R} \qquad (3)$$

で表されるマトリックス$\tilde{\mathbf{U}}^{T}\tilde{\mathbf{D}}\tilde{\mathbf{U}}$を用いて

$$\left.\begin{aligned} \tilde{\mathbf{A}} &= (\tilde{\mathbf{U}}^{T}\tilde{\mathbf{D}}^{1/2})^{-1}\,\mathbf{A}\,(\tilde{\mathbf{D}}^{1/2}\tilde{\mathbf{U}})^{-1} \\ \tilde{\mathbf{x}} &= \tilde{\mathbf{D}}^{1/2}\,\tilde{\mathbf{U}}\,\mathbf{x} \\ \tilde{\mathbf{b}} &= (\tilde{\mathbf{U}}^{T}\,\tilde{\mathbf{D}}^{1/2})^{-1}\,\mathbf{b} \end{aligned}\right\} \qquad (4)$$

としたときの$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$にＣＧ法を適用し、～のついていない元のマトリックス$\mathbf{A}$とベクトル$\mathbf{x}$で表すと、前記ＣＧ法のアルゴリズム中 1),5),6)が以下のように書き換えられる。

$$\left.\begin{aligned} &1')\ \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0\ ,\ \mathbf{p}_0 = (\tilde{\mathbf{U}}^{T}\tilde{\mathbf{D}}\tilde{\mathbf{U}})^{-1}\mathbf{r}_0 \\ &5')\ \beta_k = -\,(\,(\tilde{\mathbf{U}}^{T}\tilde{\mathbf{D}}\tilde{\mathbf{U}})^{-1}\mathbf{r}_{k+1}\ ,\ \mathbf{A}\mathbf{p}_k\,)\ /\ (\,\mathbf{p}_k\ ,\ \mathbf{A}\mathbf{p}_k\,) \\ &6')\ \mathbf{p}_{k+1} = (\tilde{\mathbf{U}}^{T}\tilde{\mathbf{D}}\tilde{\mathbf{U}})^{-1}\mathbf{r}_{k+1} + \beta_k\mathbf{p}_k \end{aligned}\right\} \qquad (5)$$

ここで、収束性が改善される代わりに反復計算１回につき、１度の後退代入計算が必要となる。また、３．で示したスケーリングの方法は$\tilde{\mathbf{U}}^{T}\tilde{\mathbf{D}}\tilde{\mathbf{U}}$のかわりにスケーリングに用いる$\mathbf{D}$のみを残したものと考えればＩＣＣＧ法のアルゴリズムを採用することができる。

### (2) 不完全コレスキー分解の方法について

ここでは(a)の方法でスケーリングした係数マトリックスに対し、不完全コレスキー分解として以下の２通りの方法を用いた。

(a) 元の係数マトリックスに対しfill-inを無視してコレスキー分解を行い、非対角要素は計算することなく対角要素のみ作成する方法で以下の手順により計算される。

$$\left.\begin{aligned} \tilde{u}_{11} &= a_{11} \\ \tilde{u}_{ij} &= a_{ij} \\ \tilde{d}_{ii} &= 1\ /\ \tilde{u}_{ii} \\ \tilde{u}_{ii} &= (\mathrm{const}) \times a_{ii} - \sum_{k=1}^{i-1} \tilde{u}_{ki}\,\tilde{d}_{kk}\,\tilde{u}_{ki} \end{aligned}\right\} \qquad (6)$$

(b) 節点の自由度のブロックを単位として、(a)と同様に対角ブロックを作成する方法。

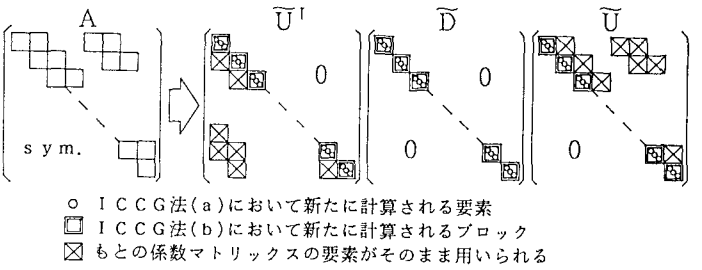
$$\left.\begin{aligned} \tilde{\mathbf{u}}_{11} &= \mathbf{a}_{11} \\ \tilde{\mathbf{u}}_{ij} &= \mathbf{a}_{ij} \\ \tilde{\mathbf{d}}_{ii} &= \tilde{\mathbf{u}}_{ii}^{-1} \\ \tilde{\mathbf{u}}_{ii} &= (\mathrm{const}) \times \mathbf{a}_{ii} - \sum_{k=1}^{i-1} \tilde{\mathbf{u}}^{T}_{ki}\,\tilde{\mathbf{d}}_{kk}\,\tilde{\mathbf{u}}_{ki} \end{aligned}\right\} \qquad (7)$$

