PNC-TN9410 87-040

単相流解析におけるポアソン型圧力方程式の高速解法の改良

1987年3月

動力炉・核燃料開発事業団
大洗工学センター

複製又はこの資料の入手については，下記にお問い合わせください。
〒311-13　茨城県東茨城郡大洗町成田町4002
動力炉・核燃料開発事業団
大洗工学センター　システム開発推進部・技術管理室

Enquires about copyright and reproduction should be addressed to: Technology Management Section O-arai Engineering Center, Power Reactor and Nuclear Fuel Development Corporation 4002 Narita-cho, O-arai-machi, Higashi-Ibaraki, Ibaraki-ken, 311-13, Japan

動力炉・核燃料開発事業団（Power Reactor and Nuclear Fuel Development Corporation）

PNC-TN9410 87-040
1987年3月

単相流解析におけるポアソン型圧力方程式の高速解法の改良

村松　寿晴*，前川　勇*
二ノ方　寿*

要　　旨

単相3次元汎用熱流動解析コードCOMMIX－PNCにオプションとして組み込んだICCG法および直接解法につき，高速化の観点から以下の改良を施した。

〔ICCG法〕　　行列要素の配列内格納法の改訂

〔直接解法〕　　ウェーブ・フロント法の採用

これらの改良の結果，旧ICCG法および直接解法に比べ，それぞれ以下に示す高速化が実現された。

	スカラー演算	ベクトル演算
〔ICCG法〕	1.89倍	1.27倍
〔直接解法〕	1.28倍	1.15倍

また，各解法を倍精度化することにより，定常解到達までのCPU時間を減少させることができる。その効果は，「もんじゅ」PLOHS解析につき単精度計算に比べ，ICCG法で1.4倍，SOR法で1.2倍，直接解法で1.1倍の加速である。

以下に，各計算における各解法の適応性を示す。

〔定常計算〕

(1) セル数および条件数によらず定常解到達までのCPU時間はICCG法が最も短い。

(2) 相対変動ΔVmax/Vmaxが大きい間はICCG法が，また小さくなるとSOR法が質量バランスを短時間の計算で満足する。

(3) 直接解法の優位性は，現在の所定常計算では見出されない。しかし，質量バランスを厳密に取る必要がある場合には有効である。

〔過渡計算〕

(1) 過渡変化の激しい問題にはICCG法が最も有利であり，SOR法がこれに続く。

* 大洗工学センター安全工学部高速増殖炉工学室

PNC-TN9410 87-040
March, 1987

Improvement of Fast Elliptic Solver in Single-Phase Thermal-Hydraulic Analysis

Toshiharu Muramatsu*, Isamu Maekawa*,
and Hisashi Ninokata*

Abstract

The ICCG (Incomplete Choleski Conjugate Gradient) method and the direct inversion method implemented in COMMIX-PNC has been improved. The improved items are as follows;

[ICCG method] Revision of storing scheme of coefficient elements in computer memory array, and

[Direct inversion method] Implementation of the wave-front method.

The following results have been obtained in comparison with the old methods in COMMIX-PNC.

[CPU time for the old method / CPU time for the new one]	Scalar processing	Vector processing
[ICCG method]	× 1.89	× 1.27
[Direct inversion method]	× 1.28	× 1.15

And, Total CPU time needed for a steady-state calculation has been reduced by 1/1.4 times for the ICCG method with double precision.

The performances of each solver are as follows;

[Steady-State Run]

(1) The ICCG solver is fastest. The efficiency does not depend the number of a computational cell and matrix condition.

(2) At the initial stage of steady-state calculations where a relative variation of dependent variables is still large, e.g., above 10^{-2}, the ICCG method is superior to the SOR method. However, with relative variation of dependent variables below

* FBR Engineering Sec., Safety Engineering Div., OEC, PNC

10^{-3}, the ICCG method requires more CPU time to the convergence than the SOR method.

(3) The direct inversion method is effective for the problems with under the requirement of tight mass balance.

[Transient Run]

(1) In the fast transient case, the ICCG method is superior to the SOR and the direct inversion method.

目　　次

List of Figures

第1章 緒　　言

単相3次元汎用熱流動解析コードCOMMIX－PNCでは，圧力に関するポアソン方程式から導びかれる連立1次方程式を解く必要がある。この連立1次方程式に関連した大型疎行列を高速に解くため，過去2年間に渡り数種の解法について改良を施してきた[1,2]。これら一連の作業から，高速解法を実現する上で最も障害となるのは，行列演算を実施する上でのインデックス操作の存在であり，これによりプログラム制御が複雑化し，またプログラムのベクトル化が計れない。

そこで，このインデックス操作を廃止あるいは最小限にし，ベクトル化が効果的に行われるようにするため，ICCG（Incomplete Choleski Conjugate Gradient）法については行列要素の格納方法を改訂し，また直接解法についてはウェーブ・フロント法を採用することを試みる。

本報告書は，今回改良を行うICCG法および直接解法の詳細と旧ICCG法および旧直接解法との計算速度の比較・検討を記すものである。

第2章　計算手法の改良

2.1　ICCG法

先に報告したICCG法の改良[2]では，係数行列の格納に以下の様なRow－by－Rowによる1次元格納方法を採用した。

この方法では，係数行列A

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & 0 \\ 0 & a_{32} & a_{33} & 0 & 0 & 0 \\ 0 & a_{42} & 0 & a_{44} & 0 & a_{46} \\ 0 & a_{52} & 0 & 0 & a_{55} & 0 \\ 0 & 0 & 0 & a_{64} & 0 & a_{66} \end{bmatrix} \qquad (2.1)$$

の非零要素のみを，以下の様な1次元配列中に格納するものである。

$$A = \boxed{a_{11}\,|\,a_{12}\,|\,a_{21}\,|\,a_{22}\,|\,a_{23}\,|\,a_{24}\,|\,a_{25}\,|\,a_{32}\,|\,a_{33}\,|\,a_{44}\,|\,a_{46}\,|\,a_{52}\,|\,a_{55}\,|\,a_{64}\,|\,a_{66}} \qquad (2.2)$$

（2.2）に示す1次元配列中の数値に関する要素の位置情報および各列の区切り情報も同様に，次に示すような1次元配列中に格納する。

$$JA = \boxed{1\,|\,2\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,2\,|\,3\,|\,2\,|\,4\,|\,6\,|\,2\,|\,5\,|\,4\,|\,6} \qquad (2.3)$$

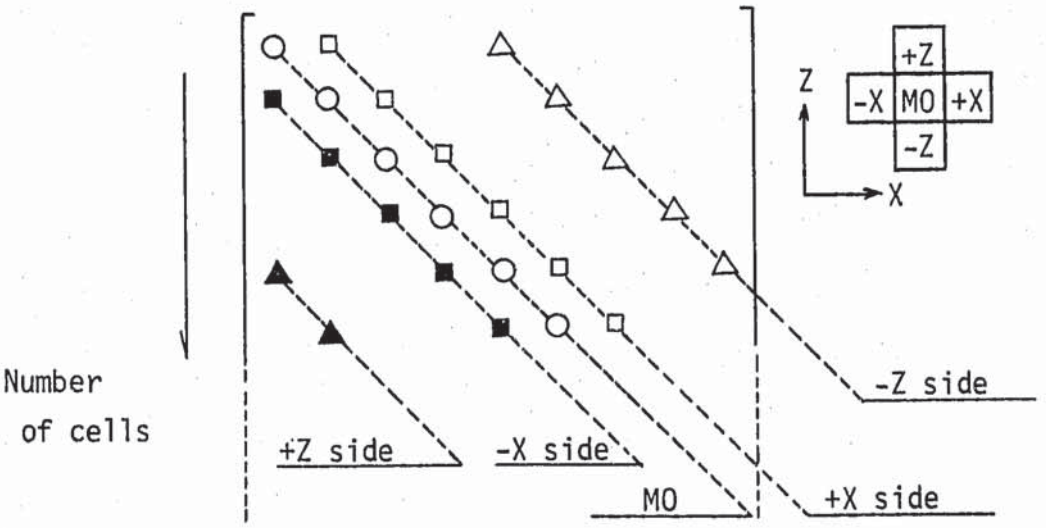
$$IA = \boxed{1\,|\,3\,|\,8\,|\,10\,|\,13\,|\,15\,|\,17} \qquad (2.4)$$

以上の格納方法を採用することにより，計算機の記憶領域の節減には有効となる。しかし，ICCG法処理群のうちの不完全コレスキー分解処理等で頻繁に実行される同列要素（例えば，前出行列A内のa_{25}とa_{55}）の抽出処理等で，特定のij成分の値a_{ij}を配列（2.2）の中から探し出すためには，位置情報を格納している配列JAおよび各列の区切り情報を格納している配列IAをその都度参照する必要がある。このためプログラム内にIF文が数多く組み込まれ，その結果プログラム制御が複雑になるとともにベクトル化が効率良く行なえなくなる。そこで，これら不具合を回避する方法を採用した。

COMMIX－PNCに於て，2次元体系について5点差分近似処理を行なった場合に作成される係数行列* の形は一般的に（2.5）の様に対称形となる。このように，非零要素が現われる場所は対角要素の上下に2本づつ計5本の線（注目セルMOおよび±X側，±Z側）となる。

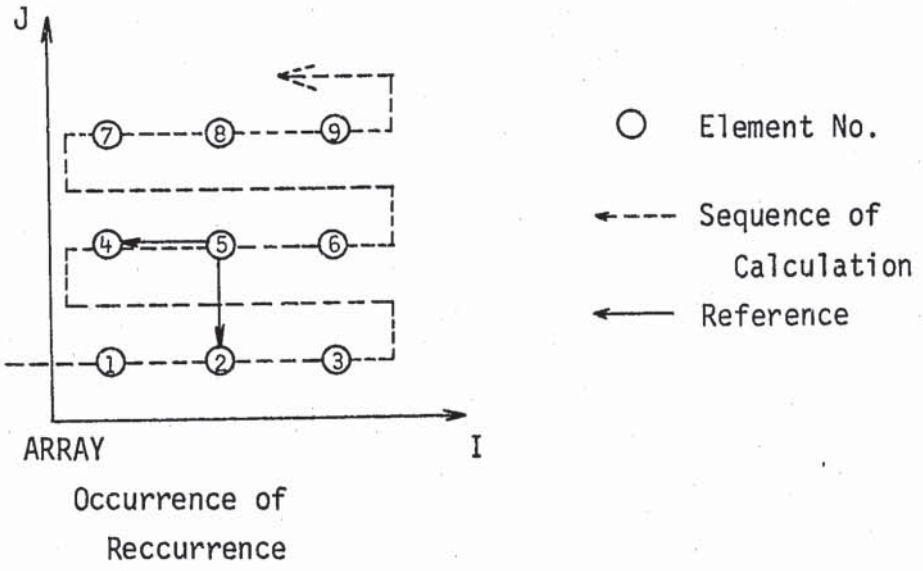
＊ 係数行列の作成については，文献1）を参照のこと。

そこで，これら各線内要素を解析体系形状の２次元配列中に格納し，計５枚の配列で係数行列を表わすことを考える。同様の事を３次元体系について行なう場合には，７点差分近似処理が必要となり，係数行列中の非零要素は対角要素の上下３本づつ計７本の線となる。このため７枚の２次元配列が必要となる。



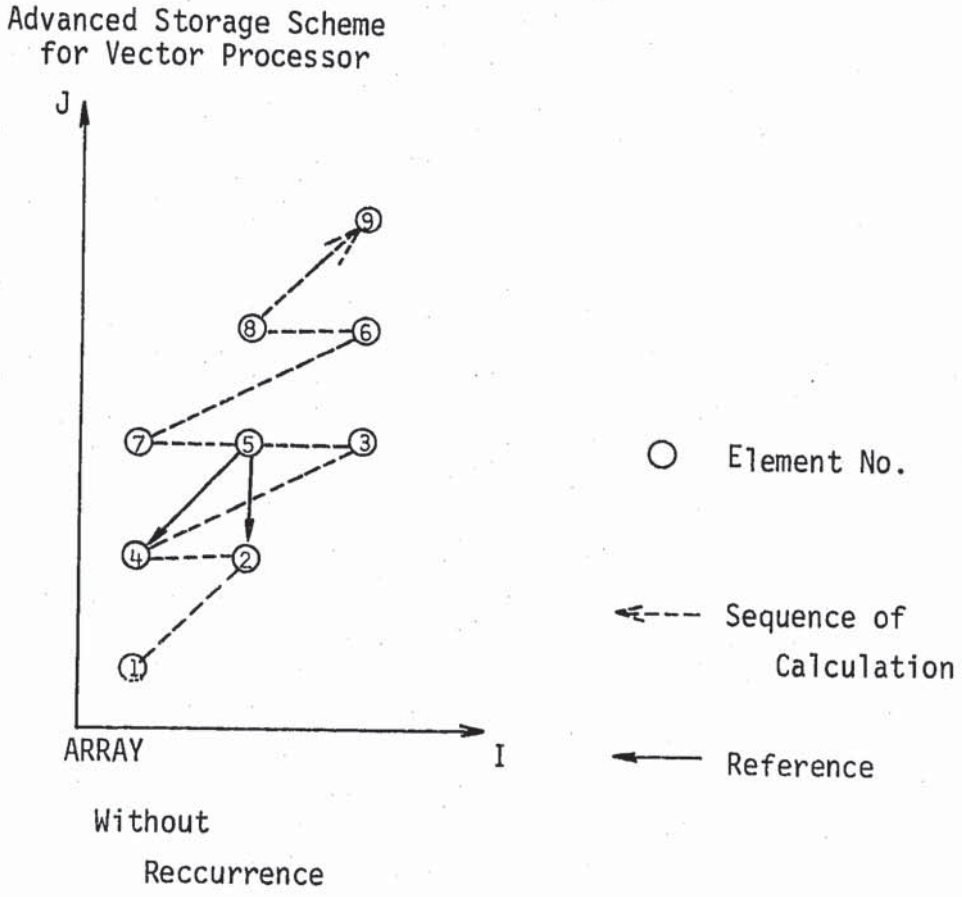
(2.5)

ところで，係数行列内の非零要素を前述の２次元配列内に通常最も良く使用される順序で格納して，下図の点線の順序で計算してゆくことを考えると，任意のｉ番目(⑤)の計算ではｉ－１番目とｉ－３番目（ここでは，②と④の要素）の計算結果を参照する必要があるため，ベクトル計算を実施する上で以下の様な回帰的参照関係が生ずる。すなわち，前の要素の処理が終了しなければ，自分自身の要素の処理に移ることができない。



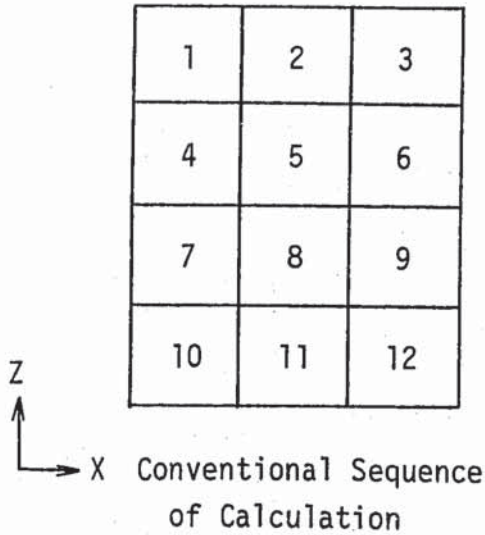
このような回帰的参照関係の存在は，ベクトル演算（並列処理）を不可能にする。すなわち，前に示した例について言えば，⑤の演算を行なうためには，④と②についての演算が終了している必要があり，④と⑤の演算は並列処理ができない。そこで，２次元配列への格納を要素①

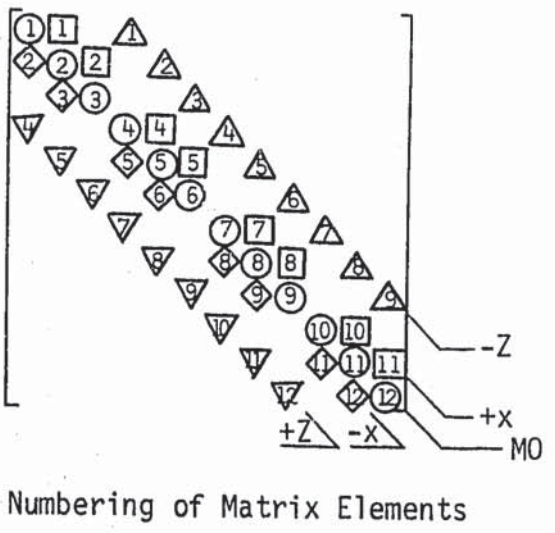
に対して斜めに格納し，縦方向に１段ずらす格納方式に変更することとする。



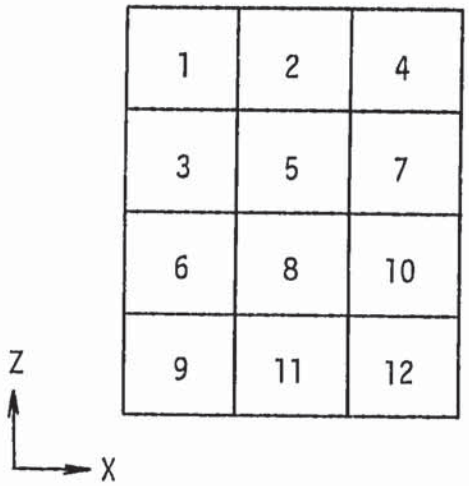
このような格納方式では，⑤の演算を行なうためには，前と同様にやはり②と④の要素の計算結果が必要であるが，⑤と③等の要素間には関連がないため，⑤と③の要素の並列処理が可能である。

例として，下図の左に示す様な２次元解析体系と計算セルの番号付けを考える。この場合の係数行列の非零要素は下図右のようになる。ここで係数行列の非零要素を示す図の記号の中の数字はセル番号を示す。





この計算セルの計算順序で計算を実行すると先に述べた通り，回帰的参照関係のある方法となる。このため，計算セルの計算順序を前に述べたように斜めに実行してゆくことを考える。



すなわち，計算順序はここで，最大12個の非零要素の値をセル1に関する数値を（1,1）の位置に格納して，以後セル2の数値を（1,2）の位置に，セル3の数値を（2,1）の位置という具合に右上から左下への方向に配列しておけば，

Sequence of Calculation
for Vector Processor

のようになり，回帰的参照関係が生じない計算が行なえるようになる。

さらにインデックス処理なしの演算を行なうため，各列を1行づつ縦に下方にずらし，配列を規定すると係数行列の非零要素の格納はそれぞれの planeについて以下の様になる。

