

はじめに

本稿は、1987年4月、本学知識情報研究所フォーラム：ソフトウェア開発の問題点にて筆者が講演した同タイトルの予稿を現時点でふりかえり、この三年間におけるスーパーコンピュータ環境の変化をふまえて再論したものである。この間、我国のスーパーコンピュータは、東大大型計算機センターがHITAC810/20から820/80に変わったように、第一世代から第二世代への移行が行なわれた。それに伴って、ソフトウェア環境にも変わったことと不変なことが見られた。そのことを3年前の予稿を今回見て実感するところがあった。よって、3年前のデータもここで抄録し、読者の便に供したい。（こういうデータはやがて散逸するのを常とする。）

1 分野別の外観

スーパーコンピュータを頂点とする数値シミュレーションの有用性に関し、今更喋々する必要は無いが、数値シミュレーションの本性上、その普及の程度には分野毎にかなりの相違が見られる。それに応じて、スーパーコンピュータに対する我々の態度も異なってくる。以下、便宜上

- 構造解析（熱解析を含む）
- 電磁界計算
- 流体解析

の三分野に大別して若干のコメントを加えたい。——ただしここで述べることはあくまで一般向けの概論である。やや専門にわたる話は、小国力氏の講演にあらわれよう。

● 構造解析分野

もっとも普及し、スーパーコンピュータ以前の時代から多くのソフトウェアの蓄積がある。そして現在でもスーパーコンピュータを有効に使うべく学会、産業界双方で最も多くの人々が従事してお

られる分野である。さて、その対応の仕方を大別すると1)、2)の二つに分けられよう。

1) 既存ソフトウェアのスーパーコンピュータ向けチューニング

(例えば、NASTRAN, ADINA, SAP/NONSAP, MARC)

問題点：3年前、以下のように書いた：I/Oをそのままでは演算部分をプログラムレベルでチューニングしても効果はわずかである。コンパティビティーの問題、保守の問題から算法にまで手を入れ難いから、効果は期待されるほど目覚しくないようである。そのままかどうかの一例を表1に示す。

表1 そのままでどうかの一例：静解析、単位はsec.

#	自由度	平均帯巾	最大帯巾	行列作成	LU solv.	その他	全体
1	2121	126	211	9.4	5.5	22.2	37.
				11.5	11.7	22.8	46.
2	7752	211	3334	28.8	47.8	28.8	122.
				35.0	112.4	35.0	194.

注：上段がVectorモード、下段がScalarモード

表1から分かるように、仮りにLU Solverの部分をチューニングにより1/10に縮めたとしても全体から見た効果は僅かである。#2の例で全体時間が122秒が約80秒に縮まるだけである。しかし日常的に使うから効果は多きいとも言える。以上が3年前の記述であるが近頃、ベクトル処理可能かどうかについてのコンパイラの解説能力がかなり向上し、もう少しよくなろうが、上記の視点から見ての大勢は似たようなものである。なお既存のソフトウェアのチューニングは、この3年で適正水準に達したとみられる。例えばNASTRAN、ADINA、SAP/NONSAP、MARC。これらは特に著名だが、ほかにもここ3年間に、スーパーコンピュータにチューニング済みの流通ソフトウェアが多数出まわった。(例えば日立のREFER(流通ソフトウェア紹介制度)登録のもので、8件だったものが24件に。)また、解析支援(I/O関連)のプリ・ポストプロセッサも多様になった。よって、これらをブラックボックスとして使う純ユーザから見た使用環境は大幅に向上した。スーパーコンピュータの普及が、ソフトハウスの企業意欲を高めたことが背景にある。

2) スーパーコンピュータを意識して新たに作る。

産業界でかなりやられているが社外秘が普通で子細は不明である。大学で手がけているものは比較的内容まで分かる。(例：東大工・三好研究室⁽¹⁾)

新規作成に当たっての留意点は、当然のことながら、

○I/Oの効率化

○ソルバーの選択とプログラミング技法

である。ソルバーとI/Oの関係も深い。例えばICCG系と帯ガウス、スカイ

ライン系とではディスクとの関わり方がちがう。半導体拡張メモリを買うかどうにかによっても様子がちがう。帯幅が均一に近いときはスカイラインよりも、普通の帯ガウスの方がかなり速いから、両方用意して使用時に選択させることもおこなわれる。