このとき、新たに計算される対角要素$\tilde{d}_{ii}$が負になってしまう場合が出て来るが、これを防ぐために元の対角要素 $a_{ii}$ に乗ずるのがconst（加速係数と呼ばれている）である。

(b)の方法では、計算される対角部分のブロックの逆行列も求める必要がある。ここでのconstは、対角部

分に相当するブロックの全ての要素に乗じている。

このように分解することで、最終的には図－５に示すようなマトリックス $\tilde{U}^T\tilde{D}\tilde{U}$ を得ることができる。ここで、書き換えられる対角要素に相当する容量が必要になる。




○ ＩＣＣＧ法(ａ)において新たに計算される要素
□ ＩＣＣＧ法(ｂ)において新たに計算されるブロック
☒ もとの係数マトリックスの要素がそのまま用いられる

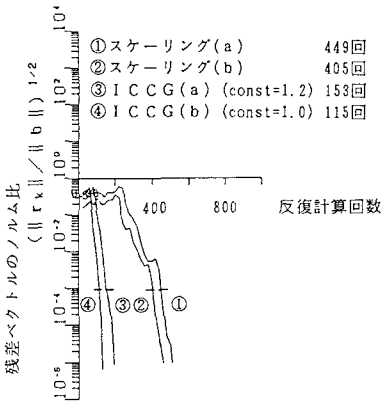
図－５　不完全コレスキー分解の仕方

(３)不完全コレスキー分解の方法が収束状況に及ぼす影響

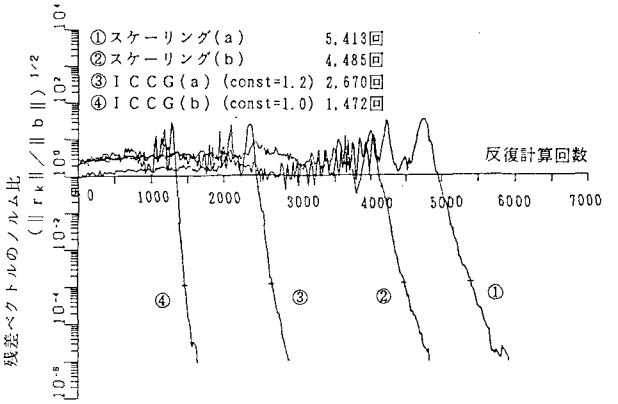
ＩＣＣＧ法での平面応力問題、平板曲げ問題それぞれの収束状況を、３．(a)および(b)の方法でスケーリングしたＣＧ法の収束状況と併せて図－６、７に示す。平面応力問題、平板曲げ問題ともにスケーリングだけの方法と比較しても反復計算回数が大幅に減少し、前処理の効果が大きいことを示している。しかし、収束に要する反復計算回数は、平面応力問題で115回であるのに対して、平板曲げ問題では依然として1472回を要している。なお、不完全コレスキー分解のアルゴリズムの中で、constの値は平面応力問題では(a)1.2、(b)1.0、平板曲げ問題では(a)1.7、(b)1.2とした。

(４)constの値について

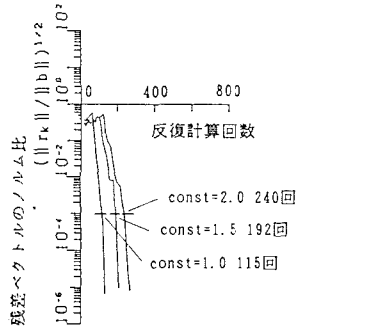
ＩＣＣＧ法では対角部分に乗ずるconstの値の決め方が問題となるが、ここで(b)の方法においてconstの値を、新たに計算される対角ブロックの対角要素が負にならないような１に最も近い値、1.5、2.0、に変化させて解析を行ない収束状況の違いを図－８、９に示す。constを大きくとるほど収束性が悪くなるのがわかる。