MO plane

1	Null	
3	2	
6	5	4
9	8	7
	11	10
Null		12

+X plane

1	Null	
3	2	
6	5	
9	8	
	11	
Null		

-X plane

	Null	
	2	
	5	4
	8	7
		10
Null		

+Z plane

	Null	
3		
6	5	
9	8	7
	11	10
Null		12

-Z plane

1	Null	
3	2	
6	5	4
	8	7
		10
Null		

先に示した行列要素の格納を具体的に示すと以下の通りとなる。

MO plane

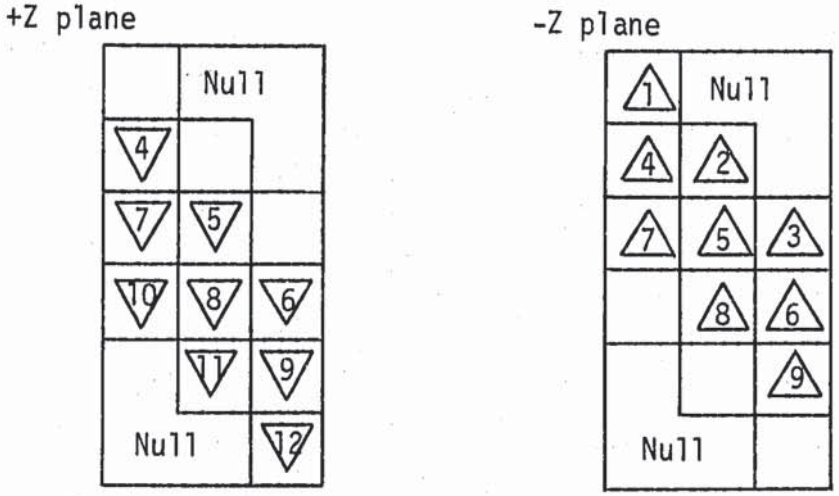
①	Null	
④	②	
⑦	⑤	③
⑩	⑧	⑥
	⑪	⑨
Null		⑫

+X plane

1	Null	
4	2	
7	5	
10	8	
	11	
Null		

-X plane

	Null	
	2	
	5	3
	8	6
	11	9
Null		12



この規定された配列のうちNullの位置には0.0の値が格納されている。

このような要素格納方法を用いたことにより，インデックス処理無しの演算が実現される。

例えば簡単のため，ベクトル**Y**とベクトル**P**との内積計算を考えてみると，

$$D = (\mathbf{Y}, \mathbf{P}) = \sum_{i=1}^{n} y_i p_i \qquad (n : \text{Array} \quad)$$

$$= \sum_{i=1}^{n} \left[\begin{array}{ccc} 1 & \text{Null} & \\ 3 & 2 & \\ 6 & 5 & 4 \\ 9 & 8 & 7 \\ & 11 & 10 \\ \text{Null} & & 12 \end{array} \right] * \left[\begin{array}{ccc} 1 & \text{Null} & \\ 3 & 2 & \\ 6 & 5 & 4 \\ 9 & 8 & 7 \\ & 11 & 10 \\ \text{Null} & & 12 \end{array} \right]$$

Y Array　　**P** Array

の演算に置き替えられ，プログラミング上は，

```
D = 0.0
DO 10  I = 1,6
DO 10  J = 1,3
D = D + y (I, J) * P (I, J)
```

```
      10 CONTINUE
```

のようなインデックス操作の無い記述となり，ベクトル演算に最適なプログラミングとなる。

今回のICCG法関係の全ルーチンは，以上の操作により計算が実施される。

2.2 直接解法

先に報告した直接解法の検討[2]では，Yale大学で開発されたLDU分解パッケージ[3,4]を採用し，係数行列の格納には旧ICCG法と同様の1次元配列格納方法を採用した。しかし，この場合にもインデックス処理操作が頻繁に行なわれ，ベクトル化する上で支障を来していた。そこでこのインデックス処理操作を最小限に押えるため，直接解法にウェーブ・フロント法* を今回適用した。この手法の詳細は文献5）に譲るが，基本的にはfill−inが生ずる場所をあらかじめ配列（フロント行列）中に規定し，これに基づいて計算を実行してゆく手法である。したがって，係数行列の形（非零要素の係数行列内の位置）が解析体系の決定とともに定まれば参照すべき要素も一意的に決定されるため，この参照情報配列を一度作成し，以後この情報を基に計算を実行してゆくものである。

---

* ただし，In−Core処理のみで外部ファイルは使用しない。

第3章　各解法の比較および検討

3.1　数値実験用解析体系

今回の改良の効果を確認するため，Fig. 3.1 に示す5種類の基本問題体系を設定した。以下に体系形状データおよび解析条件を示す。

	Case 1	Case 2	Case 3	Case 4	Case 5
Geometry	Box-2D	←	←	←	←
No. of Cell	27	108	432	864	1728
No. of Mesh	6×6	12×12	24×24	48×24	48×48
Δx [m]	0.1	0.05	0.025	0.0125	0.0125
Δz [m]	0.1	0.05	0.025	0.025	0.0125
Inlet Veloc. [m/s]	0.1	←	←	←	←
Inlet Temp. [°C]	20.0	←	←	←	←
Boundary Cond.	No-Slip with Adiabatic				
Δt [Sec.]	0.1	←	←	←	←
No. of Outer Iter.	1	←	←	←	←

上記した体系は，改良の効果を見るためのものであり，物理的意味は無い。また，比較のため，文献2)で使用したものと同一である。計算は，SIMPLEST版による層流計算であり，エネルギー計算は行なわない。計算順序は，体系内の20°Cの水が静止している状態から0.1m/s入口流速を与え，定常解が得られるまで計算を行なう。

3.2　改良の効果およびSOR法との比較

旧ICCG法と今回の改良型ICCG法について，CGループ1回の反復に要するCPU時間を計算セル数の関数として表わした結果がFig. 3.2である。また，以下に各セル数のケースについて旧ICCG法を基準とした加速率を示す。

	旧ICCG法(S)* (msec.)	改良型ICCG法(S)* (msec.)	加速率
Case 1 (27 セル)	0.451	0.451	1.226
2 (108 セル)	3.170	2.472	1.282
3 (432 セル)	14.742	12.140	1.214
4 (864 セル)	31.322	25.437	1.231
Case 5 (1728 セル)	78.146	64.989	1.202

$$\left(\text{但し，加速率} = \frac{\text{旧ICCG法でのCPU時間}}{\text{改良型ICCG法でのCPU時間}} \right)$$

以上の結果から，今回の改良による平均の加速率を計算すると約1.23となる。

Fig. 3.3は，COMMIX−PNCに於て1時間ステップ計算する上で処理される係数行列作成，不完全コレスキー分解およびCGループの時間的平均分担率を$\Delta Vmax/Vmax^{**} \approx 10^{-2}$の時点での値を用いて計算セル数の関数として示したものである。旧ICCG法「○」では，計算セル数が増加するに従がい不完全コレスキー分解の占める割合が顕著に増加するのに対し，改良型ICCG法「△」，「□」ではスカラー演算(S)およびベクトル演算(V)ともそれ程大きな増加は示さない。一方，CGループ(Solution Process)の占める割合は，旧ICCG法に比べてかなり大きくなっている。この旧ICCG法と改良型ICCG法との分担率の違いは，正にインデックス操作の有無によるものである。すなわち，計算セル数が増加するに従い収束解を求めるためのCGループ反復回数も増加するが，この増加割合の方が一過性の不完全コレスキー分解の占める割合よりも大きいことを示している。この傾向は，文献2)の時点の傾向とは逆転しており，このことからインデックス処理操作の負担がかなり大きかったと言える。

Fig. 3.4 ～ Fig. 3.8に，Case 1からCase 5についての定常状態到達までの収束曲線の比較を示す。また，以下に定常状態到達までに費したCPU時間をCOMMIX−PNCオリジナル解法のSOR法とともに示す。

	SOR法(S) (sec)	旧ICCG法 (sec)	改良型ICCG法 (sec).
Case 1	44.48	9.67	3.55
2	33.14	35.47	21.59
3	180.82	170.88	118.35
4	386.63	382.44	196.91
Case 5	1617.21	1517.29	781.89

まず，旧ICCG法と改良型ICCG法との比較では，いずれのケースについても改良型ICCG

* VP−100 スカラー演算

** (解析体系内最大変動流速/解析体系内最大流速) = 相対変動最大値

法の方が短いCPU時間で定常状態に到達しており，その加速率は平均約1.89倍である。一方，SOR法との比較でも，やはり改良型ICCG法が短いCPU時間で定常状態に到達しており，その加速率は平均で約2.02倍である。特に，ケース1の様にオリジナルの係数行列条件数*が10^4に近く7842と悪い場合には，悪条件のまま解くSOR法と前処理してから解くICCG法とで顕著な差となってCPU時間に現われている。

Fig. 3.9およびFig. 3.10にCase 5における時間ステップ1での質量バランスの収束過程と時間ステップ720(ΔVmax/Vmax$\approx 10^{-2}$)での質量バランスの収束過程の結果を示す。図中に示したSOR法に対するICCG法の収束傾向は，文献1)に示した時点のものと同じである。すなわち，計算開始とともに流体静止状態から動き始め，安定になるまでの間における過渡変化の度合いにより両解法のメリット・デメリットが明確に区分される。したがって，計算初期から流況が準安定状態（定常状態に近い状態）になるまでの間はICCG法が有利であり準安定常態に入った後はSOR法が有利になることが理解できる。

以下に，直接解法に関する比較結果を示す。Fig. 3.11～Fig. 3.15は，各ケースについての定常状態到達までの収束曲線であり，また以下は，定常状態到達までに費したCPU時間の比較である。

	SOR法(S) (sec)	旧直接解法(S) (sec)	改良型直接解法(S) (sec)
Case 1	44.48	6.44	5.03
2	33.14	59.39	51.27
3	180.82	298.13	263.63
4	386.63	1143.19	913.68
Case 5	1617.21	5727.44	4410.13

まず，旧直接解法と改良型直接解法との比較では，全ケースについて改良型直接解法が短いCPU時間で定常状態に到達している。この平均加速率は，約1.28倍である。一方SOR法との比較では，ケース1を除きSOR法が短いCPU時間で定常状態に到達している。

以上より，改良型直接解法は旧直接解法に対しCPU時間の面で改良の効果が現われているが，SOR法に対しては十分ではない。

3.3 ベクトル化の効果

今回改良を行なったICCG法および直接解法をFACOM VP-100システムを用いてベクトル演算を実施し，その効果を確認した。このベクトル演算による定常状態到達までの収束曲線をICCG法についてはFig. 3.4～Fig. 3.8内に「△」印で，また直接解法についてはFig. 3.11～Fig. 3.15内に「◇」印で併記した。以下にICCG法を用いた場合の定常状態到達までのCPU時間をスカラー演算の場合の値とともに示す。

	スカラー演算 (sec)	ベクトル演算 (sec)	加速率 **
Case 1	3.55	2.87	1.24
2	21.59	18.91	1.14
3	118.35	100.21	1.18
4	196.91	155.10	1.27
Case 5	781.89	509.47	1.53

$$\left(**\text{加速率}=\frac{\text{CPU時間（スカラー演算）}}{\text{CPU時間（ベクトル演算）}}\right)$$

この結果から，ICCG法をベクトル化し演算したことによって，定常状態到達までのCPU時間がスカラー演算の場合のそれに比べ平均約0.79倍に短縮されたことが判かる。また，前節で示したSOR法のそれに比べると平均して約0.35倍に短縮されていることが判かる。

以下に直接解法を用いた場合の定常状態到達までのCPU時間をスカラー演算の場合の時間とともに示す。

	スカラー演算 (sec)	ベクトル演算 (sec)	加 速 率
Case 1	5.03	4.36	1.15
2	51.27	46.39	1.11
3	263.63	231.25	1.14
4	913.68	798.46	1.14
Case 5	4410.13	3616.31	1.22

この結果から，ベクトル演算を行なったことにより，定常状態到達までのCPU時間がスカラー演算の場合よりも平均約0.87倍に短縮されたことが分かる。しかし，ベクトル化を施してもなお全ケースについてSOR法の方が時間的に有利であることは変わらない。

3.4 総合評価

上記してきた数値実験用解析体系に於ける各解法の比較および検討の結果から，各解法について以下の事項が明らかとなった。

(1) 解析体系の計算セル数の大小によらず今回改良を行なったICCG法が最も短いCPU時間で計算を実行する。

(2) 直接解法は逆行列を直接求めるため計算セル数の面で使用上制約を受け，強制対流場での使用では利点は見出されない。ただし，文献2)で触れた通り，自然対流問題に於ては精度の面でいづれの解法よりも有利となる。

＊係数行列が持つ固有値の最大と最小の比。この値が約10^4を越えると悪条件となり収束が悪化する。

第4章 適 用 計 算

4.1 高速実験炉「常陽」MK－I自然循環試験解析

文献1,2)で取り上げた高速実験炉「常陽」MK－I炉心での自然循環試験を対象とした2次元解析問題に今回改良を行なったICCG法および直接解法を適用した。

解析体系のセル分割はFig. 4.1に示す通りであり，解析条件等は全て文献6)のものを使用した。Fig. 4.2に定常状態到達までの収束曲線を各解法について示す。また，以下に各解法が定常状態到達までに費したCPU時間を示す。

	スカラー演算 (sec)	加速率*	ベクトル演算 (sec)	加速率*
SOR法	1121.3	—	未実施	—
旧ICCG法	1155.6	0.97	未実施	—
改良型ICCG法	681.5	1.65	483.5	2.32
旧直接解法	2473.2	0.45	未実施	—
改良型直接解法	1936.8	0.58	1614.5	0.69

まず，スカラー演算に於けるICCG法と直接解法それぞれの旧型と改良型の比較では，いずれの解法についても改良型の解法を用いた場合の方が短いCPU時間で定常状態に到達している。この加速率は，ICCG法で平均約1.3倍であり，それぞれ第3.2節で示した程度の加速率が得られている。

一方，ベクトル演算を行なった場合には，スカラー演算の場合に比べてICCG法で約1.4倍，直接解法で約1.2倍の加速率を得ている。ここで，最も短いCPU時間で定常状態に到達している解法はICCG法であり，この後，SOR法，直接解法と続く。直接解法について，計算セルの多さの面から時間的に不利となる傾向は，文献2)に示した時の状況と変わらない。

以下に計算を倍精度で実施した場合の結果について述べる。一般に共役勾配法系統の解法は計算中の数値丸め誤差に弱いことが知られており，この効果は係数行列の条件数が大きい程反復回数が多いため顕著に現われる。そこで，プログラム全体を倍精度として，単精度による結果と比較した。結果をFig. 4.3に示す。この結果から，倍精度で演算を行なった方が，いづれの解法も短い時間で定常解に到達することが判かる。特にこの傾向は，ICCG法で顕著である。以下に各解法が定常状態到達するまでに費したCPU時間を示す。

＊SOR法でのスカラー演算に対する加速率

	単精度(S) (sec)	倍精度(D) (sec)	加速率(S/D)
SOR法	1121.3	1023.7	1.10
改良型ICCG法	681.5	488.6	1.39
改良型直接解法	1936.8	1816.9	1.07

以上のことから，ICCG法を実機問題に適用する上では，倍精度で実行した方が有利であるということが明確に言える。

一方，Fig. 4.2およびFig. 4.3から分かる通り，いづれの解法についてもΔVmax/Vmaxの変動が10^{-3}を越えたあたりから収束がゆるやかとなっている。これは各時間ステップで求める反復過程の変動がΔVmax/Vmaxの変動を上回っていることによるものと思われる。そこで各時間ステップでの質量バランス判定条件(EPS 1)をデフォルト値の10^{-4}から10^{-5},10^{-6}に変更し，同じく倍精度で計算を実行した。結果をFig. 4.4に示す。 この結果より分かる通り，ICCG法については，EPS 1の変更により収束曲線の悪化は解消され，ほぼ同じ傾きで定常解の$5 * 10^{-5}$に到達している。ただし，各時間ステップでの反復回数は質量バランスの判定条件を厳しくしたことにより増加しているため，定常状態到達までのCPU時間は増加している。

SOR法についてはほぼ各時間ステップで指定判定条件に到達せず時間が進行しており，反復回数の増加のため多くのCPU時間を費している。直接解法については，1回の解を求める処理で質量残差が10^{-14}まで減少するため，EPS 1を10^{-5}，10^{-6}にした場合もEPS 1を10^{-4}にした場合と同じ収束曲線をたどる。

以下に各解法が定常状態到達までに費したCPU時間を示す。

	EPS 1=10^{-4}のケース (sec.)	EPS 1=10^{-5}のケース (sec.)	EPS 1=10^{-6}のケース (sec.)
SOR法	1023.7	3600 sec. で定常未到達	←
改良型ICCG法	488.6	657.2	789.4
改良型直接解法	1816.9	1816.9	1816.9

Fig. 4.5にEPS 1を10^{-4}から10^{-6}に変化させた場合のICCG法に対する他解法のCPU時間比を計算開始からΔVmax/Vmax=10^{-2}の期間（急激な過渡領域）について示す。この結果から，直接解法はEPS 1が10^{-5}以上でかつ急激な過渡領域においてICCG法よりも良好な結果を示すことが分かる。一方，SOR法は，EPS 1を10^{-5}に設定した時点から各時間ステップで指定された質量バランスを満足しなくなり，圧力に関する反復につき，指定回数の全反復で時間が進行するようになる。

Fig. 4.6に同様の比較を定常状態到達までの期間（ΔVmax/Vmax=5×10^{-5}）について示す。この比較は，急激な過渡領域（計算開始～ΔVmax/Vmax=10^{-2}）および緩やかな過渡領域（ΔVmax/Vmax=10^{-2}～定常）の両方を含んだ計算全体についての比較である。この結果か

らも，EPS1を10^{-5}以上に設定することにより直接解法の優位性が現われ始めているのが分るが，流動安定期に入るに従がい反復回数が減少してゆくICCG法に勝るまでには至らない。しかし，この直接解法の傾向からすると，EPS1をより厳しくなければならない様な計算（例えば自然対流問題等）では，直接解法がICCG法に精度およびCPU時間の面で勝る場合も出てくると考えられる。

一方，SOR法は先に触れた様に，EPS1を厳しく設定するに従がい，精度およびCPU時間の面で他の2者の解法に比べ不利となり，メリットは見出されなくなる。

4.2 高速原型炉「もんじゅ」PLOHS解析

文献2）で実施した標記解析問題に今回改良を行なったICCG法および直接解法を適用した。Fig. 4.7に示したセル分割および解析条件は，全て文献2）で用いたものと同一である。

Fig. 4.8に定常状態到達*1までの収束曲線を各解法について示す。また，以下に解法が定常状態*1に達するまでに費したCPU時間を示す。

	スカラー演算 (sec.)	加速率*2	ベクトル演算 (sec.)	加速率*2
SOR法	3600.0	———	———	———
旧ICCG法	4214.4	0.85	———	———
改良型ICCG法	4113.2	0.88	3976.1	0.91
旧直接解法	5411.9	0.67	———	———
改良型直接解法	5235.3	0.69	5097.4	0.71

以上の結果より判かる通り，旧型，改良型を問わずICCG法および直接解法とも定常状態到達までに費したCPU時間はSOR法のそれよりも多い。この傾向は，ベクトル化を実施しても改善されない。しかし，Fig. 4.8から判る通り，$\Delta Vmax/Vmax$が10^{-3}付近まではICCG法がかなり短いCPU時間で到達している。これは，文献2）でも触れた様にこの問題自体がかなり大きい条件数を持っているためである。すなわち，オリジナルの係数行列が45000という悪条件であるのに対し，不完全コレスキー分解による前処理の結果5000の条件数に変換されていることによる。直接解法については，計算セル数の多さからいづれの解法よりも不利となっている。

次に，この定常状態を初期条件として，Fig. 4.9に示した過渡条件で過渡計算を10秒まで行なった結果について示す。Fig. 4.10に10秒間のシミュレーションに費したCPU時間を各解法について示した。この結果では，10秒間のシミュレーションを最も短い時間で完了して

---

*1 CPU時間3600秒以上で，顕著な定常解への収束が得られなかったため，SOR法に於けるCPU時間3600秒での値（$\Delta Vmax/Vmax=1.6*10^{-4}$）を定常解とした。

