

一般曲線座標系における反復解法 (SOR法)の高速化について

藤野清次 (計算流体力学研究所) (Fujino Seiji)

田村哲郎 (清水建設大崎研究室) (Tamura Tetsurou)

森 正武 (東京大学 工学部) (Mori Masatake)

1. はじめに

スーパーコンピュータの最高性能を発揮させるには、

(1) 出来るだけ長いベクトル長の配列を確保すること

(2) 再帰的な演算を出来るだけ排除すること

(3) メモリーへのデータの参照・格納を連続的にすること

などが特に重要である。更に出来上がったプログラムの汎用性やわかり易さ等も考慮に入れると

(4) 配列のアドレス計算が簡明であること

等も効率を上げるためには必要になってくる。また数値解法が本来持つ精度や収束性自身をベクトル化等のために落とさないことも非常に重要である。

これら基本的な原則からはずれた計算技法を使っている場合、その計算技法の理論面及びプログラミング両面から検討が必要になってくる。

またスーパーコンピュータは機種ごとの「個性」が強く、その計算技法の有効性は機種によって大きな差が生じることがある。したがってより汎用性のある計算技法の開発が望ましい。

本論文では流体解析等ですでに確立されている一般曲線座標系で現れる連立1次方程式について考えることにする。一般曲線座標系とは任意形状の物体に沿って計算格子を張った物理空間から計算空間に支配方程式を写像して計算を行う方法の一つである。通常の直交座標系に比べて一般曲線座標系では交差微分項が現れ、離散化のための点数が増加する。更に例えばポアソン方程式から得られる連立1次方程式の係数行列なども対称性が崩れ非対称行列になる。したがってよく研究されている直交座標系の場合とはともかく、一般曲線座標系においてもそこで現れる大規模スパース連立1次方程式の反復解法(ここではSOR法を取り上げる)の効率化について研究することは十分意義があると考えられる。

2. MULTI-COLOR技法によるSOR法の高速化

(2-1) 計算順序の並替え

よく用いられているSOR法はそのままの計算順序ではベクトル化が出来ないことはよく知られている。そこで色々なベクトル化技法が提案されてきた。上で述べた基本原則(1)を主眼に考え出された擬似SOR法[1]や、(2)を中

心に考え出されたhyper-plane法[2]や、直交座標系でのRed-Black法SOR法[3]などがある。これらは本来のSOR法の計算の順序を各々の目的に合わせて並替えて計算を進めるものと言える。この並替えを行ってもSOR法本来の収束性は変わらないことは理論的に証明されている。[4],[5]そこで得られた結論を要約すると以下ようになる。

(a)SOR法は種々の並び替えをおこなっても最適な加速係数は等しい。

(b)僅かに生ずる収束状況の違いは、初期残差ベクトルの固有ベクトル成分が異なった分布をするために起こる。

これらの研究の成果から、ベクトル計算機用の効率を重視した並替えを行なっても数学的な収束率を損なうことは無いとの裏付けが得られたので、今後は安心して並替えの議論を進めることが出来る。

(2-2)一般曲線座標系でのSOR法の並替えについて

例えば3次元一般曲線座標系に於けるポアソン方程式は図1に示されるように2次精度の中心差分で離散化すると19点近似式になる。この場合近似式に使われる点の数が増えれば増える程効率的な計算(即ちベクトル化)は難しくなる。何故なら、更新された新しいデータを順番に使えば使う程速く収束するが、それと引き換えに演算は再帰的になりベクトル化が困難になる。一方古いデータだけを使うとベクトル化は何ら問題なく行なえるが収束は極端に遅くなってしまふ。たとえばJacobi法はその代表的なものである。是非とも数学的な収束率を保持したままベクトル長の長い計算技法を考案する必要がある。

本論文で提案する方法は、再帰的な演算が無いように3次元配列を1次元的に表現した技法と言えよう。即ち各々の格子点を色分けし、その格子点の色と関係する周囲の18個の格子点の色とが全て違うように、即ち再帰的關係が無いようにしたものである。種々の色分けのうち最少の分け方が7色であるという原理に本技法は基づく。したがってベクトル長は反対に最も長い $O(N/7)$ が確保出来る。これが本技法の最大の特長である。(図2参照)