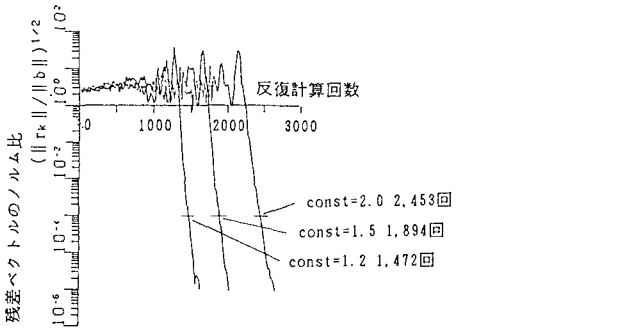

図－６　ＩＣＣＧ法の収束性（平面応力問題）




図－７　ＩＣＣＧ法の収束性（平板曲げ問題）




図－８　constによる収束性の違い（平面応力問題）




図－９　constによる収束性の違い（平板曲げ問題）

平面応力問題ではconstを1.0にした場合でも負の対角要素は出てこなかったが、平板曲げ問題では負の対角要素がでてきて、かえって係数マトリックスの性質を悪くしてしまい、収束して行く様子はまったく見られなかった。このことからconstの値は、小さいものから探すとなると手間はかかるが、対角要素が負にならない程度でなるべく1.0に近い値がよいと考えられる。

## 5．計算時間について

(b)の方法によるＩＣＣＧ法と、(a)の方法でスケーリングしたＣＧ法でそれぞれの解析に要した計算時間を表－２にまとめて示す。ここに挙げた計算時間は、次節から示すようなベクトル演算との適合性は考えておらず、７．で示す通常考えられるようなアルゴリズムに基づいたものである。ＩＣＣＧ法では反復計算回数が大幅に減少しているにも関わらず、反復計算中の前進代入および後退代入部分に時間を費やしてしまい、結果的には反復回数の減少ほどの効果はみられていないことがわかる。

表－２　解析に要した計算時間

(ETA10によるCPU-time)

| | 解法の種類 | 前処理過程 (s) | 反復計算過程 回数 | 反復計算過程 1回当り(s) | 反復計算過程 全過程(s) | 全計算過程 (s) |
|---|---|---|---|---|---|---|
| 平面応力 | スケーリング(a) | —— | 449 回 | 0.086 | 38.57 | 53.4 |
| | ＩＣＣＧ法 (b) | 0.11 | 115 回 | 0.187 | 21.49 | 36.6 |
| 平板曲げ | スケーリング(a) | —— | 5,413 回 | 0.086 | 464.95 | 475.8 |
| | ＩＣＣＧ法 (b) | 0.11 | 1,472 回 | 0.187 | 276.07 | 287.2 |

## 6．スーパーコンピュータによる効率化の可能性について

ＣＧ法の計算時間を効率化する１つの方策として近年普及しつつあるスーパーコンピュータとの適合性を考慮することが重要である。スーパーコンピュータによるベクトル計算で計算時間の効率化を目指す場合、ベクトル化率を高くし、ベクトル長をできるだけ長くとるなどプログラムやアルゴリズムに相当な配慮が必要である。

表－３は前述の平面応力問題に対角要素によるスケーリングを施すＣＧ法を適用し、東京工業大学総合情報処理センターのスーパーコンピュータ ETA-10 E とスカラー計算機 HITAC M-660K で要した計算時間を各計算部分に分けて示したものである。特に ETA-10 では通常のベクトル化演算の場合と、あえてこれを外したスカラー演算だけの場合の２つの結果を示した。

反復計算過程中の積**Ａｐ**以外の部分は、主に総自由度数の次元をもつベクトルの内積計算５回とベクトルの和２回からなり、ベクトル処理されるループの長さはいずれも総自由度数である。計算に要した時間を比較するとベクトル化演算ではスカラー演算の場合の約1/18に効率化されており、ベクトル長の長い問題の計算時間の効率化に対してベクトル計算が有効であることが分かる。