*2 SOR法でのスカラー演算に対する加速率

いるのはICCG法であり，最も長いCPU時間を使っているのは直接解法である。今回の改良で各解法のCPU時間は短縮されてはいるもののその順序は変わっていない。

以下に，10秒間のシミュレーションに使用されたCPU時間を示す。

	スカラー演算 (min)	加速率*1	ベクトル演算	加速率*1
SOR法	83	——	——	——
旧ICCG法	50	1.66	——	——
改良型ICCG法	41	2.02	38	2.18
旧直接解法	125	0.66	——	——
改良型直接解法	122	0.68	119	0.70

ここでも前節と同様に，倍精度で計算を実施し，単精度計算の結果と比較した。定常状態到達までの収束曲線をFig. 4.11に示す。この結果でも倍精度計算の方が短かいCPU時間で変動を小さくしている様子が判かる。以下に各解法が定常状態到達するまでに費したCPU時間を示す。

	単精度(S) (sec.)	倍精度(D) (sec.)	加速率(S/D)
SOR法	3600.0*2	3214.3	1.12
改良型ICCG法	4113.2	2917.2	1.41
改良型直接解法	5235.3	4986.0	1.05

上記した結果からも前節で述べたことが裏付けられる。

また，前節と同様に質量バランス判定条件(EPS1)をデフォルト値の10^{-4}から10^{-5}に変更し，倍精度で同様の計算を行なった。結果をFig. 4.12に示す。この結果もFig. 4.4に示した傾向と同じである。すなわち，ICCG法についてはEPS1を10^{-5}に設定した場合，収束曲線はほぼ直線的に減少し，その悪化は見られない。この直線的な$\Delta Vmax/Vmax$の減少は10^{-6}付近まで続く。SOR法は，各時間ステップで質量バランスを満足しないため，CPU時間の増加は多大であり，直接解法はEPS1の10^{-4}, 10^{-5}の違いによる変化は無い。

以下に定常状態*2到達までに費したCPU時間の比較を示す。

	EPS1=10^{-4} (sec.)	EPS1=10^{-5} (sec.)	比　率
SOR法	3214.3	4600secで未到達	
改良型ICCG法	2917.2	4158.7	1.4
改良型直接解法	4986.0	4986.0	1.0

---

*1　SOR法でのスカラー演算に対する加速率

*2　$\Delta max/Vmax=1.6.10^{-4}$を定常解とした。

以上の結果は，前節で示した傾向と同様であり，EPS1を厳しく設定するに従がい，直接解法の優位性が現われ始める。

第5章 緒　　言

単位3次元汎用熱流動解析コード“COMMIX−PNC”にオプションとして組込んだICCG法び直接解法につき，高速化の観点から以下の改良を施した。

〔ICCG法〕

行列演算時に頻繁に実行されるインデックス操作を廃止する為，行列要素の配列内格方法を改訂する。

〔直接解法〕

ICCG法と同様に，インデックス操作を最小限に押える為，ウェーブ・フロント法を採用する。

これら改良の結果，ICCG法については改良前に比べ約1.89倍の全体的なCPU時間の高速化が計られた。一方，直接解法については，1.28倍の高速化が計られた。また，インデックス操作を廃止あるいは最小限に押えたことにより，ベクトル化率が向上し，スカラー演算に対する加速率がICCG法については約1.27倍，直接解法については約1.15倍となった。さらに，倍精度演算を行なうことにより定常状態到達までのCPU時間が短縮されることが確認された。この倍精度演算の効果は，「もんじゅ」PLOHS定常計算につき，ICCG法で約1.4倍，SOR法で約1.12倍，直接解法で約1.05倍の加速である。また，質量バランス判定条件EPS1をデフォルト値の10^{-4}から10^{-5}に変更することにより，ICCG法においてΔmax/Vmaxの変動を10^{-6}程度まで直線的に減少させることが可能である。

以下に各計算における各解法の特徴を列記する。

〔定常計算〕

1. セル数および条件数によらず定常状態到達までのCPU時間は，ICCG法が最も短かい。
2. 計算初期のΔVmax/Vmaxの大きい期間はICCG法が，また小さな期間はSOR法が質量バランスを短時間で満足する。
3. 直接解法の優位性は，現在のところ定常計算では見出されない。しかし，質量バランスを厳密に取る必要がある場合には有効となる可能性がある。

〔非定常計算〕

1. 過渡変化の激しい問題はICCG法が最も有利であり，SOR法がこれに続く。
2. 直接解法はどの計算ケースもICCG法およびSOR法に比べ計算時間がかかるが，質量バランスを厳密にとる必要があるケース（例えば，自然対流場）では，厳密な収束解が得られるという意味で重要な方法である。

謝　　辞

ICCG法の改良にあたっては，FHL（株）フアコム本部システム第5部の南一生氏ならびに網岡大氏には多大なる御協力を得ました。ここに感謝の意を表します。

参 考 文 献

1) 村松　他，“多次元伝熱流動解析コードの整備改良(Ⅱ)タスク3：PCG法の検討”，PNC資料，SN941　85−90，1985.5

2) 村松　他，“多次元伝熱流動解析コードの整備改良(Ⅱ)タスク3：圧力に関するポアソン方程式の高速解法の検討”，PNC資料　N941086−008　1986.1

3) S.C. Eisenstat, "Yale Sparse Matrix Package I: Symmetric Codes", Yale Univ. 1977

4) S.C. Eisenstat, "Yale Sparse Matrix Package II: Nonsymmetric Codes", Yale Univ. 1977

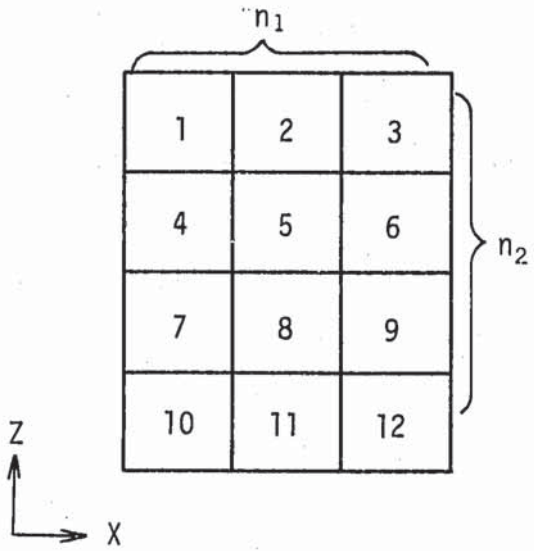
5) 戸川隼人　“マトリクスの数値計算”，オーム社　1979

6) M. Takahashi, "JOYO Mark-I Natural Circulation Analysis with COMMIX-1A", PNC SN941 84-99, 1984

付録　ICCG法使用時のメモリの見積り

ここでは，第2.1節で示したサンプル問題を取り上げ実行に必要なコア・メモリ数を改良前，改良後につき数値で比較する。

解析体系は，以下に示す通りである。



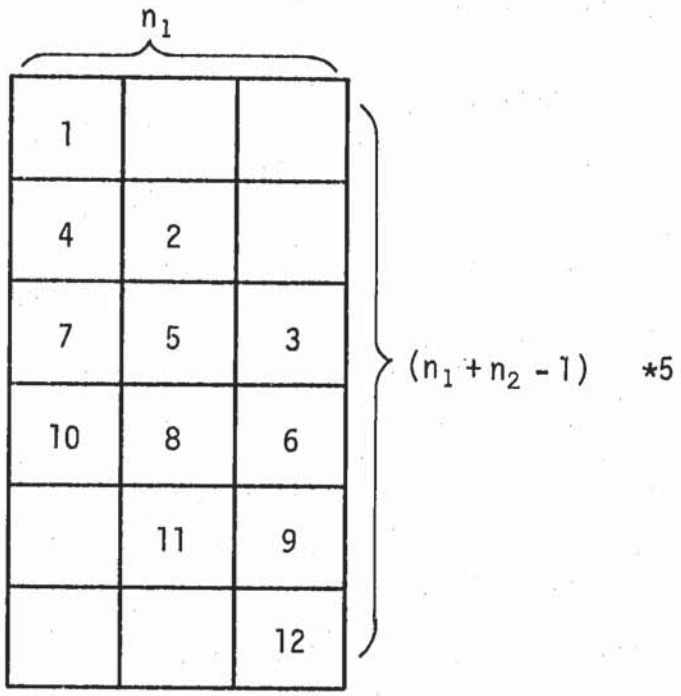
これにより作成される係数行列は，以下の様な形になる。

$$\begin{bmatrix}
1 & 2 & & 3 & & & & & & & & \\
4 & 5 & 6 & & 7 & & & & & & & \\
 & 8 & 9 & & & 10 & & & & & & \\
11 & & & 12 & 13 & & 14 & & & & & \\
 & 15 & & 16 & 17 & 18 & & 19 & & & & \\
 & & 20 & & 21 & 22 & & & 23 & & & \\
 & & & 24 & & & 25 & 26 & & 27 & & \\
 & & & & 28 & & 29 & 30 & 31 & & 32 & \\
 & & & & & 33 & & 34 & 35 & & & 36 \\
 & & & & & & 37 & & & 38 & 39 & \\
 & & & & & & & 40 & & 41 & 42 & 43 \\
 & & & & & & & & 44 & & 45 & 46
\end{bmatrix}$$

改良前のICCG法では，この係数行列を1次元配列中に零要素を取り除いた形で格納していたことから，必要となる配列領域は要素数の46である。

一方，改良後のICCG法では，第2.1節で述べた通り，5点差分近似問題では5枚の2次元配列を用意し，かつ回帰的参照を避けるため，各列を1行づつずらす。

これにより必要とたる配列領域は，



となり，総数90である。

これら係数行列の他に，不完全コレスキー分解を行なった結果を格納する領域および作業領域等を含め，両者を比較すると，

	改 良 前	改 良 後
係数行列	46	90
不完全コレスキー分解結果	92 *1	108 *2
解ベクトル・作業ベクトル等	10＊12 *3	10＊18 *4
	258	378

となる。

＊1　L, L^T =2＊要素数

＊2　$L_D, L_X, L_Z, L_D^T, L_X^T, L_Z^T = 6 * n_1 * (n_1 + n_2 - 1)$

＊3　10＊セル数

＊4　$10 * n_1 * (n_1 + n_2 - 1)$

ここで，要素数をn として一般化すると

配　列	改良前	改良後
係数行列	n	$5 \cdot n_1 \cdot (n_1 + n_2 - 1)$
不完全コレスキー分解結果	$2 \cdot n$	$6 \cdot n_1 \cdot (n_1 + n_2 - 1)$
解ベクトル作業ベクトル等	$10 \cdot n_1 \cdot n_2$	$10 \cdot n_1 \cdot (n_1 + n_2 - 1)$
	$3n + 10 n_1 n_2$	$21 \cdot n_1 \cdot (n_1 + n_2 - 1)$