更にアドレス計算を単純化するために、各軸方向の格子点数に以下の様な約束を設けることにする。

- (1)第1軸の格子点数: $7m+P$ ($m=1,2,3$)
- (2)第2軸の格子点数: $7n-P$ ($n=1,2,3$)
- (3)第3軸の格子点数: 任意

但し $P=2,3,4,5$ とする。([6],[7] 参照) この様にすることによって初めて、ベクトル長が $O(N/7)$ と非常に長いベクトル計算機向きSOR法が実現する。格子の番号付けは自然なOrdering、即ち $(1,1,1), (2,1,1), (3,1,1), \dots, (NX,1,1), (1,2,1), (2,2,1), \dots, (Nx,NY,1), (1,1,2), \dots, (NX,NY,NZ)$

とする。この時、7色SOR法の計算順序は

(1)第1番目の色の格子点、1,8,15,22,29,

(2)第2番目の色の格子点、2,9,16,23,30,

(3)第3番目の色の格子点、3,10,17,24,31,

(第4、5、6番目の色の格子点も同様とする。)

(7)第7番目の色の格子点、7,14,21,28,35,

となる。<Appendix 1.にプログラム例を示す>

この計算技法は最初に述べた高速化の基本原則(1),(2),(4)を主眼に考えたアルゴリズムである。

(2-3) 7色SOR法の効率の数値実験

式(1)に示す2階の微分項を含む次の恒等式を考え、7色SOR法の効率の数値実験を行なった。

$$U_{xx} + U_{yy} + U_{zz} + U_{xy} + U_{yz} + U_{zx} = f \quad (1)$$

解析領域は $[0,1] \times [0,1] \times [0,1]$ であり、右辺項 f は以下の様に与えられる。

$$f(x,y,z) = e^{xyz} \{ (xy)^2 + (yz)^2 + (zx)^2 + (x+y+z)(1+xyz) \} \quad (2)$$

この時真の解は $U = e^{xyz}$ となる。各項を2次精度の中心差分で離散化する。

例えば $U_{xy} = 0.25 (U_{i+1,j+1} + U_{i-1,j-1} - U_{i+1,j-1} - U_{i-1,j+1})$ となる。

(k は省略する) 離散化方程式は全体として19点近似式になる。収束条件は相対残差 L_2 ノルムで 4×10^{-4} 、初期値は全て0、計算は単精度演算、格子点数は約1万点から約10万点まで10ケースについてテストを行なった。計算結果を表1に示す。

表1. 7色SOR法の計算速度 (単位:Mflops)

格子点数	VP-200	VP-400E	SX-2	S-820/80
(1) 30x19x18	278	363	844	852
(2) 30x26x26	280	361	854	809
(3) 37x26x31	280	362	868	869
(4) 37x33x33	282	370	870	859
(5) 44x33x34	281	368	872	919
(6) 44x40x34	281	367	874	865
(7) 51x40x34	282	371	875	824
(8) 51x47x33	281	370	874	872
(9) 58x47x33	282	370	876	828
(10) 58x54x32	282	372	880	886
平均速度	281	367	869	858

各ベクトル計算機の最大演算速度は、VP-200は 570Mflops, VP-400Eは1700 Mflops, SX-2は 1300Mflops, そしてS-820/80は 2000Mflopsである。

7色SOR法の平均速度が、それぞれの計算機の最大演算速度に対してどのくらいの比率に相当するかを計算してみると、それぞれVP-200は49%, VP-400Eは22%, SX-2は67% そして S-820/80は43%になった。

一般に数値解法の計算速度を比較する場合（精度を比較する場合と違い）、どうしても相対的な比較になることが多い。このような場合、効率の良し悪しの判断は比較する相手の選び方によって大きく左右される。そこで本論文ではより普遍的で、かつ実的な有効性の議論をするために、絶対的な基準に近いものとして計算速度のMflops値を採用することにした。3次元の19点近似式をSOR法で解く場合、四則演算の回数は一定である。即ちNを未格子点数とした時、誤差見積も含めて21N個の加減算と21N個の掛算、合計42N個の演算が必要になり、次の(3)式でMflops値は計算出来る。

$$\text{Mflops} = 42N \times \text{Iter} / (\text{CPU-time} \times 10^6) \quad (3)$$

ここでNは総格子点数, Iterは反復回数そしてCPU-timeは秒単位を表すものとする。参考までに他の解法の結果を次の表2に示す。使用した計算機がVP-400EとS-820/80の場合のみを示すが他の計算機でも同様の結果が得られた。（[8] 参照）