SIMUS（日立、機械研究所開発）の例⁽²⁾

Solverは、外部記憶（ディスク）を用いた、3行3列同時分解のスカイラインである。コンピュータはHITAC-S810/20、約1万接点（3万自由度）の問題をスカラモードにて約3600秒、ベクトルモードにて155秒、加速率23倍を得たと報告している。

三好等の報告⁽¹⁾によると、主メモリ27.5 MB、拡張半導体記憶128MB使用という環境にてICCG系の場合約20000節点が可能、スカイライン系の場合約6000節点止まりという。また13616節点にて主メモリ15MB、拡張半導体メモリ66MB使用のICCG系にて8分14秒という。

以上が3年前の報告の通りであるが、現時点に立って若干のコメントを加える。SIMUSについてはその後2編の論文^(2-a, 2-b)が出た。応用例として、半導体の薄膜形成時の熱応力と真性応力の重畳付加の解析が報告されている。^(2-c)三好等の報告もその後何編かの論文と成書⁽⁴⁾によって追跡することができる。SIMUSは、伝統的な構造解析の応用分野を越えて大変形・熱応力解析プログラムが応用された好例である。

今述べているカテゴリーに入るプログラム開発は、

- 1) 大資本の投入可能な分野（例、半導体産業、自動車産業、原子力）
- 2) 現象のモデル化から新規にとり組む必要のある分野

の二条件を満たす分野において成長が促進される。ここ3年の間に、上の例のほかにも数例を見聞した。^(5, 6)米国一辺倒の応用ソフトウェアから脱却が、この方面から促進されることを期待したい。

なお、近頃この分野で、境界適合法と部分構造反復法、あるいは両者の組合わせを許すプログラム開発が盛んに見受けられる。実際、境界適合法から生成される行列は、非対称ながら帯行列で、帯ガウスのベクトル計算効率が高いからである。また、部分構造反復法は、収束の問題を新たに持ちこむけれども、構造の大規模化によるメモリーネックを余裕のできたCPU能力に肩代りさせることを許す手

法と見られるからである。収束の問題は、部分構造を大胆に重ねることである程度解決させることができる。ICCG法、ブロックスカイライン法、境界適合法・部分構造反復法の三者は三つ巴の様相を呈しよう。

● 電磁界計算分野

ここでも、初めに3年前の文章をそのままかけ、後でコメントを加える。

構造解析に次いで産業界でも設計計算に使われている分野である。ただし社外秘のものが普通で、内容の詳細を知ることができるのは大学の開発によるものに限られる。（岡山大学、中田研など。）

有限要素法、境界要素法、（差分法を一般化した）境界適合法などが使われている。定常の静磁界計算は、弱い非線形問題ならば実用的な三次元問題まで手がとどくようになったと言われる。強い非線形問題となると物理的考察をベースとしたモデル化の問題を含めて問題が多いと言われる。例えば文献（3）。

‘動く磁場’の過渡解析となると格段に難しくなる。対称正定値問題でなくなるからである。Solverとして、非対称系に対する帯あるいはスカイラインガウスあるいはPBCG（前処理つき双共役勾配法）が必要となることは言うまでもない。

デバイスCADは電気の問題ではあるが、本質的にはむしろ連立の移流拡散問題だから、そして物性値がけた違いにちがう場を扱うため問題が多いが、スーパーコンピュータの大きな顧客である。

以上が3年前の文章だが、現時点での見直しとコメントを加える。

まず、ここ3年間に産業界でこの方面の進歩が著しかったと言える。詳細は企業秘密で不明だが、学会発表ばかりでなく東芝レビュー、日立評論といった一般PRを兼ねたものにも載るようになったから、普及が著しかったものと見られる。