となる。簡単のため，要素数nを$4 * n_1 . n_2$ と近似し，両者の比をとると

$$R=21n_1(n_1+n_2-1)*\frac{1}{22n_1n_2}$$

$$\approx \frac{(n_1+n_2-1)}{n_2}$$

が得られる。

要素格納に於て，最も効率の良い場合（n_1 ;有限，n_2; n_1 より極めて大）を考えると，

$$R\fallingdotseq\frac{n_2}{n_2}$$

$$\fallingdotseq 1$$

逆に，最も効率の悪い場合（$n_1=n_2$, n_2 が極めて大）には，

$$R\fallingdotseq\frac{2n_2}{n_2}$$

$$\fallingdotseq 2$$

となる。

第4.1節で示した，「常陽」の解析に於ては，$n_1=31, n_2=71$ であるから

$$R=\frac{(n_1+n_2-1)}{n_2}$$

$$=\frac{(31+71-1)}{71}$$

$$=1.42$$

となり，改良前のICCG法に比べ約1.42倍のメモリの増加となる。また，実領域は，以下に示す通りである。

オリジナル（Ver.12.0）	0.85 Mb
旧ICCG法	2.60 Mb
改良型ICCG法	3.70 Mb

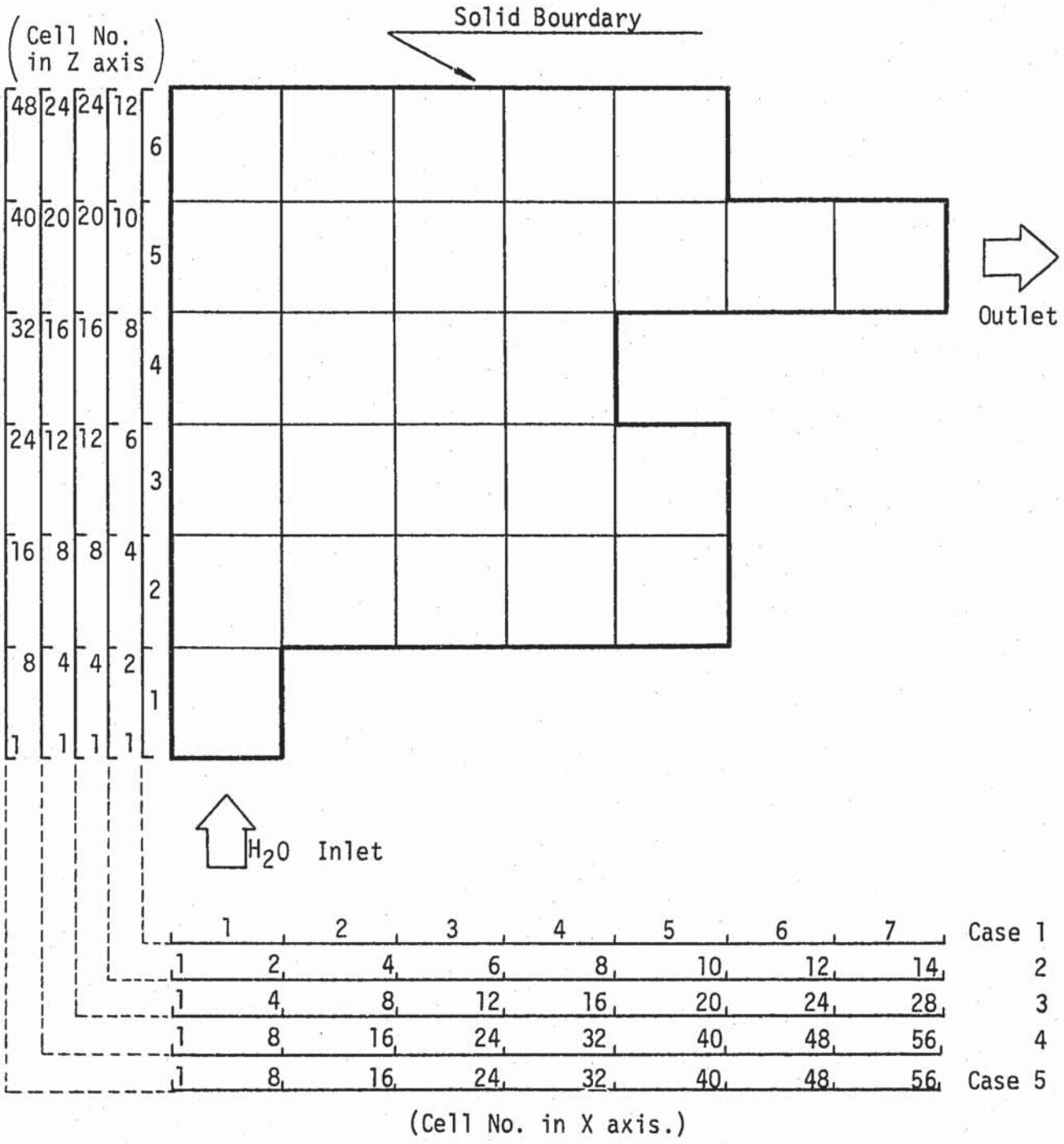


Fig. 3.1 Mesh Arrangements for Numerical Experiment

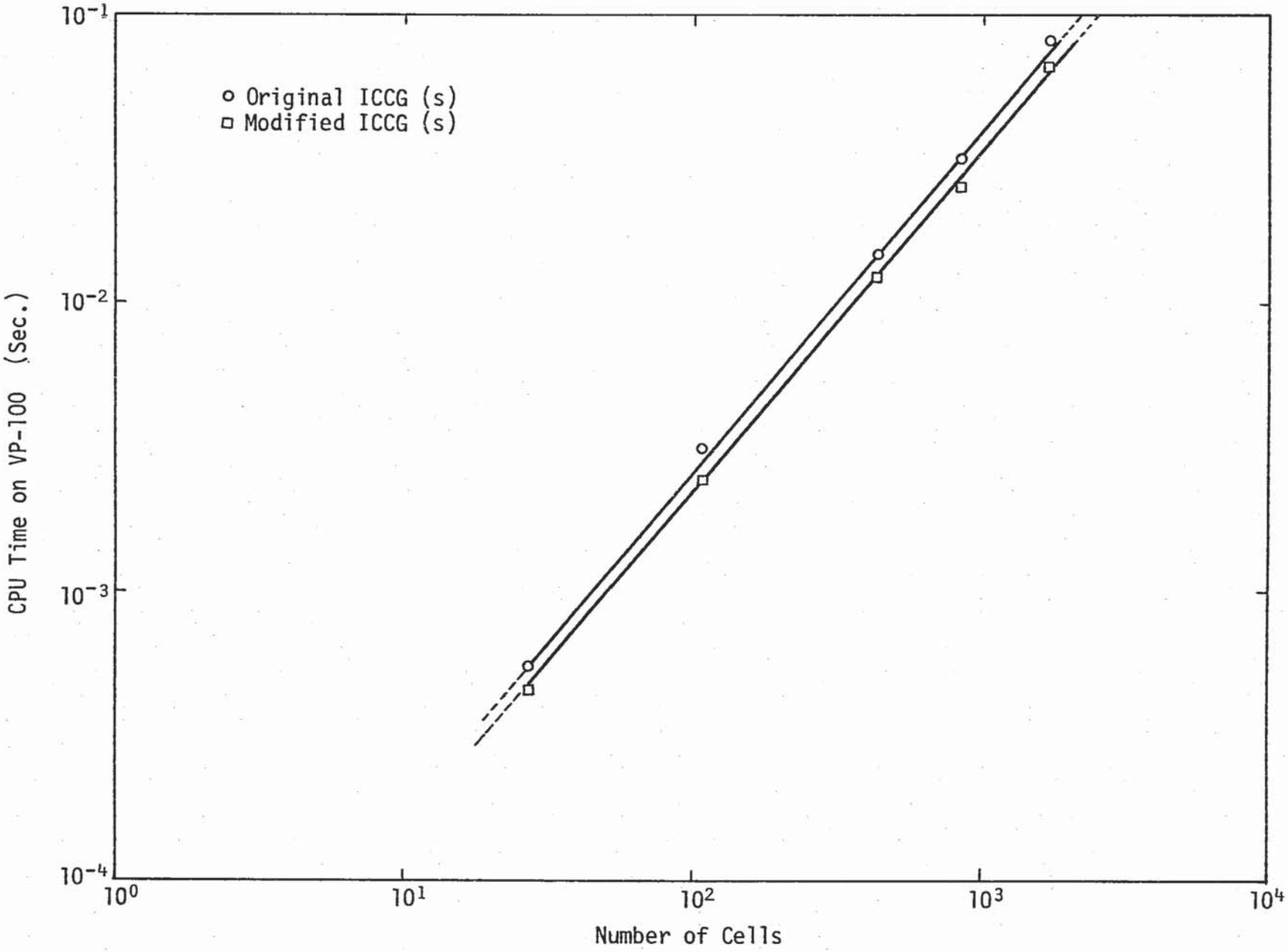


Fig. 3.2 Effect of Improvement for CPU Time of One Iteration Process

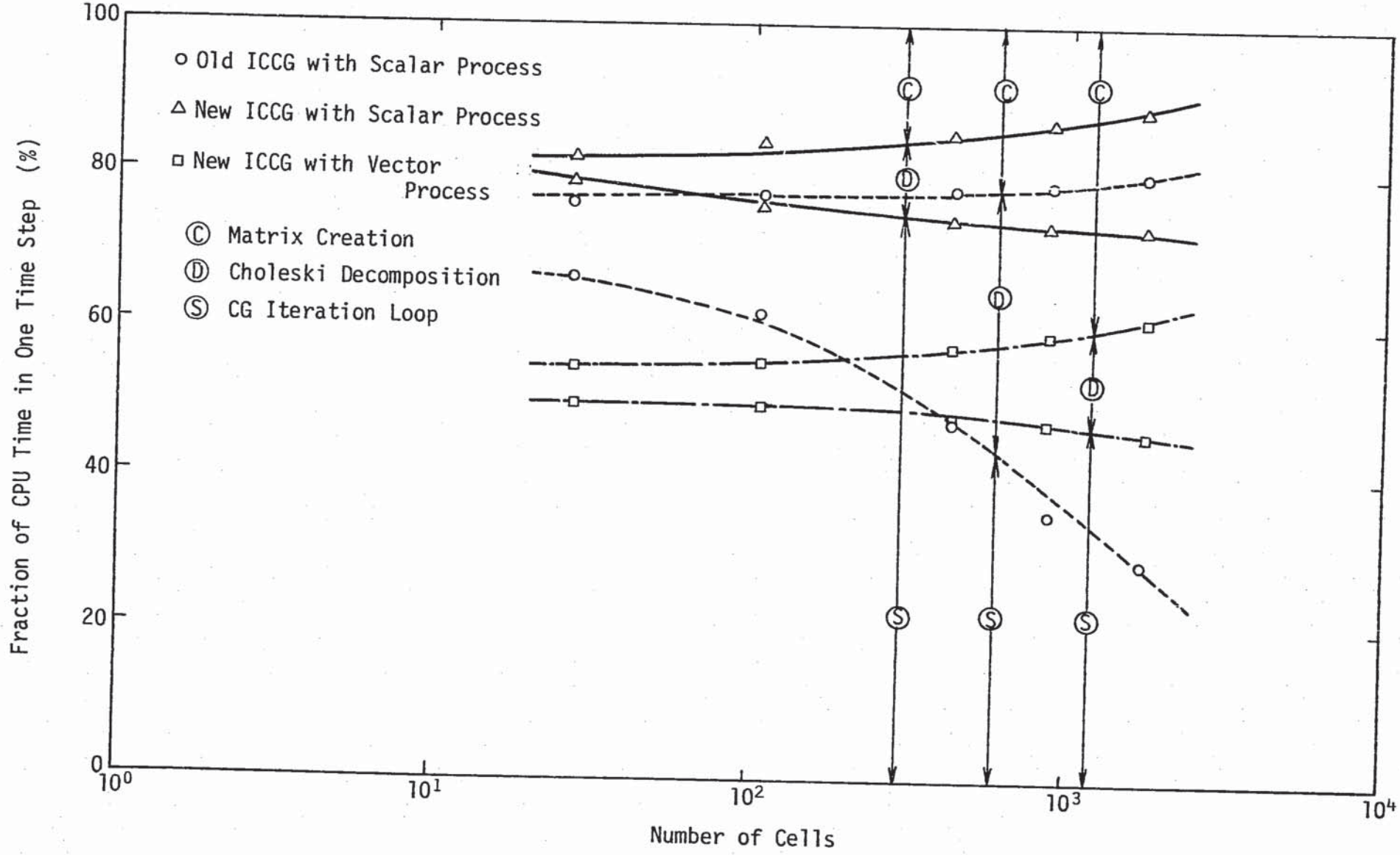


Fig. 3.3 Fraction of Matrix Creation, Choleski Decomposition and CG Iteration Loop of CPU Time in One Time Step