表2. 他のSOR法の計算速度（単位:Mflops）

格子点数	擬似SOR法	Checker-Board SOR法	
(1) 30x19x18	181 [82]	93 [50]	*各欄で左側の数値はS-820/80, [] 内の数値はVP-400Eの 結果を表す。
(2) 30x26x26	172 [89]	89 [54]	
(3) 37x26x31	204 [94]	109 [58]	
(4) 37x33x33	196 [98]	105 [63]	
(5) 44x33x34	232 [105]	127 [65]	
(6) 44x40x34	230 [111]	126 [69]	
(7) 51x40x34	265 [118]	146 [74]	
(8) 51x47x33	257 [127]	142 [82]	
(9) 58x47x33	291 [133]	161 [87]	
(10) 58x54x32	295 [136]	164 [93]	
平均速度	232 [109]	126 [70]	

表1、表2から7色SOR法が他のSOR法に比べて圧倒的に効率がよいことがわかる。ただしこの表の中で、擬似SOR法はChecker-Board SOR法よりMflops値が大きく一見効率的に見える。しかしこの方法は1方向のみJacobi法を適用

しベクトル化したものなので、この方法の収束率はSOR法本来の収束率と比べて大幅に低下している。〔1〕したがって収束までの反復回数は通常のSOR法より非常に多くなり効率が落ちる。

しかし表1からわかる様に各ベクトル計算機の最大演算速度との比ではまだ低率なので、この7色SOR法はまだ改良の余地が残されていると推測出来る。（計算速度の絶対的評価基準導入のポイントがここに有る。）

3. データを等間隔で参照・格納するときの効率

第2章で述べた7色SOR法の特長の1つは、アドレス計算の簡略化のために採用した7つおきに同じ色の格子点を配置することであった。このことはプログラムの中では例えば解ベクトル用の配列は1つで済む代わりに、データの参照や格納は当然7つおきになることを意味している。ところが汎用のベクトル計算機（正確には、数値実験で使用した上記の計算機では）に於いてはデータを連続的に参照・格納した時にその計算機が持つ最高性能が発揮されるように設計されている。したがって、上の数値実験で得られた7色SOR法の効率はその影響を少なからず受けている（低下している）と推察される。（どの程度効率が低下するか知りたいのだが、残念ながらそれを記述した資料は余り多くない。）そこで以下の様な簡単なプログラムを作成し、その低下率を見積った。

```
(例)      L=N * m                (m個おきの参照・格納)
          DO 100 i=1,L,m
            A(i)=b(i)+C(i)*D(i)
100      continue
```

図3. 等間隔データの参照・格納の検証用プログラム
（演算は単精度、データ数は10万個x500回）

結果を表4にまとめた。連続的にデータを参照・格納した時に比べてどの程度効率が低下するかが知りたいポイントで、Mflops値そのものには大した意味は無い。

表4. 等間隔データの参照・格納時の計算速度

	VP-200	VP-400E	SX-2	S-820/80
連続	350(100%)	445(100%)	469(100%)	517(100%)
2個おき	290	320	454	563
3個おき	190	185	398	450
4個おき	110	160	398	253
5個おき	205	210	398	520
6個おき	220	225	398	517
7個おき	110(31%)	185(42%)	398(85%)	526(102%)

表4からわかるように、連続的に計算した場合を各々100%とした時、計算機によっては2個以上等間隔にデータを参照・格納すると効率がかなり落ちる。特に7個おきの場合効率が約30%と大幅に落ちてしまう。特にVPの2機種はその傾向が目だつ。またS-820/80の場合はCPU計測のばらつきが大きく再考の余地が有る。それに対してSX-2の場合はほとんど効率が低下しないことがわかる。この結果を元に7色SOR法に次のような改良を施した。

4. 新しい7色SOR法の効率について

本論文では高速化が主目的であるため、第3章の得られた知見から判断して、汎用性を少し犠牲にして等間隔アドレッシングを見送った方がよいことがわかる。そこで第1章で述べた高速化のための基本原則(1),(2)は従来通り踏襲し、更に(3)の指針を取り入れた新しい7色SOR法を考えることにする。