一方、反復過程中の積**Ａｐ**はベクトル演算との適合性を考慮していない標準的なアルゴリズムを用いているため、ベクトル長は節点の自由度である。計算に要した時間を比較すると、ベクトル化演算ではスカラー演算の場合と比較して逆に約 1.6倍の計算時間を要しており、ベクトル化に不向きなアルゴリズムであるためにベクトル計算の性能を全く生かしていないことが分かる。しかし、積**Ａｐ**は反復計算中で計算量のもっとも多い部分であり、この部分に要する計算時間の効率化が解法全体の効率化に及ぼす影響は大きい。積**Ａｐ**はマトリックスとベクトルの乗算であり、

表－３　ベクトル化演算とスカラ演算の比較　　単位：秒

| 計算機 | データ入力 | 剛性行列 | 反復計算 | 反力・応力 | 全計算 |
|---|---|---|---|---|---|
| ETA-10 (V) | 1.83 | 4.29 | 38.57 | 8.70 | 53.40 |
| ETA-10 (S) | 1.82 | 6.81 | 28.64 | 13.65 | 50.92 |
| M-660 K | 0.60 | 9.60 | 59.82 | 18.48 | 88.50 |

| 計算機 | 回数 | 反復計算１回 Ａｐ１回 (s) | 反復計算１回 他１回 (s) | 反復計算１回 反復１回 (s) | 全反復 (s) |
|---|---|---|---|---|---|
| ETA-10 (V) | 449 | 0.0854 | 0.0005 | 0.0859 | 38.570 |
| ETA-10 (S) | 450 | 0.0544 | 0.0092 | 0.0636 | 28.641 |
| M-660 K | 449 | 0.1182 | 0.0150 | 0.1332 | 59.822 |

(V)：ベクトル化演算　(S)：スカラ演算

データに依存関係がないためベクトル計算機との適合性を考慮することは可能であると考えられ、以下検討した結果を示す。

## 7．ベクトル計算との適合性を考慮したアルゴリズムについて

ここでは、係数マトリックス**A**の格納方法と積**Ap**の計算方法について、通常考えられるような基準とするアルゴリズムと、本論文で提案するベクトル化を考慮したアルゴリズムのそれぞれを明らかに示す。更に、ＩＣＣＧ法のアルゴリズムを用いてスケーリングを行う場合のベクトル化アルゴリズムもあわせて示す。

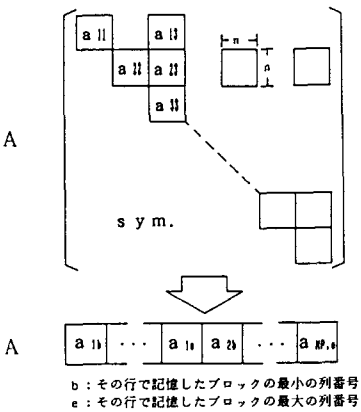
(1)基準とするアルゴリズム

ＣＧ法の最大の利点は、これに要する記憶容量が直接法と比べて少なくてよいという点であり、アルゴリズムを考える上でもこの点にできるだけ配慮するのが普通である。係数マトリックス**A**の格納法としては、マトリックスの対称性とスパース性を考慮し、節点の自由度を単位とするブロックを考え、図－１０に示すように上三角部分の非零マトリックスだけを行方向に詰めて一次元的に記憶するものとした。

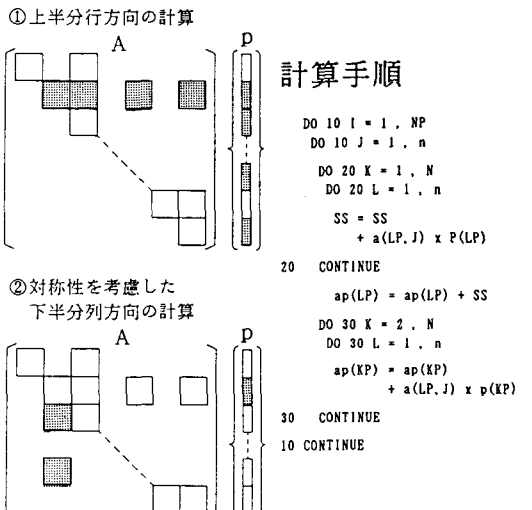
係数マトリックス**A**は上三角部分だけしか記憶していないので、積**Ap**の計算は図－１１に示すように上三角部分と下三角部分に相当する２つの計算部分に分けて行うことになる。計算の手順は、まず30のループで係数マトリックスの上三角部分の行方向のブロックと修正ベクトルの乗算を行い、続いて50のループでこれに対称な位置にある係数マトリックス下三角部分の列方向のブロックと修正ベクトルの乗算を行う。ここで、ベクトル処理されるループ長は、ブロックの大きさ、すなわち節点の自由度である。