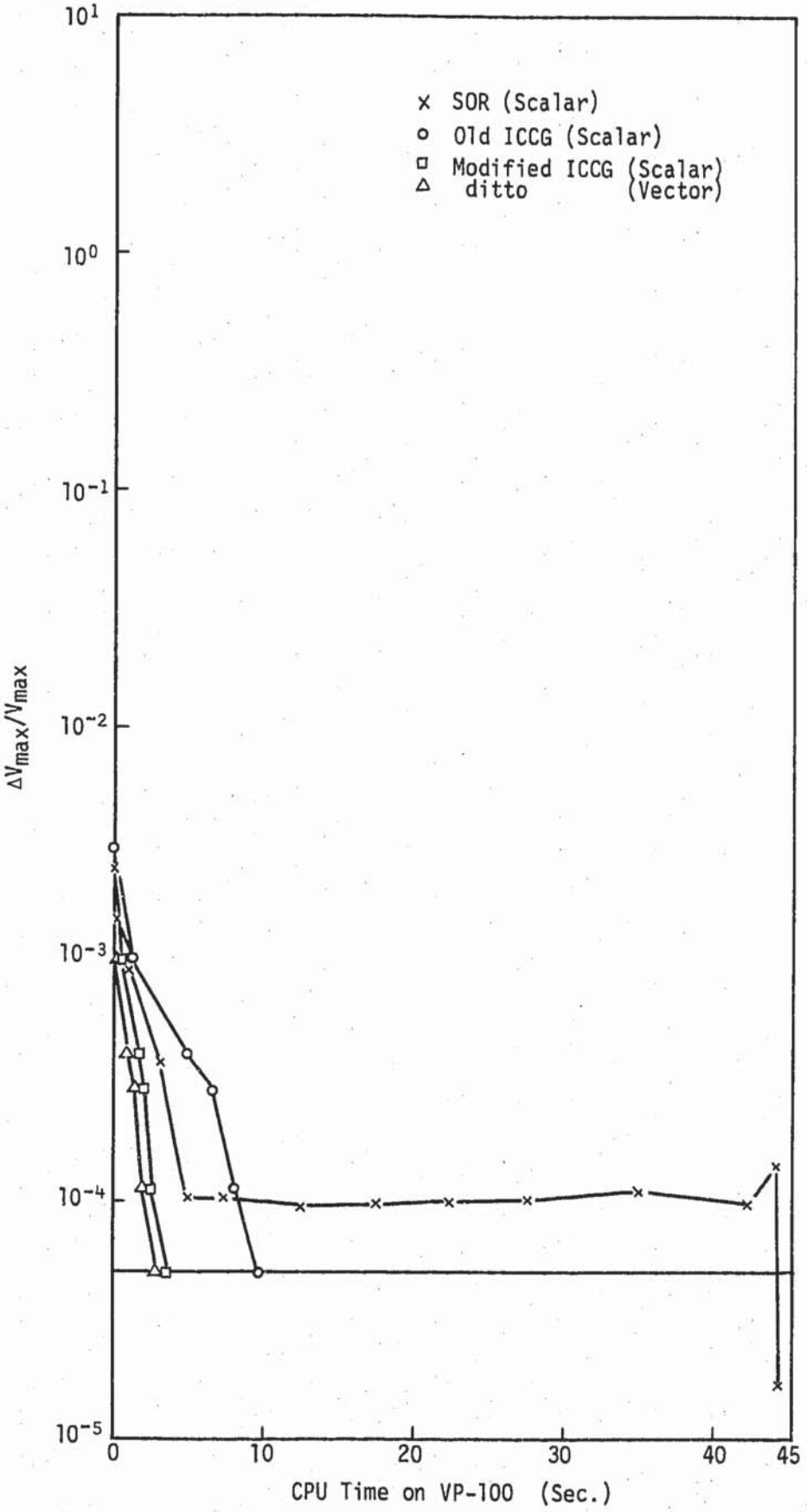


Fig. 3.4 Effect of Improvement for Convergency on 27 cells Problem

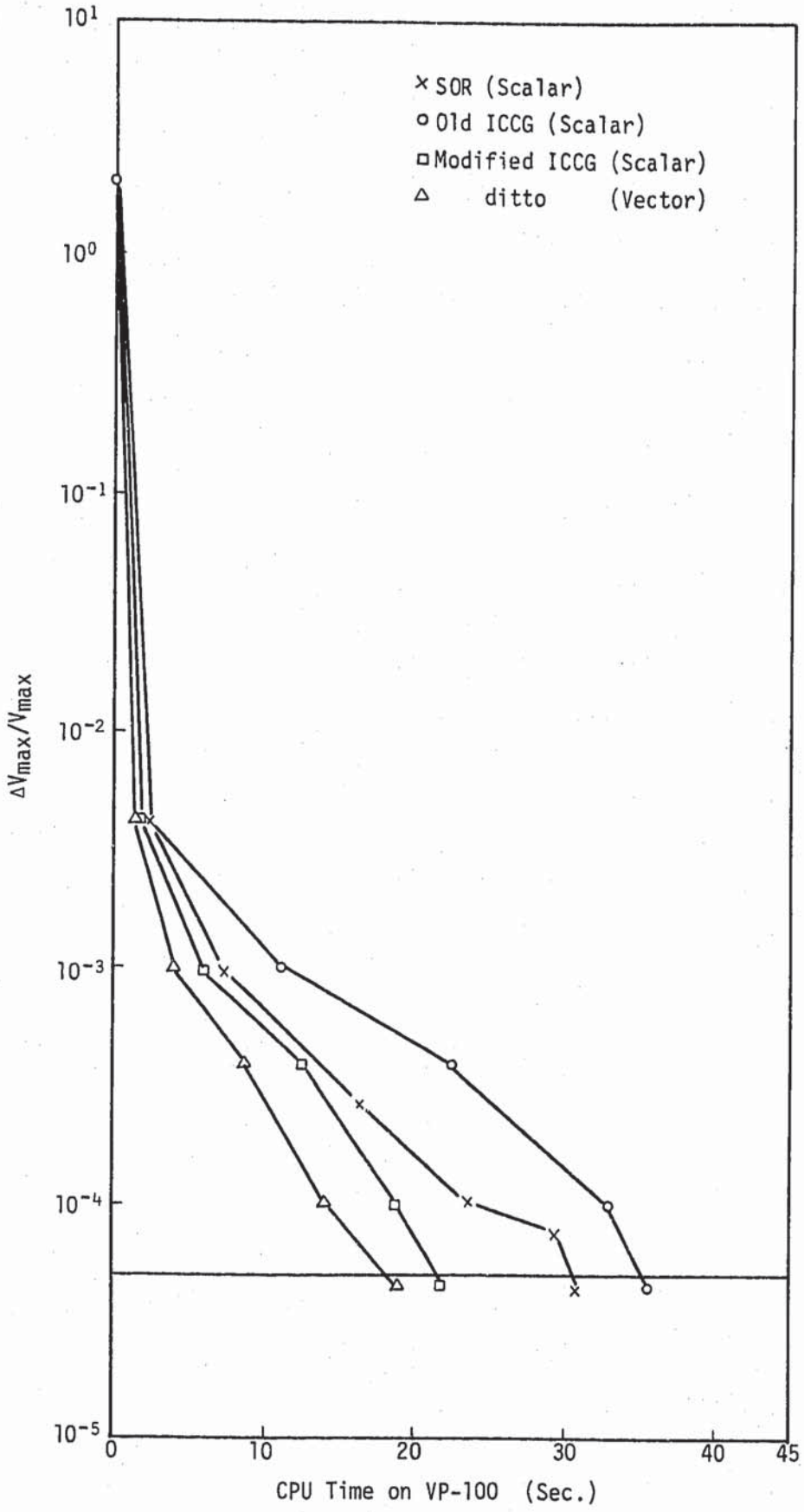


Fig. 3.5 Effect of Improvement for Convergency on 106 Cells Problem

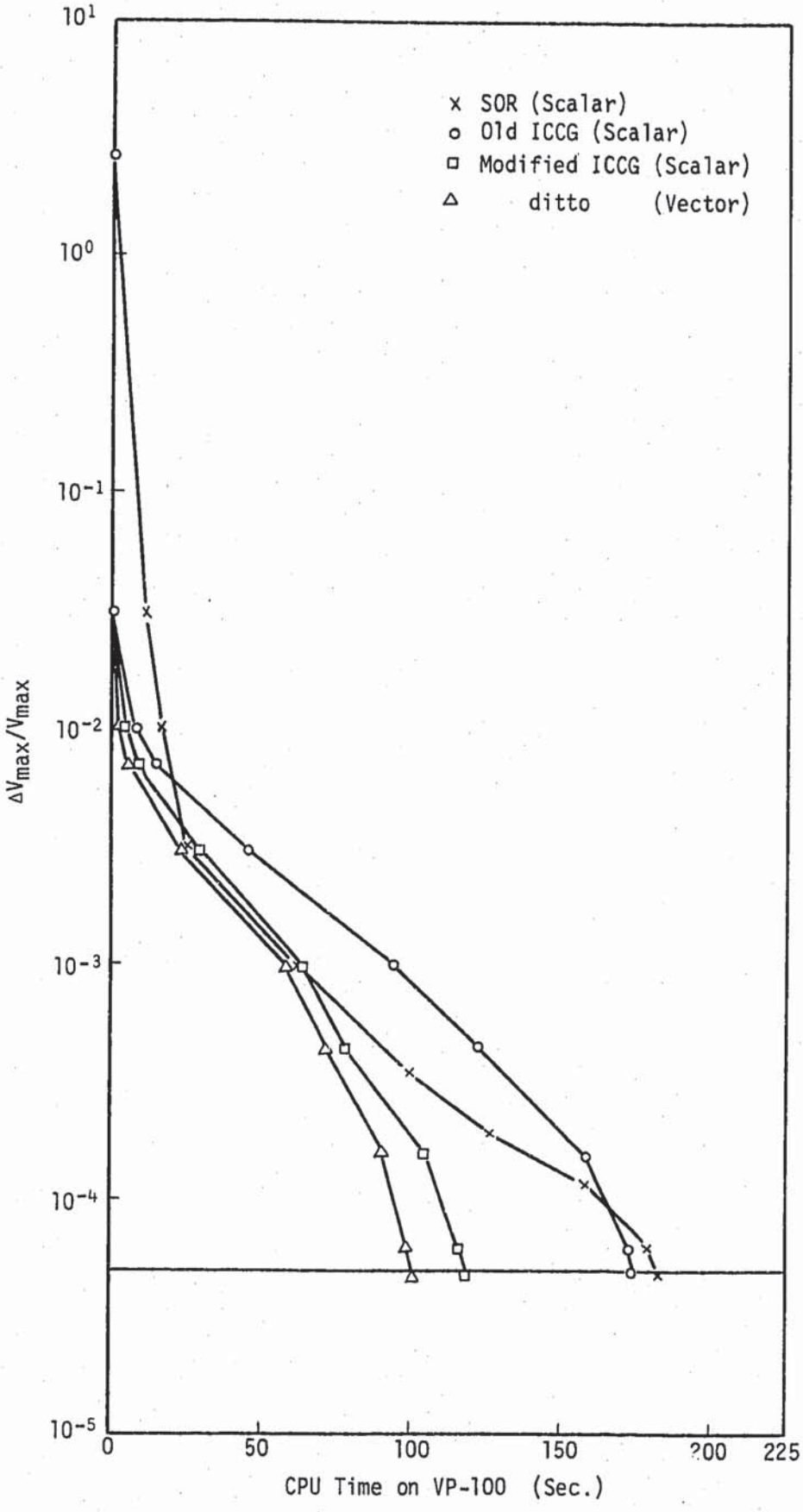


Fig. 3.6 Effect of Improvement for Convergency on 428 Cells Problem

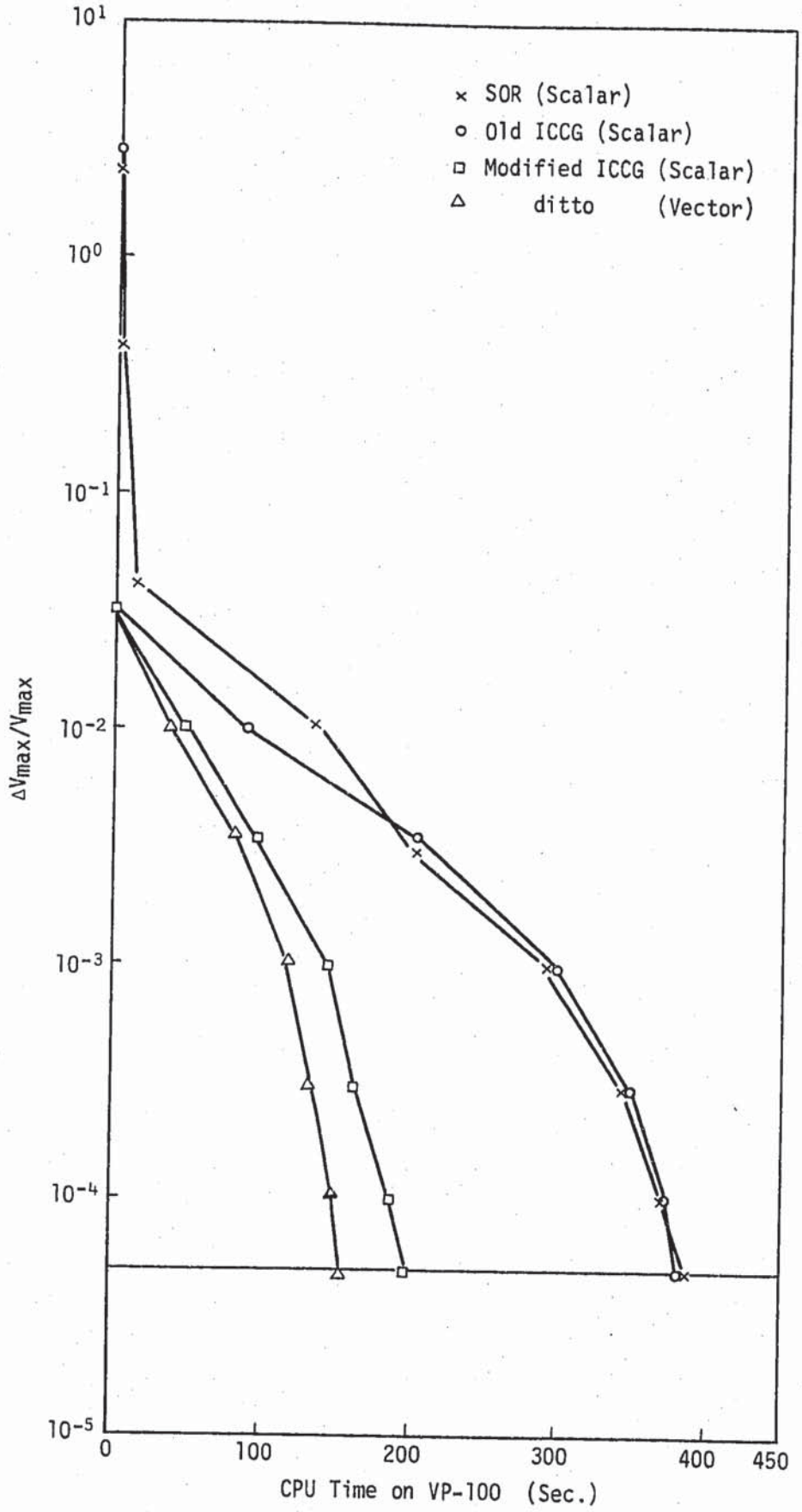


Fig. 3.7 Effect of Improvement for Convergency on 860 Cells Problem

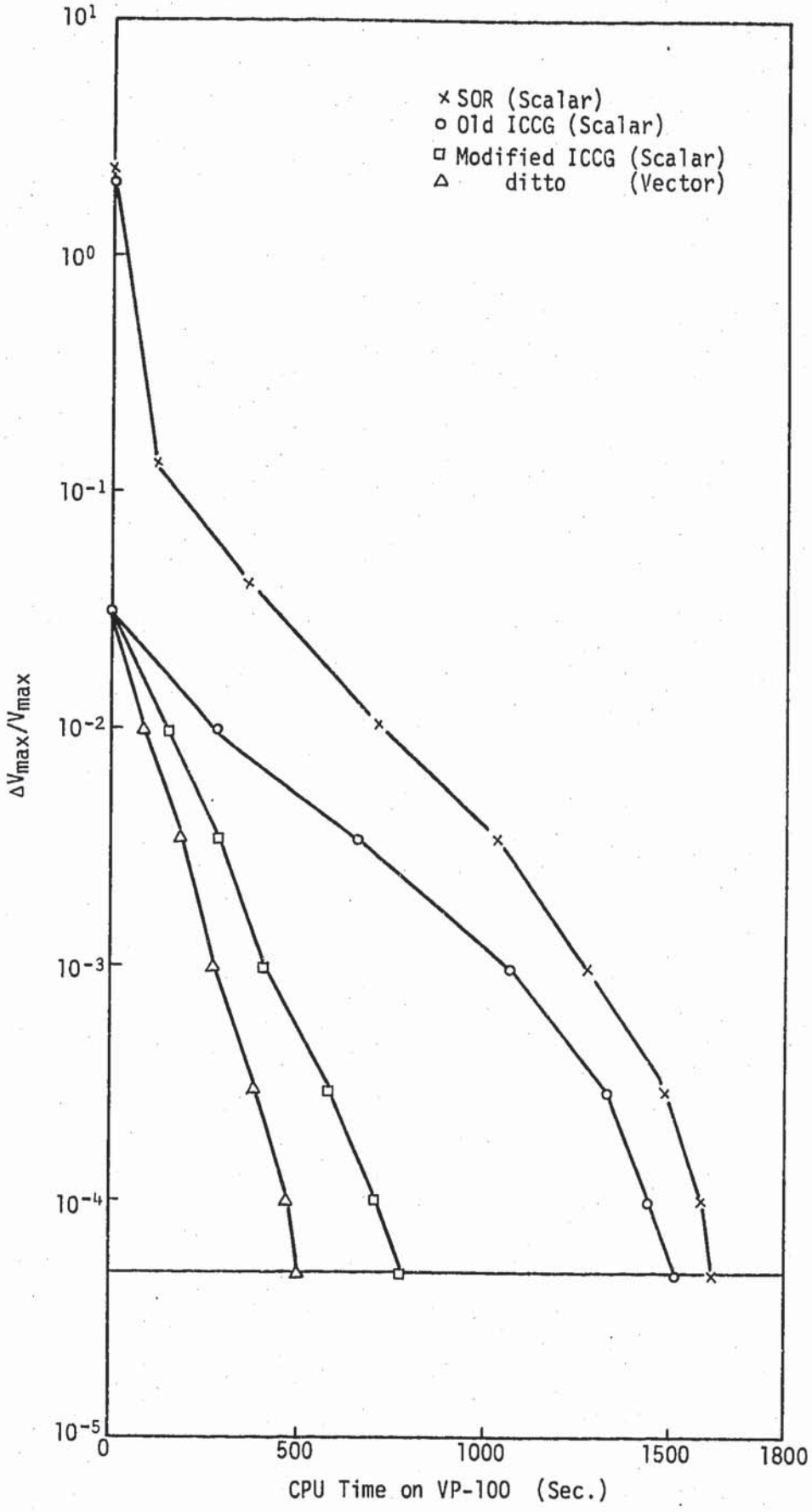


Fig. 3.8 Effect of Improvement for Convergency on 1728 Cells Problem

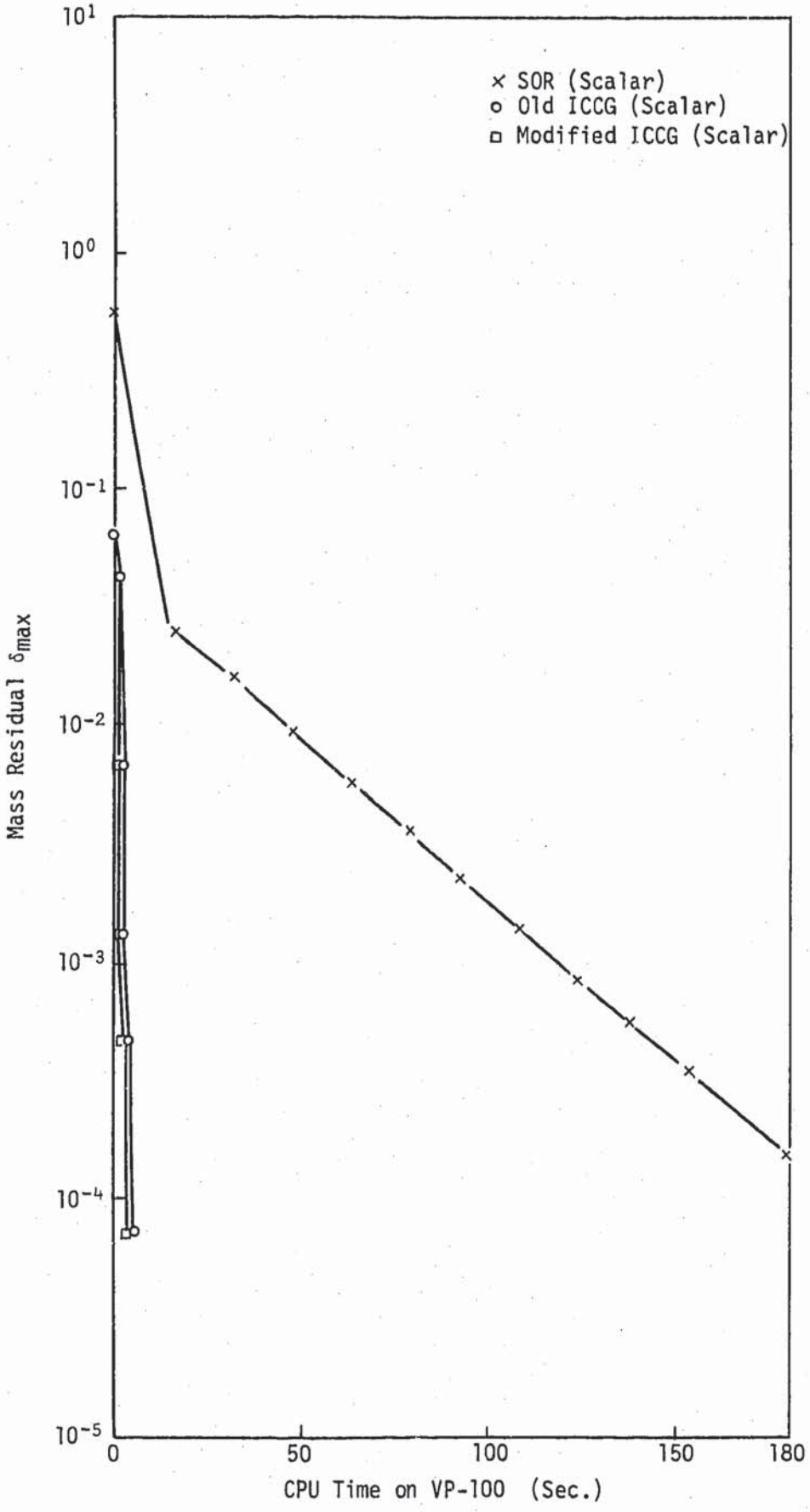


Fig. 3.9 Comparison of Mass Balance Convergency on 1728 Cells Problem (Time Step=1)

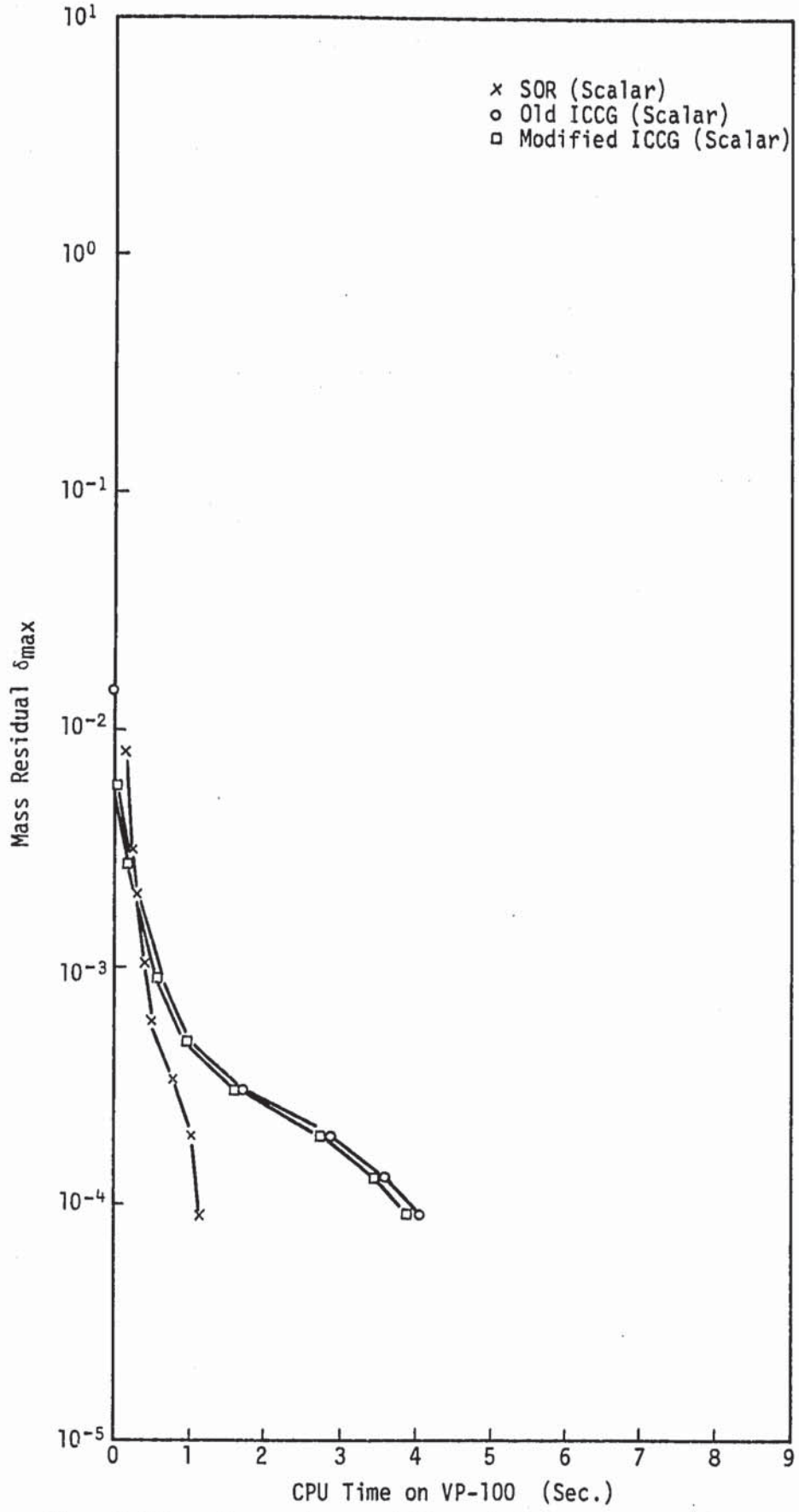


Fig. 3.10 Comparison of Mass Balance Convergency on 1728 Cells Problem (Time Step=720, $\Delta V_{max}/V_{max}=10^{-2}$)