そのためには、各色に対応してそれぞれ配列を持たせさえすれば解決する。すなわちプログラムは、係数行列用配列が7個、右辺定数用配列も7個そして解ベクトル用配列も7個必要になってくる。そうすれば各配列へのデータの参照・格納は連続的に出来ることになる。唯、この場合プログラムのステップ数は当然改良前の7倍に増大してしまい、配列毎のアドレス指定が煩雑になる。(2-3)と同じ問題を新しい7色SOR法で解いた。結果を表5に示す。

表5. 新しい7色SOR法の計算速度(単位:Mflops)

格子点数	VP-200	VP-400E	SX-2	S-820/80
(1) 30x19x18	441	568	996	1504
(2) 30x26x26	458	561	1002	1479
(3) 37x26x31	453	562	1017	1506
(4) 37x33x33	458	570	1030	1496
(5) 44x33x34	461	568	1024	1542
(6) 44x40x34	458	570	1028	1555
(7) 51x40x34	460	571	1029	1573
(8) 51x47x33	457	570	1025	1503
(9) 58x47x33	459	570	1035	1560
(10) 58x54x32	461	572	1034	1567
平均速度	457	568	1022	1529

新しい7色SOR法の平均速度は、表1の結果と比較して、VP-200では1.63倍、VP-400Eでは1.55倍、SX-2では1.17倍、そしてS820/80では1.78倍と速くなった。したがって計算機の最大演算速度に対する比率は、VP-200は80%、VP-400Eは33%、SX-2は79%、そしてS-820/80は76%となり、VP-400Eを除きかなりの高率を達成することが出来た。

なお、今回のテストに使用したプログラムは、CPU計測のルーチンと強制ベクトル命令以外は同一である。また一般曲線座標系で解析するプログラムに、新しい7色SOR法のプログラムを実際に組込むには若干の注意が必要になる。

Appendix 2.にまとめたので参照されたい。

5. まとめ

一般曲線座標系の下で現れる3次元の19点近似のポアソン方程式に対してベクトル長が $O(N/7)$ で見積られる7色SOR法を提案した。この解法に基づいたプログラムを複数の汎用ベクトル計算機上で効率を測定し、一部を除いて最大演算能力の75%から80%の効率を達成することが出来た。またベクトル計算機を使用する上で有効な技法も報告した。

6. 参考文献

- [1] 杉原、小柳、森、「ベクトル計算機におけるSOR法的方法について」、数値解析研究会、情報処理学会、1987.3
- [2] 村田、小国、唐木、「スーパーコンピュータ（科学技術計算への適用）」、丸善、1985
- [3] M.Ortega, 'Introduction to Parallel and Vector solution of Linear Systems', Plenum Press, 1988
- [4] 横川三津夫、「逐次過大緩和法（SOR法）のベクトル・パラレル処理」JAERI-M, 日本原子力研究所、1988
- [5] Loyce M.Adams, Harry F.Jordan, 'Is SOR color-blind?', SIAM.J.Sci. Stat. Comput., Vol.7, No.2, April, 1986
- [6] Fujino, S., Tamura, T. and Kuwahara, K., "Application of the RAINBOW SOR technique to Fluid Flow Analysis in the 3-D generalized Curvilinear Coordinate System", Proceedings of 6th International Conference on 'Numerical methods in Laminar and Turbulent Flow', Swansea, U.K., July, 1989
- [7] Fujino, S., Tamura, T. and Kuwahara, K., "Multicolored Poisson Solver for Fluid Flow Problems", Proceedings of the 8th GAMM Conference on Numerical Methods in Fluid Mechanics, University of Technology Delft, The Netherlands, September, 1989
- [8] 藤野、田村、「一般座標系2、3次元に於けるSOR法のベクトル化について」、数値解析研究会、情報処理学会、1989.10