3次元の（うず電流を含む）磁界計算の現状については、この方面の権威：中田高義教授（岡山大）による見通しのよい総論：文献（7）を見るとよい。従来法；すなわち磁気ベクトルポテンシャル A と電気スカラーポテンシャル ϕ の連立系による‘ $A-\phi$ 法’に代わって、磁気スカラーポテンシャル Ω も使用する‘ $A-\phi-\Omega$ 法’や電流ベクトルポテンシャル I と Ω を用いる‘ $I-\Omega$ 法’が推奨されている。何れも最近（1988年）発表の手法である。むすびに次の言葉が見られる：“うず電流を含む非線形問題は、形状が相当単純でも、スーパーコンピュータで10時間以上のCPU時間を必要とするので、まだ実用段階に到達したとは言い難い。ヒステリシスを含む問題も同様である。”

ここ3年の間に、いくつかのプログラムが流通ソフトウェアとして市販されるようになった。しかし、中田教授も言っている通り、三次元磁気解析は定式

化に問題があることもあり、また解析方法は乱立状態にあるから、ブラックボックスとしてこれらの流通ソフトウェアを使うことは戒められよう。

デバイスCADについて付言しておく。現在通常使われているモデルは、‘ドリフト・拡散モデル’と呼ばれるもので、電子密度 n 、正孔密度 p 、電位 ψ に関する連立偏微分方程式系：

$$\begin{cases} q \frac{\partial n}{\partial t} = \text{div}(J_n) + G - U, & \text{ただし } J_n = q(D_n \nabla n - n \mu_n \nabla \psi) \\ q \frac{\partial p}{\partial t} = -\text{div}(J_p) + G - U, & \text{" } J_p = -q(D_p \nabla p + p \mu_p \nabla \psi) \\ \text{div}(\epsilon \nabla \psi) = q(p - n + N_D - N_A) \end{cases}$$

である。もとは量子統計学的なボルツマン方程式からモデル化されたもので、 D_n 、 D_p （拡散係数）、 μ_n 、 μ_p （移動度）、 G （発生）、 U （再結合）の諸項に適当なモデル化式が使用される。その所が難しい。モデル化に問題はあるけれども、三次元の過渡解析まで、一応の実用の域に達した。それはスーパーコンピュータあつてのことであり、また数値計算手法の長足の進歩あつてのことと言われる。（ICCG法、PBCG法、PCGS法がおおいに使われる。）

このごろ、プロセスCAD→デバイスCAD→回路CADを統合したシミュレーションシステムの開発が盛んである。例えば文献（8）。また一方では、デバイス寸法の微細化、デバイスの多様化に伴い、上記モデルでは扱いきれぬ問題がクローズアップされ、もつと精密なモデル、たとえば‘緩和時間モデル’がとり上げられるようになった。それはやはりボルツマン方程式をモデル化したものではあるが、先程のドリフト・拡散モデルと比べるとはなはだ複雑である。（たとえば先程のドリフト・拡散モデルで、電子1個の連続方程式が、 j 個の伝導帯の各々についての3つの保存式（粒子数保存、運動量保存、エネルギー保存）に細分化される。 $j=3$ なら9個の保存式）。モデルを数値シミュレーション用に関じさせることの大変さもさることながら、数値計算的にも、東京湾から太平洋に漕ぎ出したような難儀が待ちかまえていよう。ともかく、デバイスCADは、これからますますスーパーコンピュータの大顧客となろう。

なお、従来の磁気計算（すなわちマクスウェルの電磁気学のワク内の計算）とは違ったものに、磁気記録、光磁気記録に関連した数値シミュレーションが話題になりつつあることを付言しておこう。そこでは、磁区の形成、回転といった、モデル化の段階からの問題がある。

●流体解析の分野

スーパーコンピュータにとっても最大の難敵である。このことは、次期、次々期のスーパーコンピュータにとってもそうであろう。現実のながれは乱流だからである。今注目されているLES (Large Eddy Simulation) も、 $100 \times 100 \times 100$ のあみ目でようやく実用化の門口にたつ、というほどのことであろうか。

—3年前の報告ではまことに素気なくこれだけ書いた。今、同じ行数で要約せよと言われたら、やはりこの程度のことしか言えないと思うが、ここですこし付言しよう。流れは我々人類にとって最も身近でありながら、数値シミュレーションの対象として、最大の難敵であるとは皮肉なことである。しかも、地球大気が乱流だからこそ我々生物は生きていれるのである。(太陽熱が乱流拡散されるから焼死しない!)