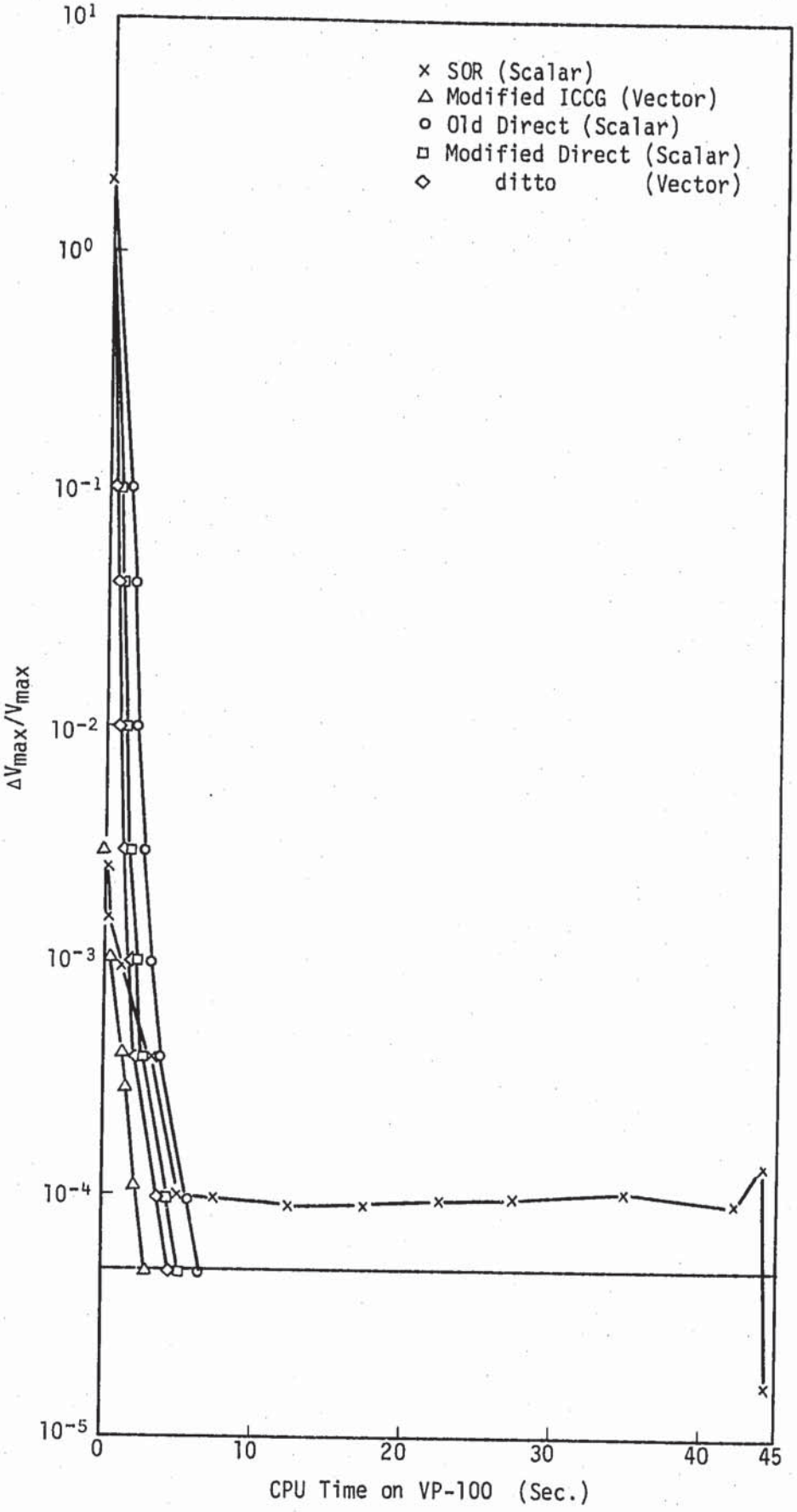


Fig. 3.11 Effect of Improvement for Convergency on 27 Cells Problem

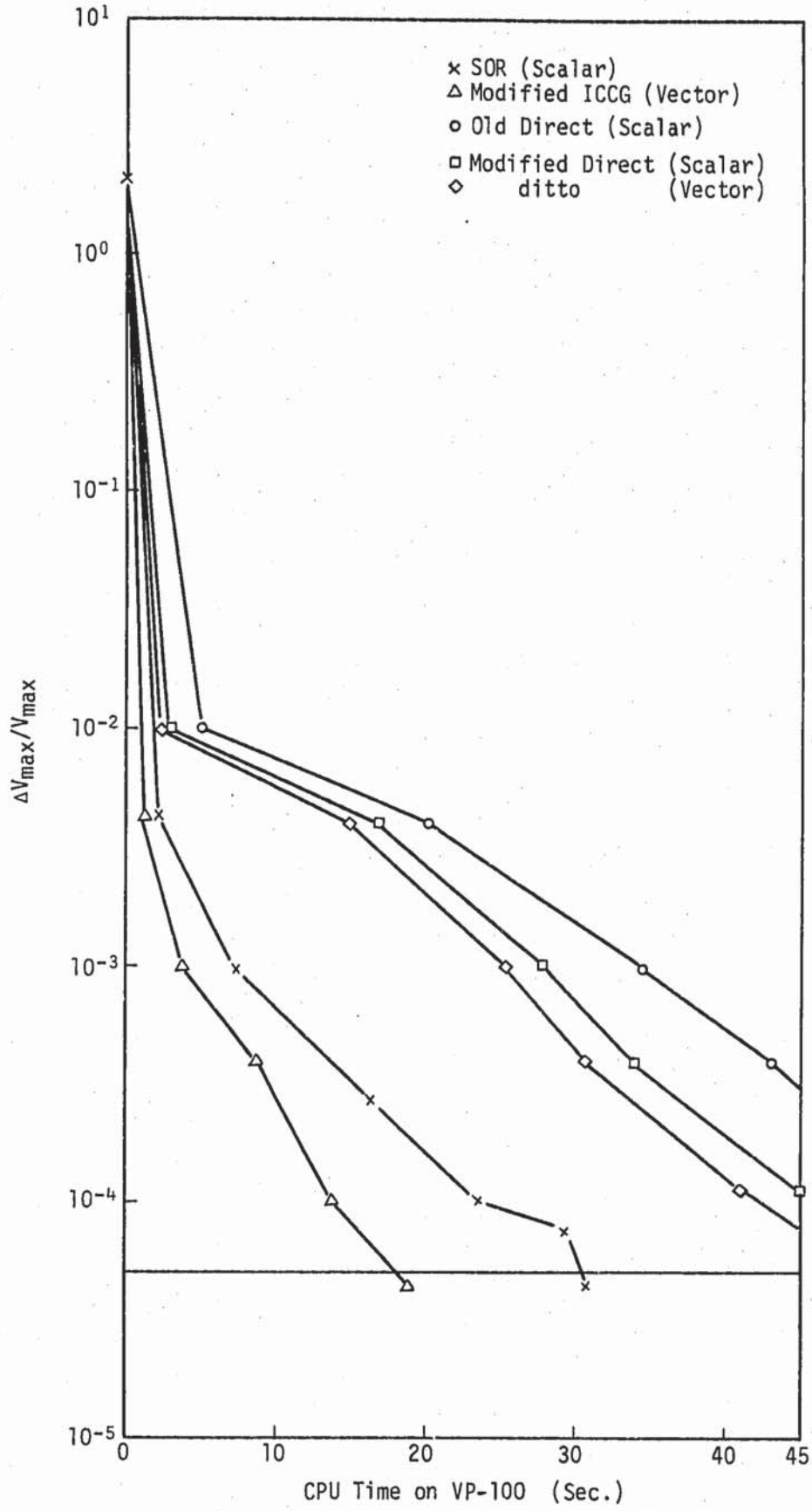


Fig. 3.12 Effect of Improvement for Convergency on 106 Cells Problem

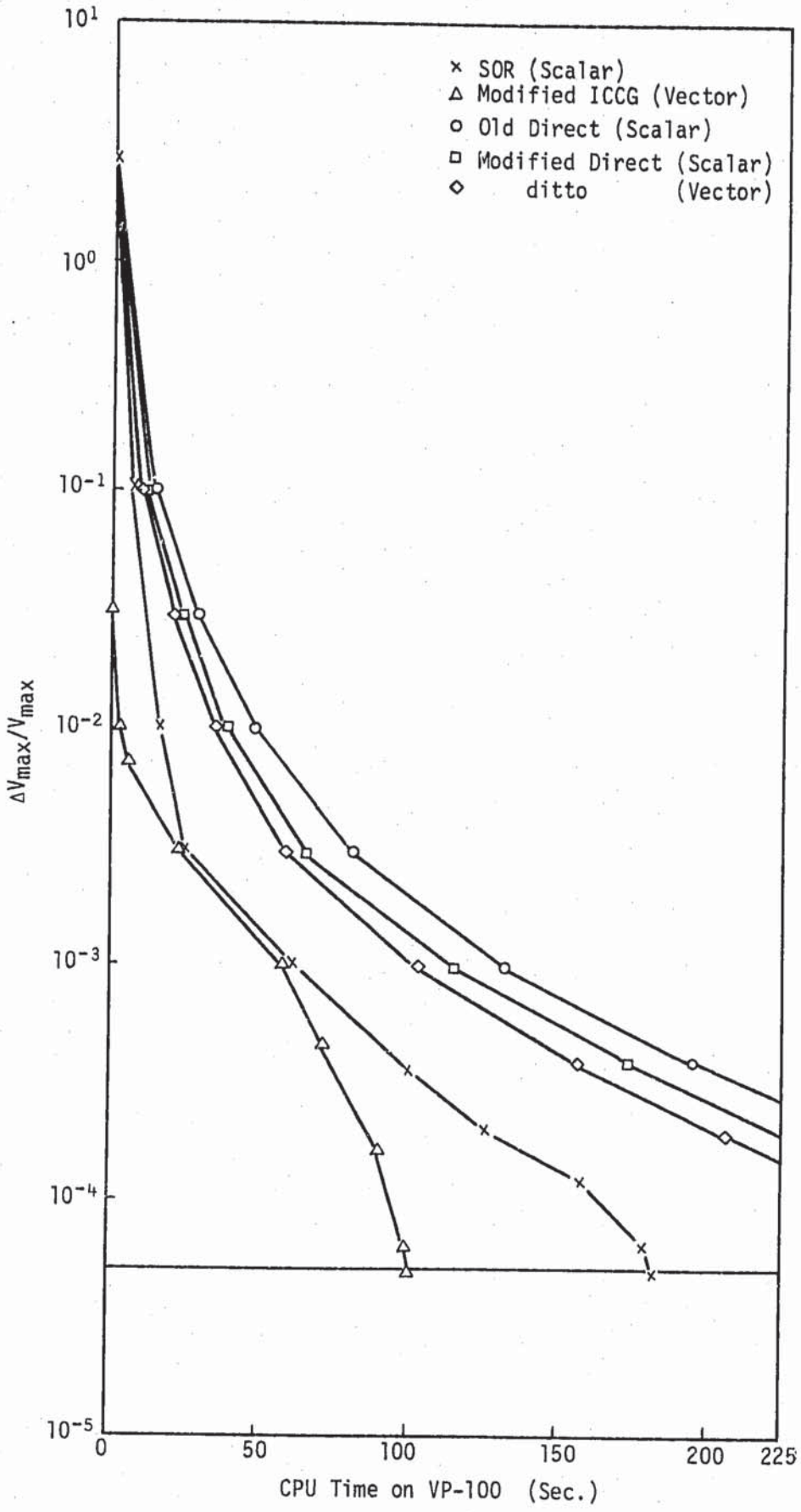


Fig. 3.13 Effect of Improvement for Convergency on 428 Cells Problem

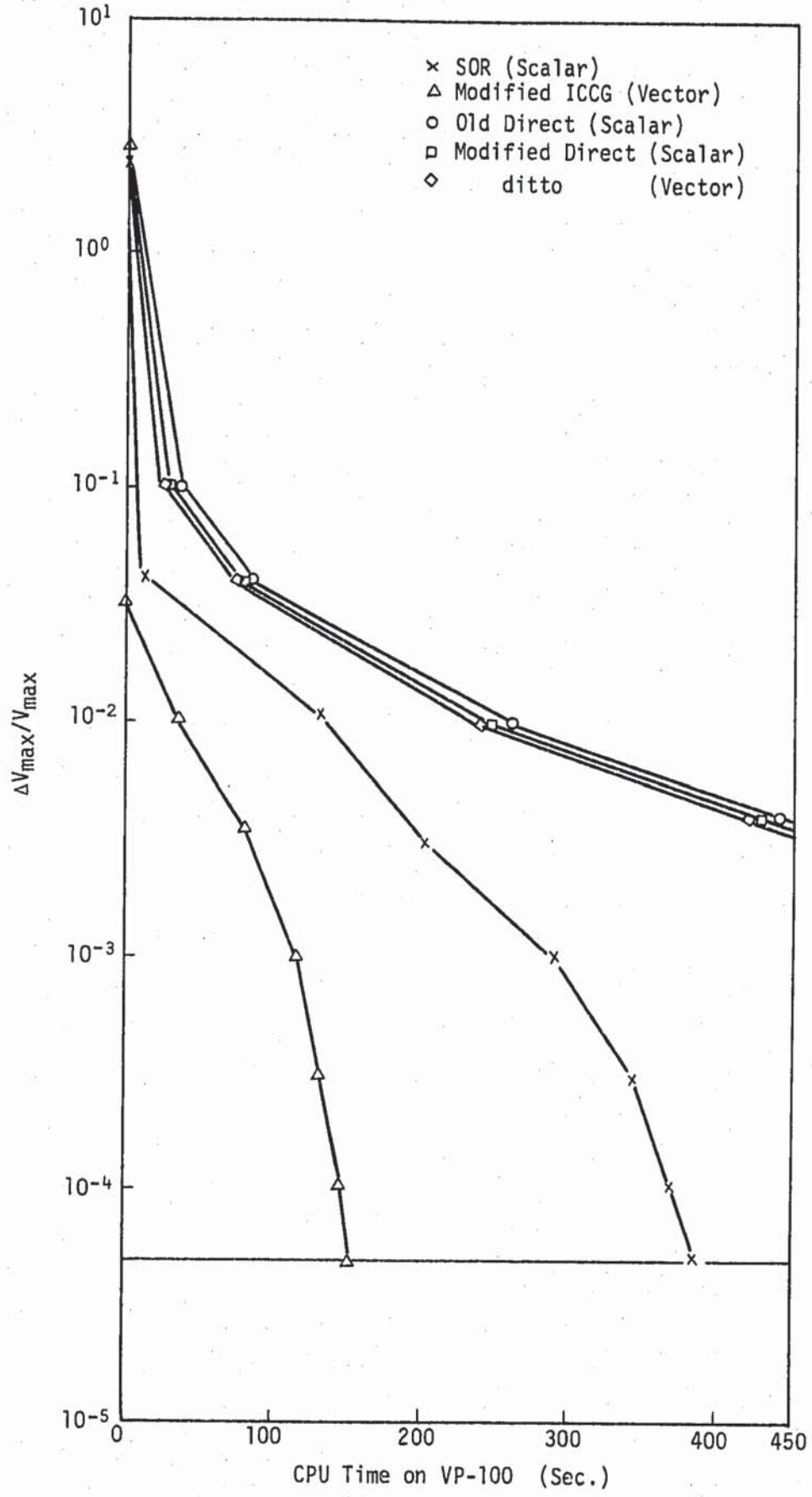


Fig. 3.14 Effect of Improvement for Convergency on 860 Cells Problem

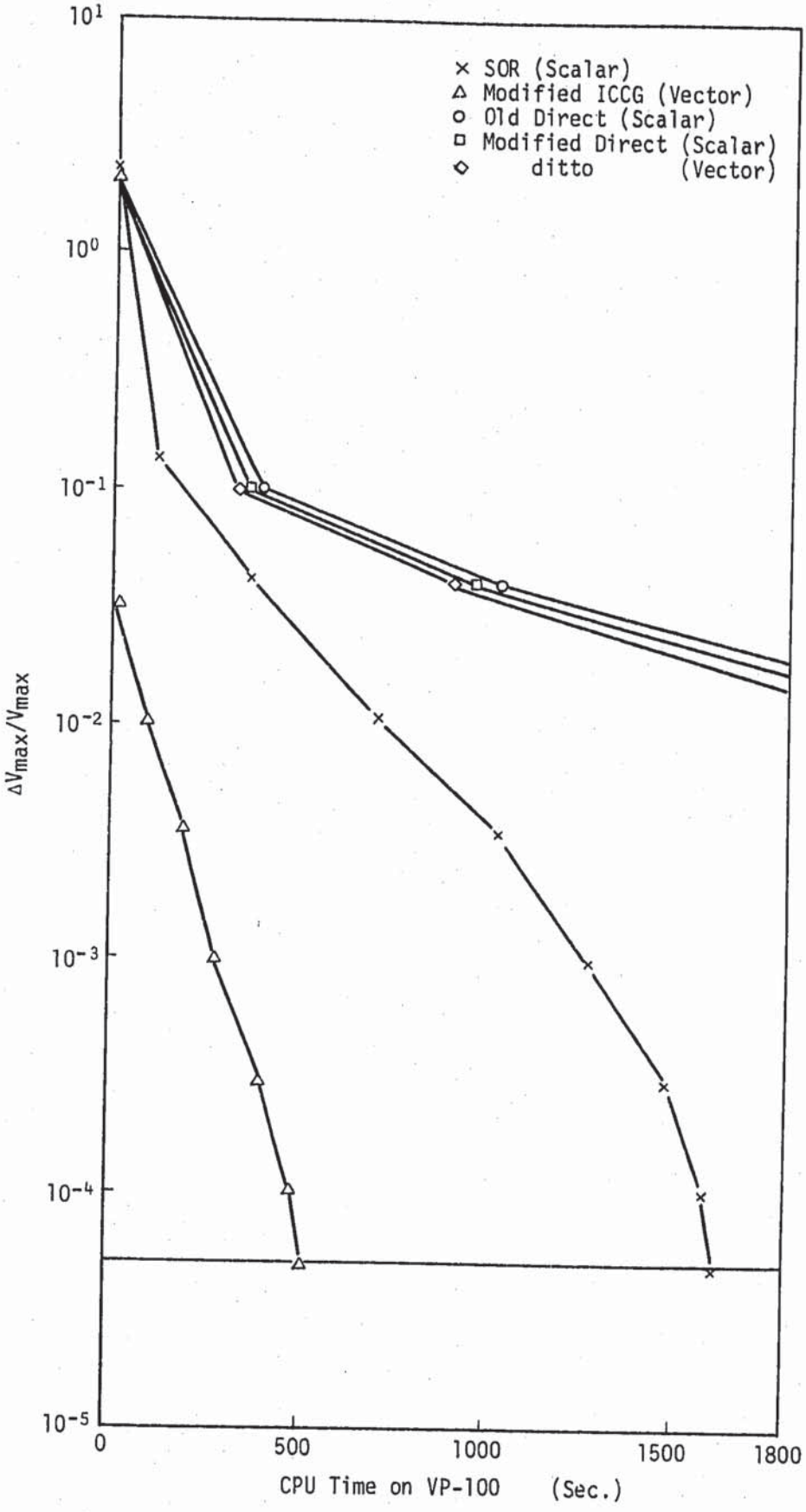


Fig. 3.15 Effect of Improvement for Convergency on 1728 Cells Problem

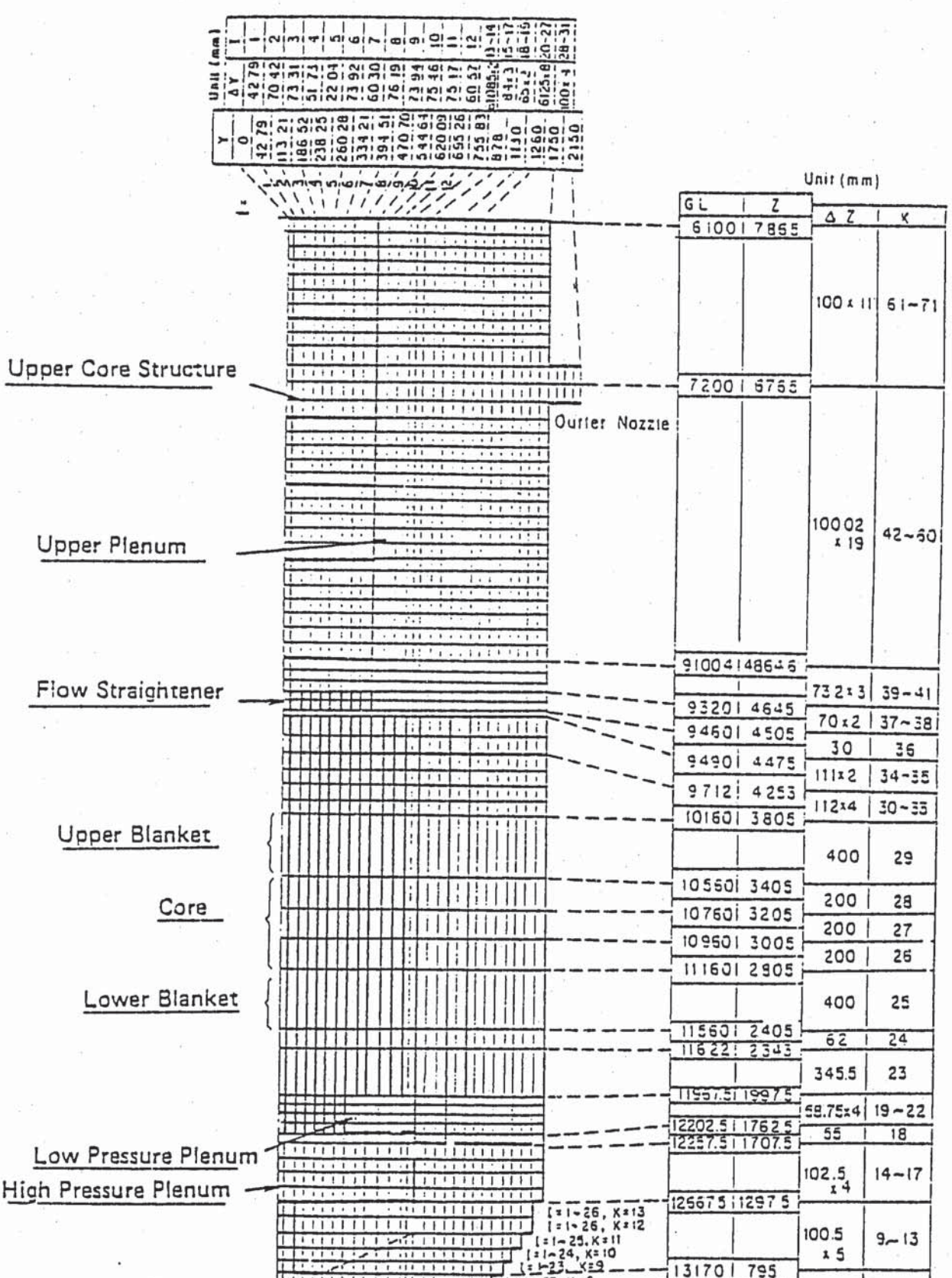


Fig. 4.1 Cell Partitioning for Two-Dimensional Analyses of JOYO MK-I Natural Circulation Test

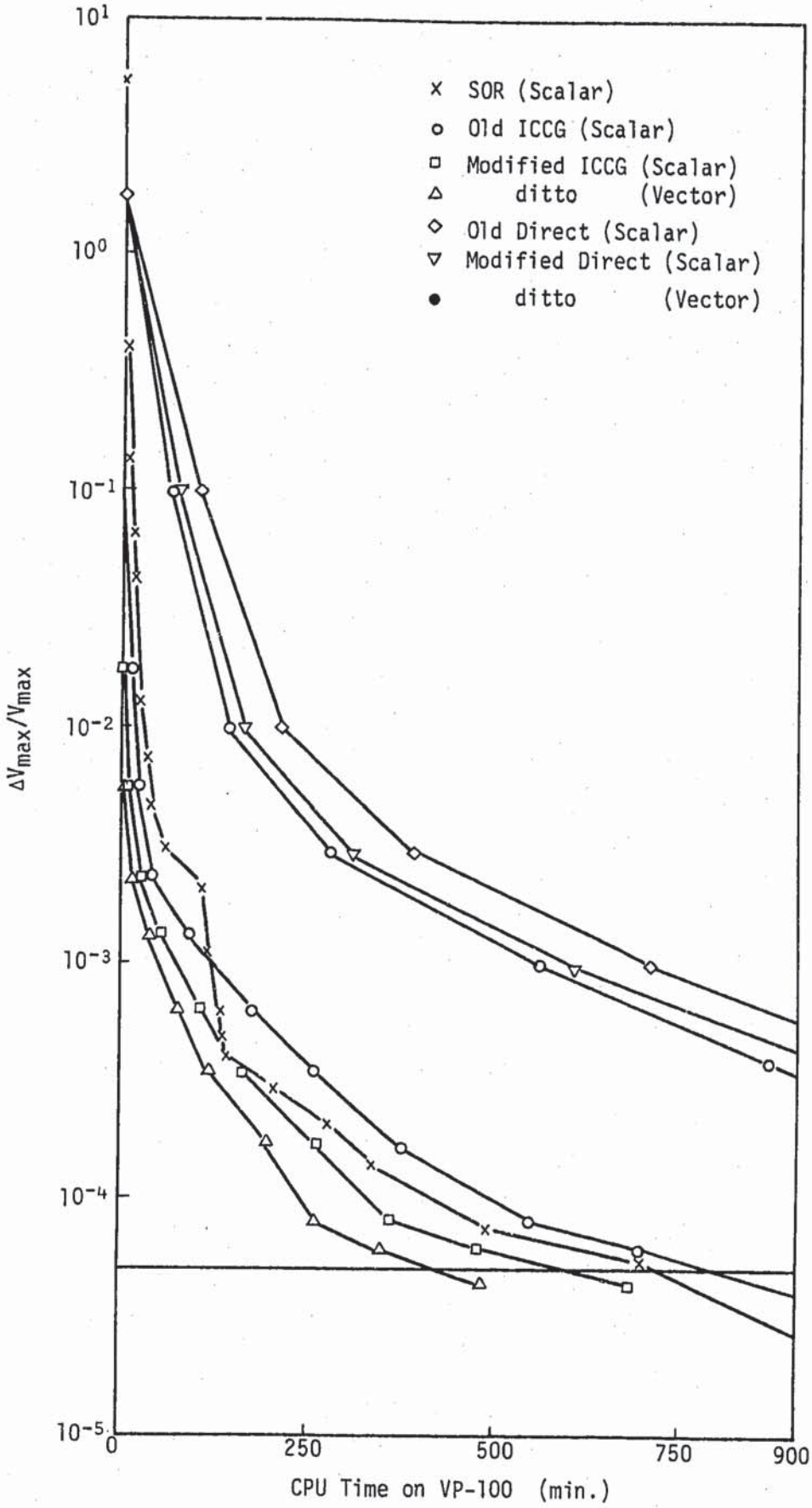


Fig. 4.2 Effect of Improvement for Convergency on Steady-State Run of JOYO Natural Cirulation Test.