(2)ベクトル化を考慮したアルゴリズム

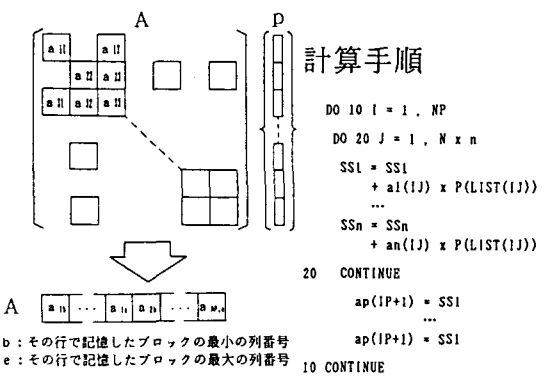
ベクトル計算との適合性を考慮して、ベクトル長をできるだけ長くとるために、基準とするアルゴリズムを次のように変更した（図－１２）。第１に、積**Ap**の計算を係数マトリックスの上三角部分と下三角部分に相当する２つの計算部分に分けることを避けるために、係数マトリックスは対称性を考慮せずに全ての非零ブロックを行方向に一次元的に記憶するものとした。このため約２倍の記憶容量が必要となる。第２に、係数マドリックスは非零ブロックだけを記憶しているので、乗算の相手となる修正ベクトルの要素は不連続となりこれを求めるには手間がかかるが、記憶した係数マトリックスの各要素の列番号をリストベクトルに記




図－１０　係数マトリックスの対称性を考慮した記憶法




図－１１　基準とするアルゴリズム




図－１２　ベクトル化を考慮したアルゴリズム

憶することにより行方向の乗算を１つのDOループで連続的に処理できるようにした。これにより、アルゴリズムも比較的簡単な見通しのよいものとなった。

(3)ＩＣＣＧ法のアルゴリズムを用いたスケーリングのベクトル化アルゴリズム

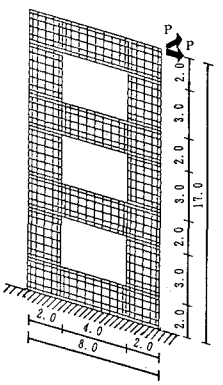
ＩＣＣＧ法では、反復計算ごとに１回の前進代入および後退代入計算が必要になるが、これらの計算部分にはデータに依存関係が存在しベクトル化が困難である。しかし、４．(１)で述べたようにＩＣＣＧ法のアルゴリズムでスケーリングを行う場合には、非対角要素がなくなるためにデータに依存関係がなくなりベクトル化が可能となる。

普通、対角ブロックでスケーリングを行う場合は対角ブロックだけを取り出したマトリックス**D**の 1/2乗を求める必要があるが、この計算のアルゴリズムは非常に複雑で見通しの悪いものとなる。一方、ＩＣＣＧ法のアルゴリズムを用いてスケーリングを行う場合は、各対角ブロックの逆行列を１回だけ求めるだけで比較的簡単なアルゴリズムとなる。この場合、反復計算ごとにマトリックス$\mathbf{D}^{-1}$と残差ベクトルとの乗算が必要となるが、各対角ブロックに対応する計算部分をインラインに展開することによりベクトル長は節点数となり、ベクトル計算機により高速に処理することができる。

## ８．数値計算例に基づくいくつかの検討

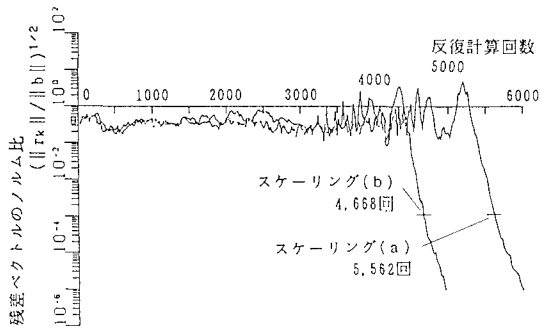
７．に示したアルゴリズムの有効性を検討するために、図－１に示した解析対象に対して面内と面外の荷重の両方を考慮して１節点６自由度として解析する総自由度数 5,490の問題(以下複合問題(１)と呼ぶ(図－１３))と、この問題の要素分割の縦横をさらに２分割して総要素数を４倍にした複合問題(２)(節点数3,510、要素数3,264、総自由度数20,790)を解析対象として設定した。