流れ場の解析といってもそれぞれ特定の目的に応じてモデル化がおこなわれ、数値解法もそれに依存して選ばれる。たとえば、

- a) 飛行機の廻りの流れの計算：応力や揚力、抗力の計算が目的
- b) 大気の大循環：天気予報が目的
- c) 室内気流：温度や、ゴミの分布を知りたい。
- d) タービン内流れ：エネルギー変換を効率よく行ないたい。

などである。

数値シミュレーションの立場から(算法に入る前に)二大別すると便利である

●圧縮性流れ (上例ではa)とd))

●非圧縮性流れ (上例ではb)とc))

何れにせよスーパーコンピュータによってゆつくりではあるが、実用的価値が高まって来ていることは確かである。例えば天気予報の当たる確率が気象庁のスーパーコンピュータの導入を境として、庶民に分かる程度に上がったことは日常の話題になっている。また、最近の日立評論によると、メーカーの宣伝もあるから元気がよいことは割り引いて聞くとしても⁽⁶⁾実際の製品開発に応用され実績を上げているという。

スーパーコンピュータS810/20により、従来の汎用機M200Hと比べ、圧縮性流れ解析(差分法を一般化した境界適合法)で約20倍、非圧縮性流れ解析(有限要素法による $k-\epsilon$ モデル)では約30倍の性能向上が得られたという。30倍ということは、30時間→1時間だから、非現実→現実という革命が起こる。LESについての適用例も報告されている。

この章のしめつくくりとしてひとこと：

1) スーパーコンピュータの有用性がこの3年間に広く認められ、それが日常的に、けた違いに多くの人によって使われるようになった。

2) また使い方も変わった。例えば一人の人が自分の室から端末を通して汎用機とスーパー機を（ときには個人用のワークステーションも）交互に使う機会が増した。そのためスーパー機にだけ有効なようにチューニングしたプログラムを別立てで用意し保守するのがわずらはしくなった。一方コンパイラの改善のため、少しの配慮によって汎用、スーパー機共に効率のよいプログラム作成が可能になった。

3) 流通ソフトウェアが急伸した一方、既成のものでは済まない応用分野が急速に成長した。そして現象の新しいモデル化の必要性が高まり、メモリネックがますます痛感させられるようになった。

この章の文献：

(1) 三好俊郎、高野直樹ほか：スーパーコンピュータによる大規模構造解析；連続体力学における大規模計算研究集会論文集 PP-12(1987.3)

(2) 斉藤直人、坂田信二ほか：スーパーコンピュータ用高速構造解析プログラムの開発；同上、PP13-14(1987.3)

(2-a) 斉藤、外：薄膜多層構造体応力解析プログラムSIMUS2D/Fの開発；機械学会論文集A-55、515、PP1652-1656(1989.7)

(2-b) 斉藤、外：高速構造解析プログラムSIMUSの開発；同上A-53、495、PP2187-2192(1987.11)

(3) 宮田浩一、宮健三：磁性材料における三次元非線形磁界解析；(1)に同じ論文集PP34-39(1987.3)

(4) 村田・小国・三好・小柳編著：工学における数値シミュレーションPP81-92(1988)丸善。

(5) 中島幸雄：スーパーコンピュータによるタイヤ構造解析；日立評論 Vol72,(1990.3)PP39-44

(6) 守田邦宜ほか：機械工学におけるスーパーコンピュータの高度利用；同上(1990.3)PP21-26

(7) 中田高義、藤原耕二：三次元磁界計算の現状と展望、応用磁気学会研究会資料60.3(1989),PP19-28

(8) 松屋他：半導体設計におけるスーパーコンピュータの応用、日立評論VOL72,(1990.),PP9-16

2 汎用、スーパー機共用のガウス消去法系諸算法とプログラム技法

前章の最後でもふれたように、汎用機、スーパー機別立てのプログラムを用意し、保守するということは甚だ不便である。一方スーパー機用のコンパイラのソース解説能力が向上し、少しの配慮によって両者共用のプログラム作成がかなり容易になった。