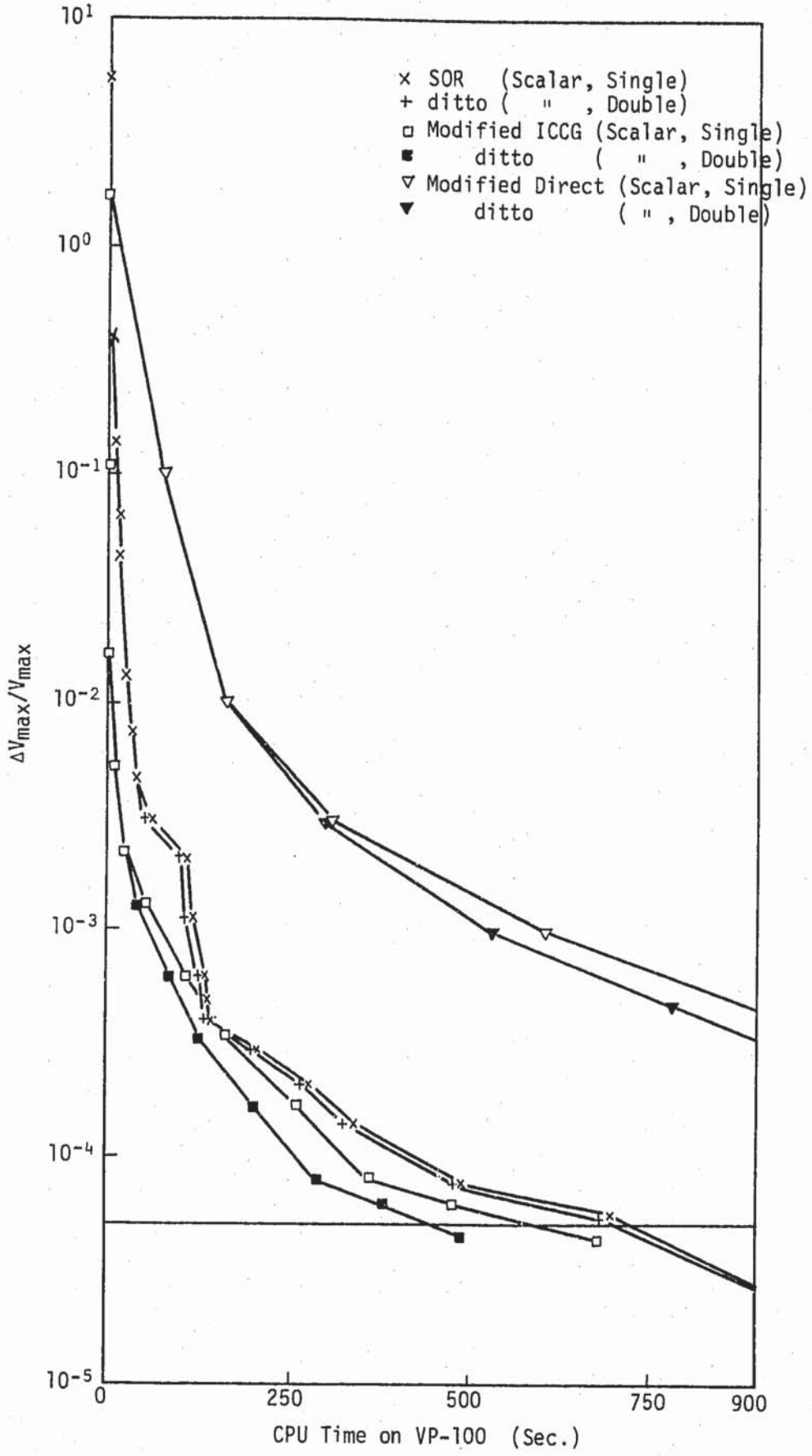


Fig. 4.3 Effect of Double Precision Run for Convergency on Steady-State Calculation of JOYO Natural Circulation Test

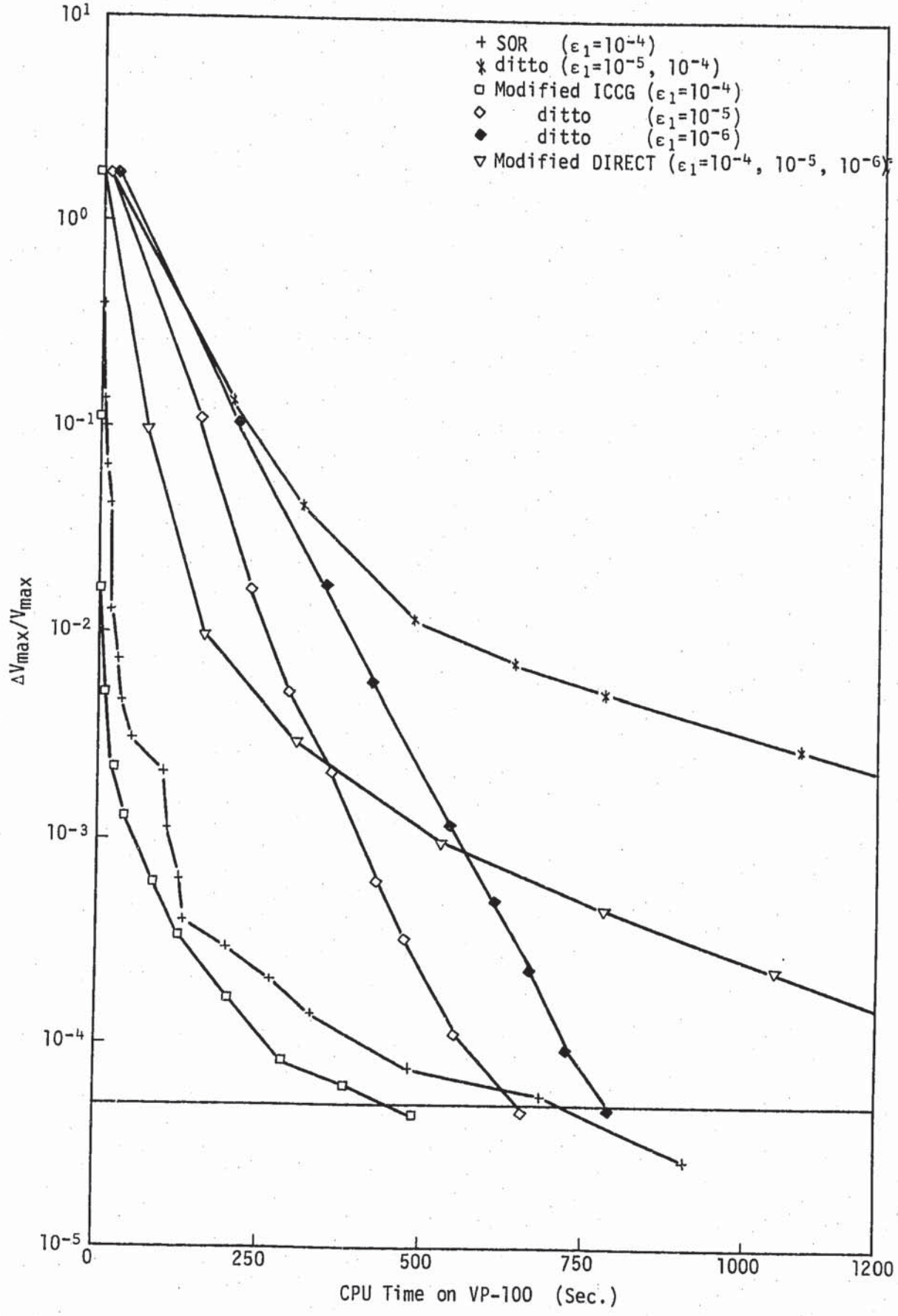


Fig. 4.4 Effect of Mass Balance Criteria for Convergency on Steady-State Calculation of JOYO Natural Circulation Test

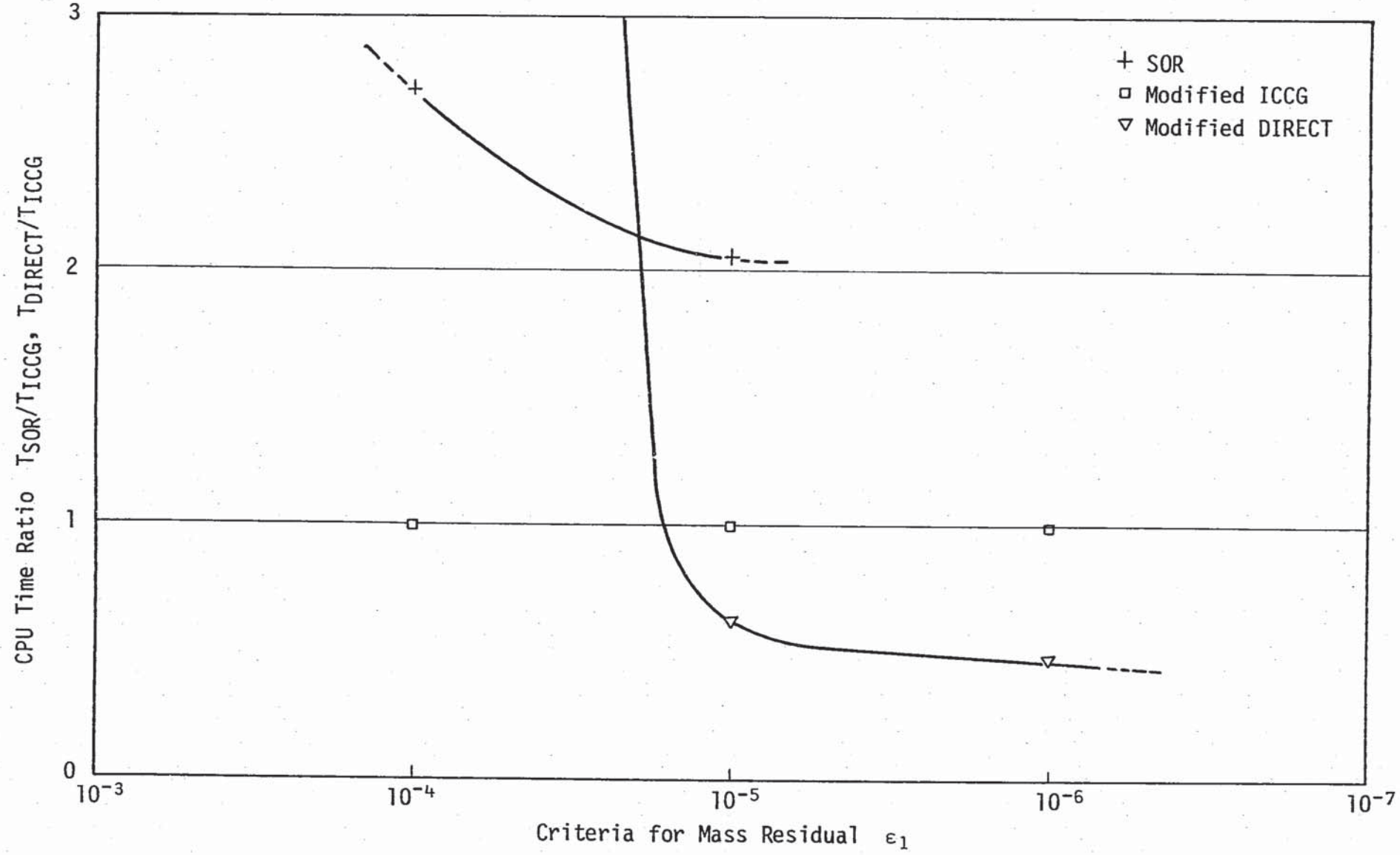


Fig. 4.5 Comparison of CPU Time Ratio under Various Criterias for Mass Residual ε_1 on Steady-State Calculation of JOYO Natural Circulation Test. ($\Delta V_{max}/V_{max}=10^{-2}$)

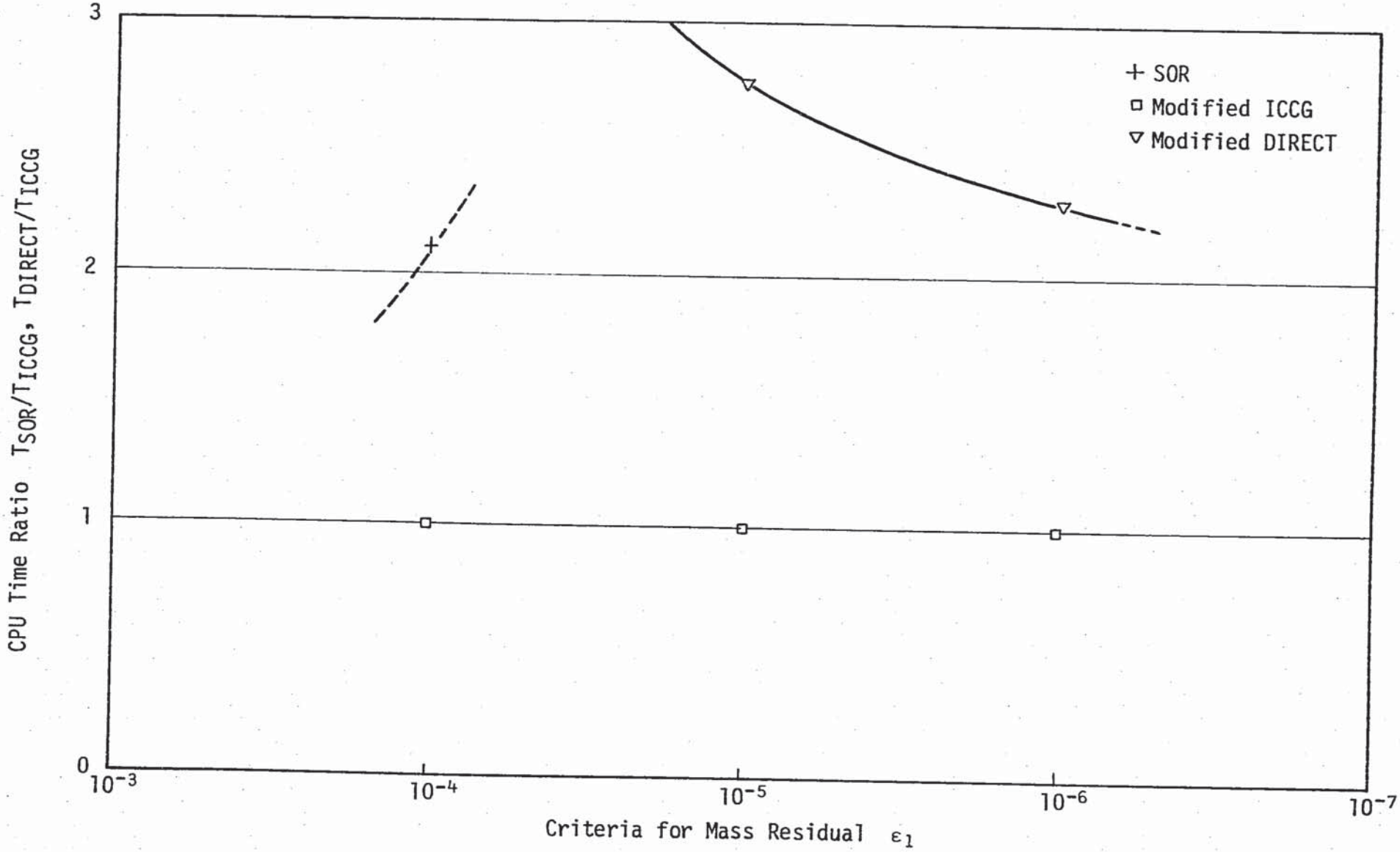


Fig. 4.6 Comparison of CPU Time Ratio under Various Criterias for Mass Residual ε_1 on Steady-State Calculation of JOYO Natural Circulation Test. ($\Delta V_{max}/V_{max}=5\times10^{-5}$)