解析に用いた計算機システムは、東京大学大型計算機センターのスーパーコンピュータ HITAC S-820/80（以下S820）と東京工業大学総合情報処理センターのスーパーコンピュータETA 10-E（以下ETA10）である。それぞれの計算機のベクトル演算を処理するパイプラインの数は、S820が加算４本、乗算４本、乗算に直列に接続している加算が４本の計１２本、ETA 10では加算と乗算が同時に行えるパイプラインが２本である。（ただしETA10は8CPUのうちの1CPUについてである。）それぞれの最大性能は、S820 が3000MFLOPS、ETA10 が380MFLOPSであると言われている。[11),12)]

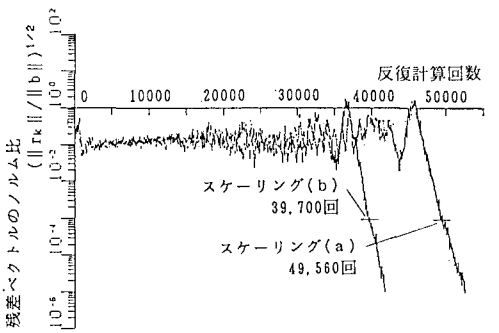


図－１３　複合問題(１)

複合問題(1)に対して３．(a),(b)のそれぞれのスケーリングの方法を適用した場合の収束状況の違いを、図－１４に示す。また、



図－１４　複合問題(１)の収束状況



図－１５　複合問題(２)の収束状況

(a)のスケーリングの方法で７．で示した基準としたアルゴリズム、効率化を考慮しベクトル長を長くしたアルゴリズム、スカイライン法、それぞれで解析に要した計算時間と記憶容量を、(b)のスケーリングの方法では効率化を考慮したアルゴリズムでの計算時間と記憶容量を、使用した計算機ごとにまとめて表－４に示す。基で示すのが基準としたアルゴリズム、効で示すのが効率化を考慮しベクトル長を長くしたアルゴリズムである。複合問題(2)に対する収束状況は図－１５である。この時、(b)のスケーリングの方法で効率化を考慮したアルゴリズムにより要した計算時間を表－４に併せて示す。(a)のスケーリングの方法でベクトル長を長くしたアルゴリズムでは、基準としたアルゴリズムに比較して相当な短縮がみられており、その効果が大きいことがわかる。ここで、S820とETA10ではその短縮率が違うが、これにはパイプライン構成の違いが大きく影響しているものと思われる。複合問題(2)については、スカイライン法による解析は行なっていないが、ちなみにその場合には39MBの容量が必要になってくる。この問題では総自由度数が20,790であるにもかかわらず、約４万回もの反復回数を要している。ＣＧ法は、対象が大きくなればそれだけスカイライン法に比較して記憶容量の面で有利にはなるが、桁落ち等の影響も大きくなり収束性が非常に悪くなるという一面を持っていることが改めて確認された。

表－４　解析に要した計算時間

| | | 解法の種類 | 要する容量 | 反復計算過程 | | | 全計算過程 |
|---|---|---|---|---|---|---|---|
| | | | | 回数 | １回当り(s) | 全過程(s) | (s) |
| 複合問題(1) | S820 | スケーリング(a)基 | 1.52 MB | 5,503 回 | 0.069 | 375.9 | 379.8 |
| | | スケーリング(a)効 | 2.68 MB | 5,483 回 | 0.0032 | 17.5 | 21.6 |
| | | スケーリング(b)効 | 2.90 MB | 4,609 回 | 0.0034 | 15.5 | 19.9 |
| | | スカイライン法 | 5.47 MB | —— | —— | —— | 7.5 |
| | ETA10 | スケーリング(a)基 | 1.52 MB | 5,621 回 | 0.111 | 624.0 | 639.2 |
| | | スケーリング(a)効 | 2.68 MB | 5,562 回 | 0.025 | 139.2 | 154.6 |
| | | スケーリング(b)効 | 2.90 MB | 4,668 回 | 0.027 | 125.8 | 141.4 |
| | | スカイライン法 | 5.47 MB | —— | —— | —— | 26.7 |
| 複合問題(2) | ①* | スケーリング(b)効 | 11.3 MB | 39,000 回 | 0.0123 | 480.4 | 497.1 |
| | ②* | スケーリング(b)効 | 11.3 MB | 39,700 回 | 0.102 | 4053.4 | 4111.4 |