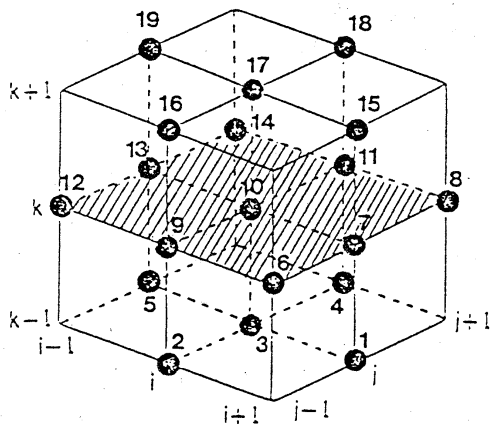


図1 3次元一般曲線座標系に於ける19点近似

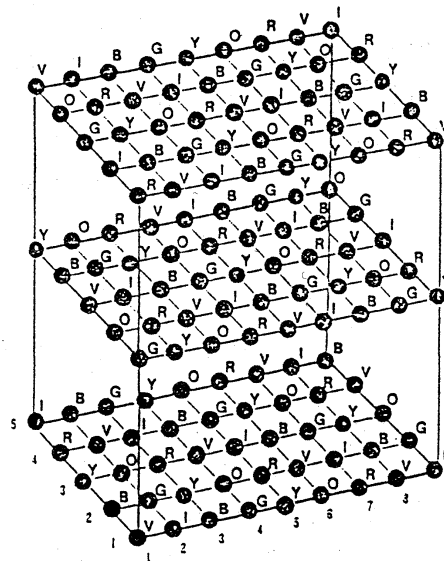


図2 7色の色分け

<Appendix 1.>

7色SOR法のプログラム例

外側のループが7色の色に対応し、内側のループは先頭番地と計算間隔を表している。

```

PARAMETER (MX=58,MY=54,MZ=32)
PARAMETER (N=MX*MY*MZ,M1=MX,M2=MX*MY)
DIMENSION XS(1-M2:N+M2),AS(N+M2,19),BS(N)

C ***** RAINBOW SOR *****
000273 400 CONTINUE
000274 ER=0.0
000275 S DO 410 K=1,7
      *VOCL LOOP,NOVREC
000276 V DO 410 L=K,N,7
000277 V GOSA=BS(L)-(AS(L,1)*XS(L-M2-M1)+AS(L,2)*XS(L-M2-1)+
      + AS(L,3)*XS(L-M2) +AS(L,4) *XS(L-M2+1)+
      + AS(L,5)*XS(L-M2+M1)+AS(L,6) *XS(L-M1-1)+
      + AS(L,7)*XS(L-M1) +AS(L,8) *XS(L-M1+1)+
      + AS(L,9)*XS(L-1) +AS(L,10)*XS(L)+AS(L,11)*XS(L+1)+
      + AS(L,12)*XS(L+M1-1)+AS(L,13)*XS(L+M1)+
      + AS(L,14)*XS(L+M1+1)+AS(L,15)*XS(L+M2-M1)+
      + AS(L,16)*XS(L+M2-1)+AS(L,17)*XS(L+M2)+
      + AS(L,18)*XS(L+M2+1)+AS(L,19)*XS(L+M2+M1))
000278 V ER=ER+GOSA*GOSA
000279 V XS(L)=XS(L)+OMEGA*GOSA
000280 V 410 CONTINUE

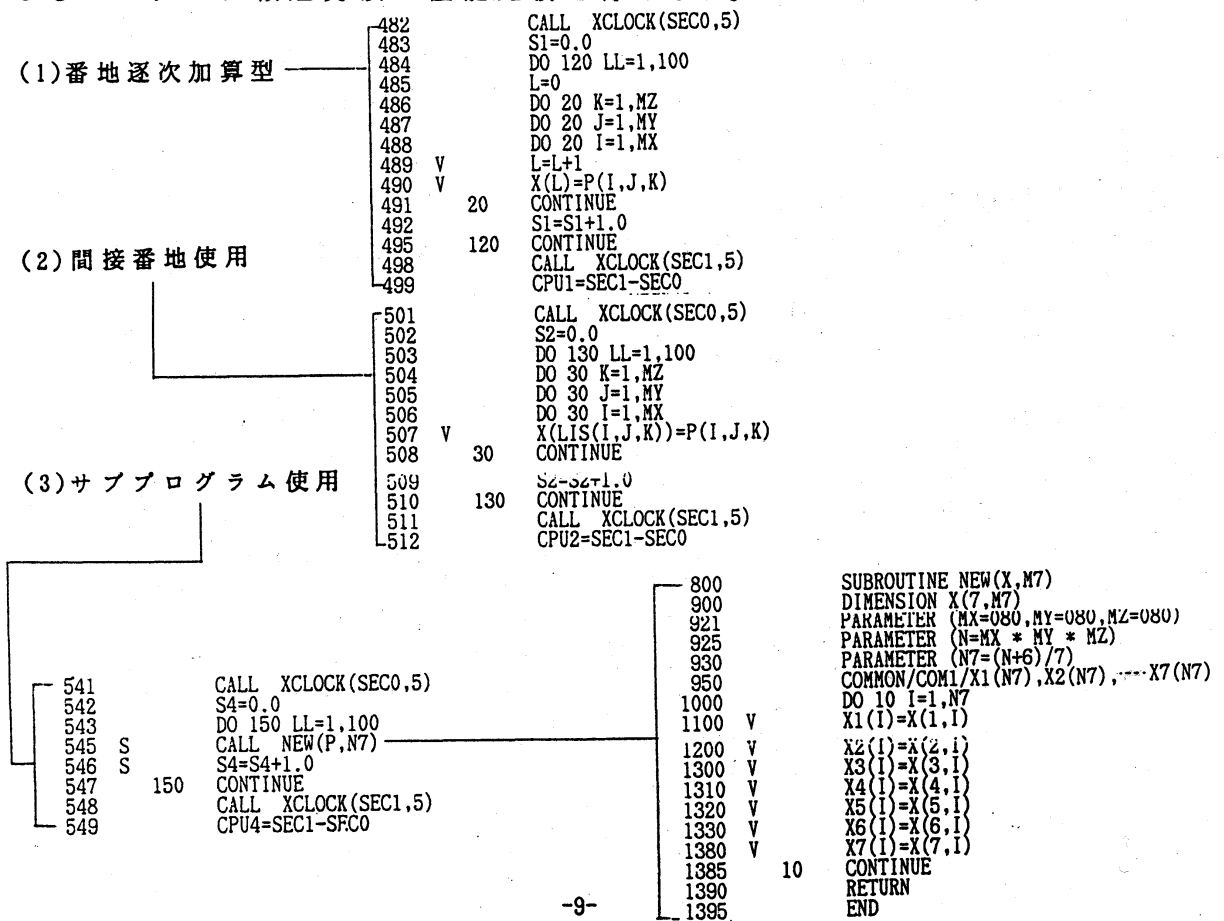
```