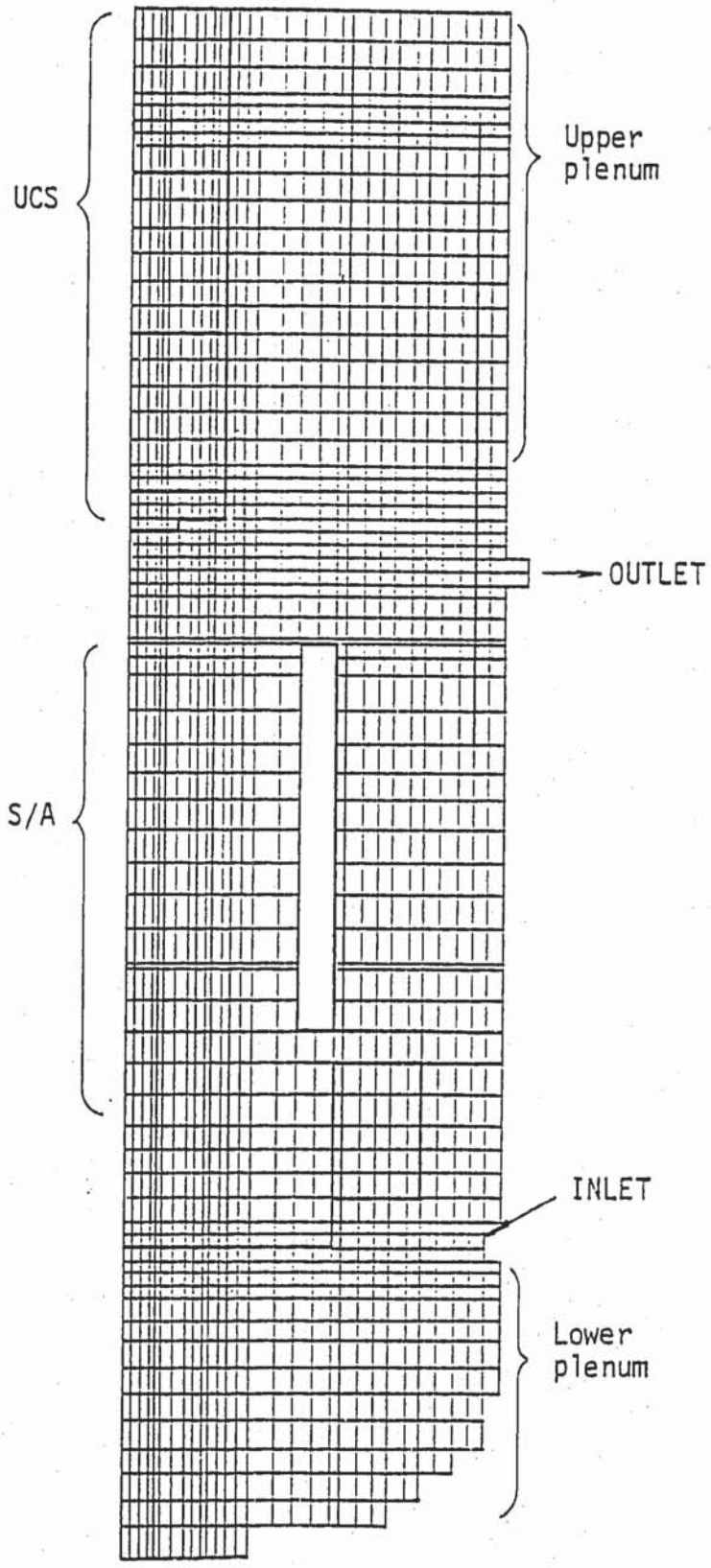


Fig. 4.7 Mesh Arrangement for MONJU PLOHS Analysis

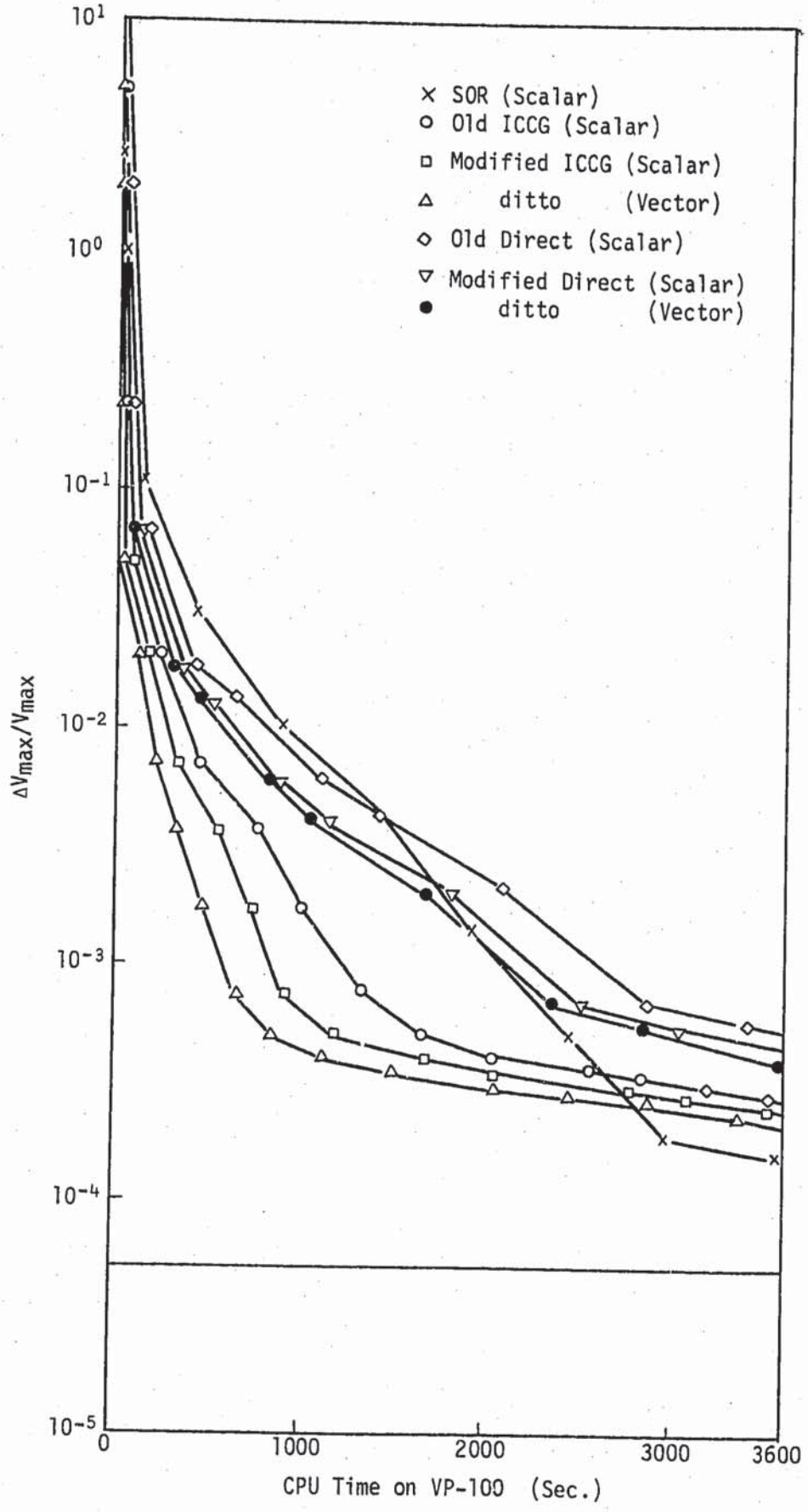


Fig. 4.8 Effect of Improvement for Convergency on Steady-State Run of MONJU PLOHS Analysis

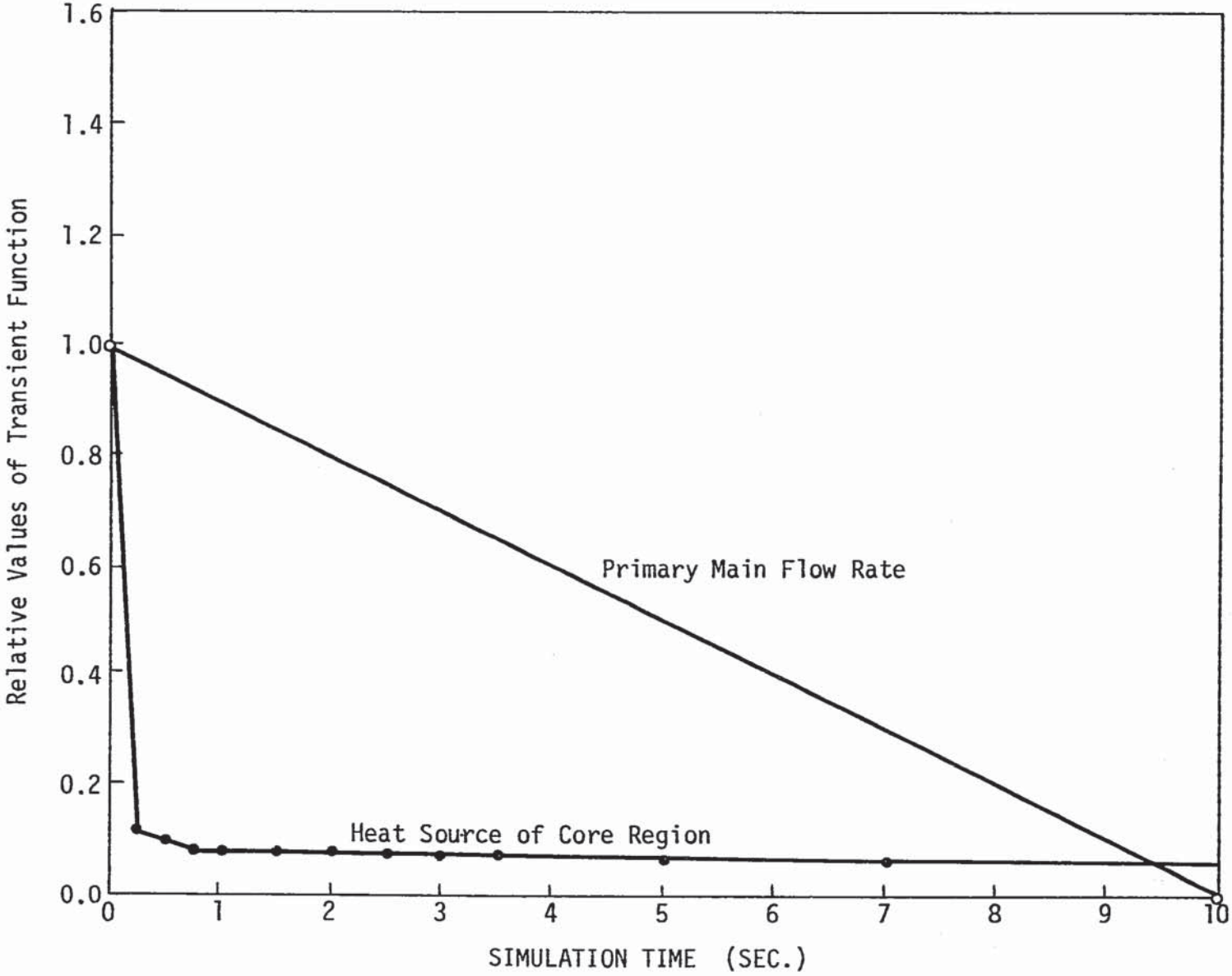


Fig. 4.9 Flow Rate and Heat Source Transient on MONJU PLOHS Analysis

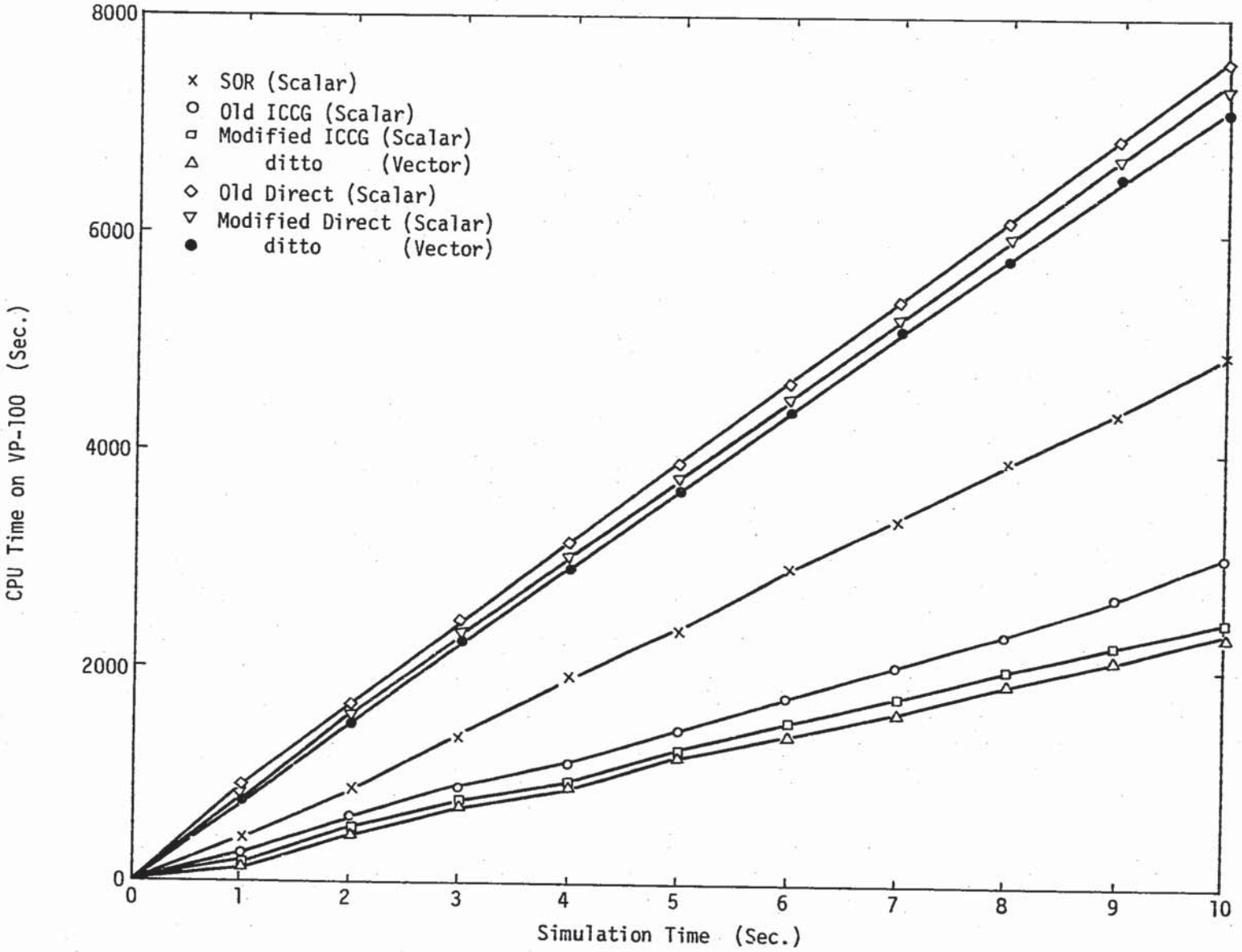


Fig. 4.10 Comparison of CPU Time for Each Solver on MONJU PLOHS Transient Analyses

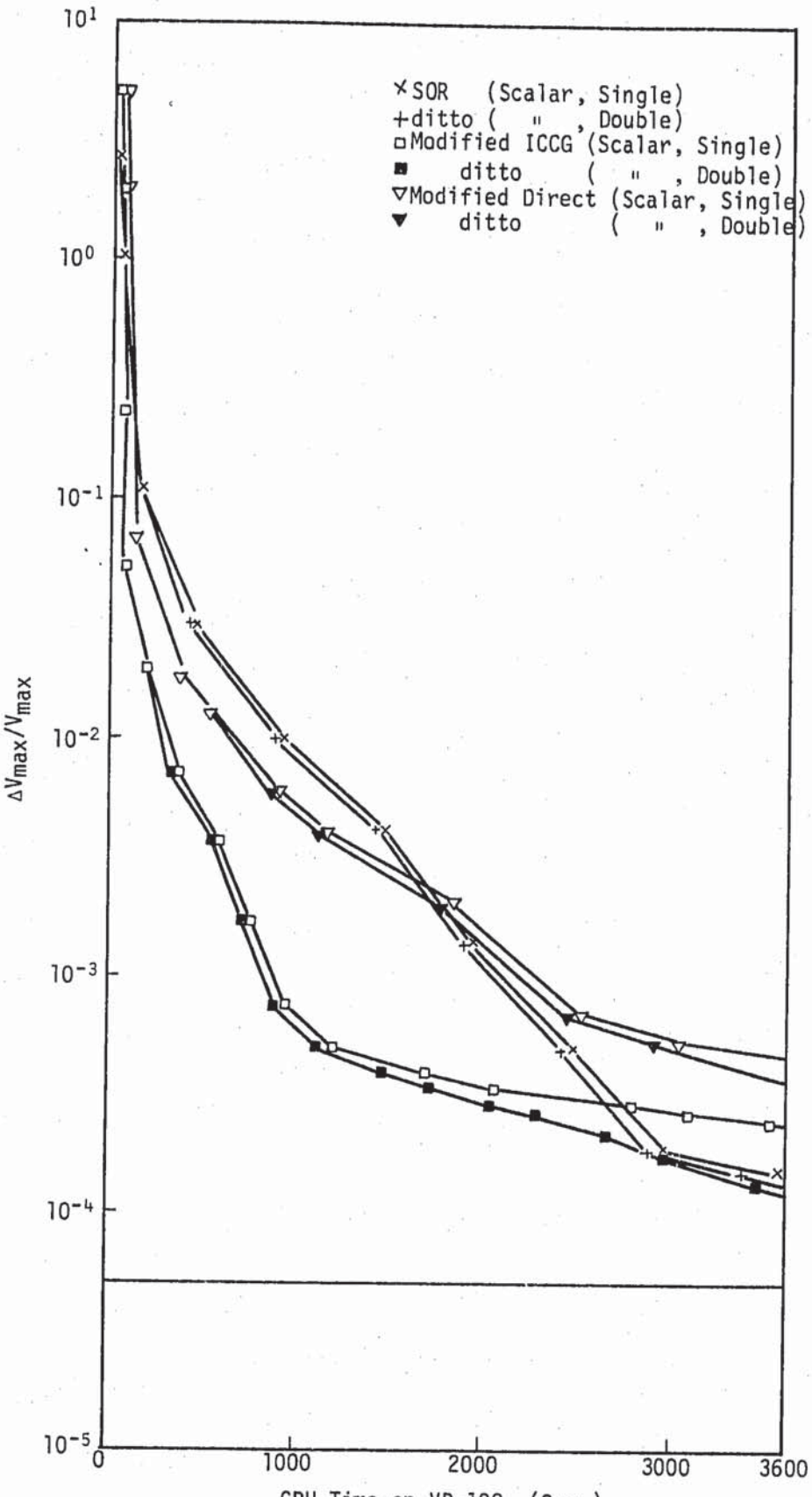


Fig. 4.11 Effect of Double Precision Run for Convergency on Steady-State Calculation of MONJU PLOHS Analysis

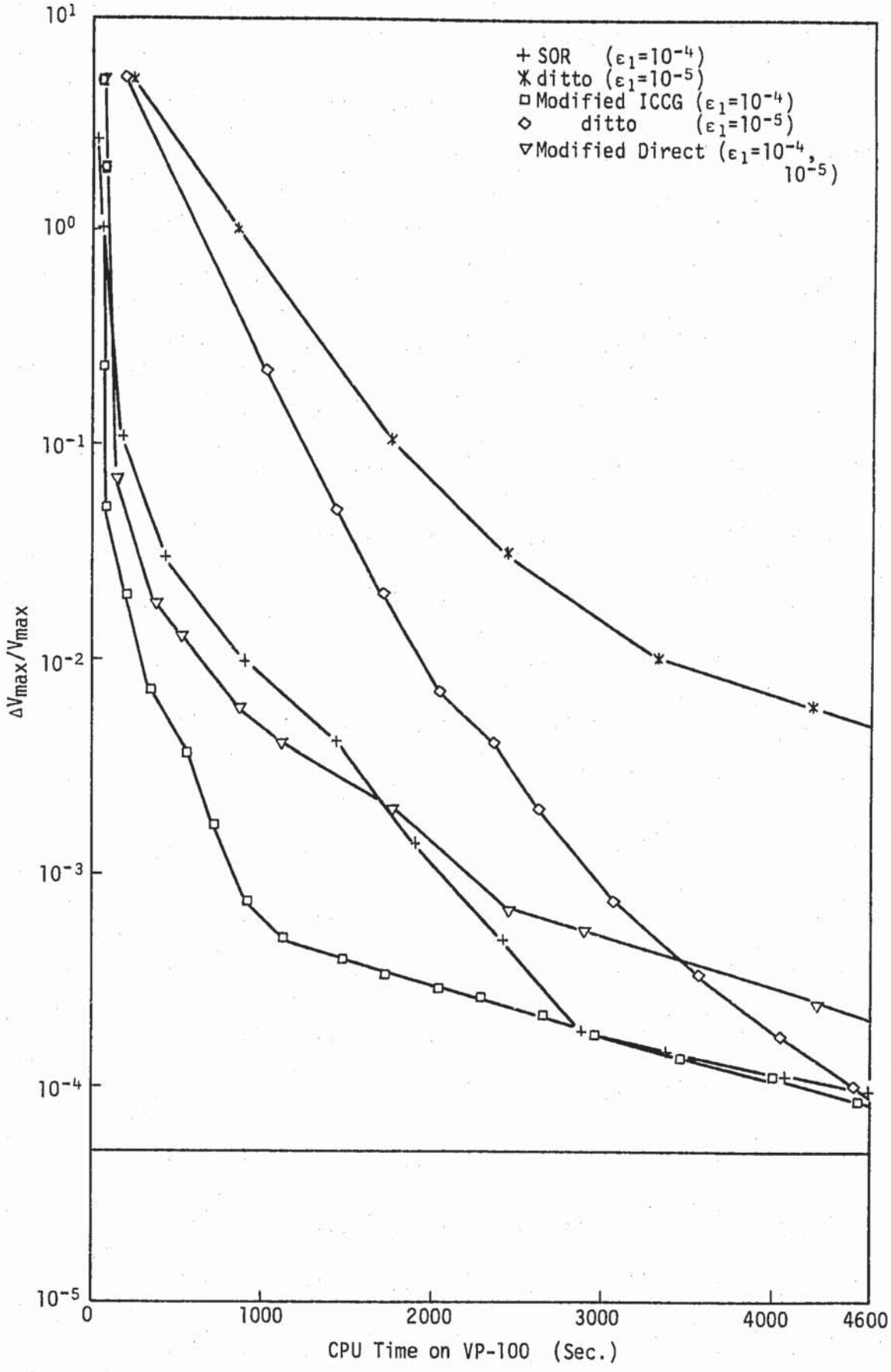


Fig. 4.12 Effect of Mass Balance Criteria for Convergency on Steady-State Calculation of MONJU PLOHS Analysis