①*：Ｓ８２０　②*：ＥＴＡ１０

９．おわりに

以上、有限要素法で対象とすることとなる疎な大次元連立一次方程式の効率的な解法のアルゴリズムを構築することを目的として、ＣＧ法を基礎とする種々の連立一次方程式のアルゴリズムの収束性、要する計算時間、などその特性を比較検討した。

ＣＧ法の収束性が問題の種類やスケーリングの有無、方法により大幅に異なってくることを確認した。不完全コレスキー分解を介するＩＣＣＧ法では、スケーリングを施すだけのＣＧ法と比較して大幅な収束性の改善がみられることを確認した。不完全コレスキー分解中の加速係数の値は収束性に大きな影響を与えるが、ここでは、加速係数の値を種々変化させて数値計算を行うことにより、その値は対角要素の値が負にならない範囲でＩ．０に近いほど収束性がよいことを明らかにした。

ＩＣＣＧ法では収束性は大幅に改善されるが、反復計算ごとに前進代入および後退代入計算を要するために、計算時間においては収束回数ほどの改善はみられなかった。ベクトル計算機の性能を引き出すためにはベクトル長をできるだけ長くとることが重要であるが、前進代入および後退代入計算にはデータの依存関係が存在するため係数マトリックスと修正ベクトルの積$\mathbf{A}\mathbf{p}$と比べて十分なベクトル長をとることは難しい。このため、ＣＧ法における積$\mathbf{A}\mathbf{p}$の計算部分に着目して、できる限りベクトル長を長くとるアルゴリズムを具体化し、スーパーコンピュータ上での有効性を検討した。通常の方法に比べて容量は多く必要とするが、計算時間を基準とした効率化の点で有効であることを確認することができた。

参考文献

1) 三好,高野,吉田:"スーパーコンピュータによる大規模構造解析",構造工学における数値解析法シンポジウム論文集,Vol.11,pp.217～220,1987

2) 三好,高野,吉田:"スーパーコンピュータによる三次元有限要素解析",構造工学における数値解析法シンポジウム論文集,Vol.10,pp.287～292,1986

3) 藤掛,村野:"不完全共役勾配法とスカイライン法との連立一次方程式解法効率の比較",構造工学における数値解析法シンポジウム論文集,Vol.10,pp.28～33,1986

4) 三好,高野:"構造解析における反復解法について",構造工学における数値解析法シンポジウム論文集,Vol.12,pp.19～22,1988

5) Meijerink,J.A and van der Vorst,H.A:"An iterative solution method for linear systems of which the Coefficient matrix is a symmetric M-Matrix", Math.Comp,Vol.31,No.137,pp.148-162,1977

6) Kershaw,D.S:"The Incomplete Cholesky-Conjugate Gradient method for the iterative solution of system of linear equations",J.Comp.Phys.,No26,pp.43-65,1978

7) Meijerink,J.A and van der Vorst,H.A:"Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical ploblems",J.Comp.Phys.,No.44,pp.134-155,1981

8) 村田、小国、唐木:"スーパーコンピュータ",丸善,1985

9) 島崎眞昭:"スーパーコンピュータとプログラミング",共立出版,1989

10) 戸川隼人:"マトリックスの数値計算",pp.95-102,オーム社,1971

11) 日立製作所ソフトウェア工場言語プログラム部:"S-820の効率的な使用法",東京大学大型計算機センターニュース,Vol.20,No.6,1988

12) 日本シーディーシー株式会社システム営業部:"UNIXベース・スーパーコンピュータETA10の紹介",日本機械学会ＲＣ８９計算機援用機械設計のための有限要素法に関する研究分科会資料,1988

（１９９０年１０月１２日受付）