<Appendix 2.>

新しい7色SOR法の実際のImplementation時の注意

実際に新しい7色SOR法を一般曲線座標系の解析プログラムの中にImplementする場合以下の事に留意する必要がある。本文で述べたように、新しい7色SOR法では効率化を図るために、連立1次方程式用の配列として、係数行列用に7個、右辺項用に7個、そして解ベクトル用に7個それぞれ1次元配列を準備する必要がある。

ところが、一般的に3次元の物理空間に対応して配列を3次元配列:P(i,j,k)にとった方がわかり易くかつ便利ことが多い。一方本方法ではX1(i),X2(i),X3(i),X4(i),X5(i),X6(i),X7(i)という各1次元配列で計算を行なう。そのため配列Pから配列(X1,X2,...,X7)へのデータ転送(逆方向へのデータ転送も含む)も効率よくした方がよい。SOR法の反復計算の最初と最後の2回だけ転送すればよいケースではその負荷は大したことはない。しかし反復の途中で、何回も右辺項を修正したり境界条件の処理をしたりするケースも多く、元の物理空間に対応した3次元配列を直接取り扱い出来る方(アドレス計算などの手間のため)が都合が良い。その場合データ転送のオーバーヘッドはかなり大きくなる。そこで考えられる3つのデータ転送方法の性能比較を行なった。



数値実験結果を表Aに示す。方法(3)が最も効率がよいことがわかる。

ただしこの場合ベクトル長が(A)50、(B)80と非常に短いため、下の結果は必ずしもその機種のパフォーマンス全体を表しているわけではない。ここでの関心は3次元配列から1次元配列へのデータ転送時間を短くする方法を見つけることにある。

表A. 3次元配列から1次元配列へのデータの転送速度

(A) データ数が50×50×40個の場合 (単位: mili sec)

	VP-200	VP-400E	SX-2	S-820/80
(1)番地逐次加算型	2.72	2.16	0.94	0.87
(2)間接番地使用	19.0	18.9	2.16	3.09
(3)サブプログラム使用	0.93	0.56	0.29	0.27

(B) データ数が80×80×80個の場合 (単位: mili sec)

	VP-200	VP-400E	SX-2	S-820/80
(1)番地逐次加算型	12.0	9.31	4.50	3.86
(2)間接番地使用	94.7	94.0	9.83	14.7
(3)サブプログラム使用	4.68	2.77	1.49	1.38