算法（アルゴリズム）の選択にあたっては次の方針をとる：

a： 汎用機、スーパー機共に効率良い算法

b： 特に主メモリを越える大規模計算に使える算法

そして、算譜（プログラム）上にプログラマが陽に書く‘陽アンロール’と、コンパイラが行なう‘陰アンロール’に注目しながら話を進める。

さて、ガウスの消去法と総称される算法には、いくつかの変種があるが、結局正統的なガウス、即ちk段の消去過程を、

$$\text{do } \left[\begin{array}{l} i = k+1, n \\ j = k+1, n \end{array} \right] \quad a_{ij}^{(new)} = a_{ij}^{(old)} - \frac{a_{ik} \cdot a_{kj}}{a_{kk}}$$

によって行なうものが最適である。実際このガウスを變形して、再内側のループをベクトルの内積型の演算

$$S = S + l_{ik} u_{kj} \quad (k \text{ を動かす})$$

で行なわせるものをクラウト法と総称するが、それは上記 a、b 何れの要求にも合格しない：a に関してはスーパー機にとってアンロールし難いため遅く、b に関しては、密行列においては後述のタテブロックガウスのような有効な対処法がなく、帯行列においては部分軸選択によるリスト i p (k) 使用のステートメントが最内側にあらわれるため（遅いばかりでなく）ブロック化ができない。詳細は（1）に譲るが、データだけはここに示しておく。表 1 参照。これは部分軸選択ありのプログラムによるものであるが、部分軸選択なしの場合（クラウトもブロック化可能となるが）やはり表 1 と傾向的には同様で、汎用機ではほとんど同じ、スーパー機では遅くなる。対称正定値用の帯ガウスと帯コレスキーについても同様である。（後述）

表 1：列ガウスとクラウトの比較 (sec)

	HITAC-M260H (国産)			S-810/20 (東大, 87.8)		
元数	GLU1 (ガウス)	CLU0 (クラウト)	CLU1 (クラウト)	GLU1	CLU0	CLU1
400	18.8	18.0	18.2	0.23	0.69	0.51
600	63.1	62.1	59.4	0.68	1.94	1.31
800	149	238	140	1.54	4.64	2.72

2. 1 密行列ガウス

正統的ガウスを列ガウスと行ガウスの2型に分類する。

列ガウス

do $k=1, n$

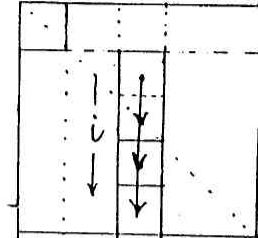
do $i=k+1, n$

$$a(i, k) = -a(i, k) / a(k, k)$$

do $j=k+1, n$

do $i=k+1, n$

$$a(i, j) = a(i, j) + a(i, k) * a(k, j)$$



行ガウス

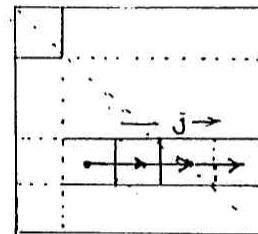
do $k=1, n$

do $i=k+1, n$

$$a(i, k) = -a(i, k) / a(k, k)$$

do $j=k+1, n$

$$a(i, j) = a(i, j) + a(i, k) * a(k, j)$$



密行列に対しては、汎用機の場合必ず列ガウスによらねばならぬことは周知であるが、スーパー機においても同じである。表2参照。理由は列型のとき最内側の添字(i)が動き、連続番地参照となるからである。

次頁に列ガウスGLU1のプログラムを示す。

表2 列ガウスと行ガウスの比較(sec)

元数	汎用機			スーパー機		
	GLU1 列ガウス	GLUR0W 行ガウス	行 列 比	GLU1	GLUR0W	行 列 比
200	2.4s	2.43	1.01	.089	.125	1.40
400	18.8	25.0	1.33	.528	.818	1.55
600	63.1	109.4	1.73	1.57	2.51	1.60
800	149.	445.	3.00	3.49	8.40	2.40

陽アンロールと陰アンロール:

* 86.10, 旧コンパイラ(アンロールせず)による。

スーパー機においては、並列化の程度によって仮に並、上、特と称するクラス展開が同じシリーズ内になされている。(例えばS820/40, /60, /80)そこで、上、特機の場合、その並列の程度に応じて仕事をさせたい。そのための手法にアンロール手法がある。そのうち陽アンロールとはプログラム上で陽にアンロールプログラムを書くことを言い、陰アンロールとはコンパイラがアンロールできるか否かを判別しておこなうもののこととする。さて、

```

*
*               列型ガウス
*
SUBROUTINE GLU1( N, A, IP, EPS, IR, WK )
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(N,N), IP(N), WK(N)

*
IR = 0
DO 100 K = 1, N
*
*               部分軸選択
    AMAX = ABS(A(K,K))
    IPK = K
    DO 110 I = K+1, N
        AIK = ABS(A(I,K))
        IF( AIK.GT.AMAX ) THEN
            IPK = I
            AMAX = AIK
        END IF
110    CONTINUE
    IP(K) = IPK
*
    IF( AMAX.GT.EPS ) THEN
        IF( IPK.NE.K ) THEN
            W = A(IPK,K)
            A(IPK,K) = A(K,K)
            A(K,K) = W
        END IF
*
*               α の計算
        DO 120 I = K+1, N
            A(I,K) = -A(I,K)/A(K,K)
120        WK(I) = A(I,K)
*
        DO 130 J = K+1, N
*
*               方程式入れ換え
            IF( IPK.NE.K ) THEN
                W = A(IPK,J)
                A(IPK,J) = A(K,J)
                A(K,J) = W
            END IF
*
*               Gaussの消去法
            T = A(K,J)
            DO 140 I = K+1, N
                A(I,J) = A(I,J)+WK(I)*T
140            CONTINUE
130        CONTINUE
*
    ELSE
        IR = IR+1
        IP(K) = K
        DO 150 I = K+1, N
            A(I,K) = 0.000
150        CONTINUE
    END IF
100 CONTINUE
RETURN
END

```

```

*
*               右辺の計算
*
SUBROUTINE GSLV1( N, A, B, IP, IR )
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(N,N), B(N), IP(N)
*
*               前進消去
IF( IR.EQ.0 ) THEN
*
    DO 100 K = 1, N
*
*               方程式入れ換え
        IF( IP(K).NE.K ) THEN
            W = B(IP(K))
            B(IP(K)) = B(K)
            B(K) = W
        END IF
*
*               Gaussの消去法
        T = B(K)
        DO 110 I = K+1, N
            B(I) = B(I)+A(I,K)*T
110        CONTINUE
100    CONTINUE
*
*               後退代入
        B(N) = B(N)/A(N,N)
        DO 200 K = N-1, 1, -1
            T = B(K+1)
            DO 210 I = 1, K
                B(I) = B(I)-A(I,K+1)*T
210            B(K) = B(K)/A(K,K)
200        CONTINUE
    END IF
    RETURN
END

```

GLU1 (列型カウス):

do k = 1, n

部分軸選択 (略)

do i = k+1, n

$a(i, k) = -a(i, k) / a(k, k)$

$ak(i) = a(i, k)$

do j = k+1, n

$\beta = a(k, j)$

do i = k+1, n

$a(i, j) = a(i, j) + ak(i) * \beta$

二列同時カウス (j のループをアンロール)

do j = k+1, n, 2

$t = a(k, j)$

$u = a(k, j+1)$

do i = k+1, n

$a(i, j) = a(i, j) + ak(i) * t$

$a(i, j+1) = a(i, j+1) + ak(i) * u$

に対し、j のループを陽にアンロールしたものを2列同時ガウスという。上の右側。ところでこの算法では、最内側ループでロード3本、ストア2本のパイプがある。じつは汎用上位機種もパイプライン方式の計算機になっているから、これら機種に共通してストア（とブランチ）は負担となる。そこで上記GLU1の、kのループをアンロールすることが考えられた。すなわちk段とk+1段の消去過程を1度に実行するように仕向けるのである。部分軸選択が無ければ簡単だが、部分軸選択のためかなり厄介である。プログラムの大筋を次頁に示す。if文を使わないで主要ループを構成するため、DIMENSIONに工夫がされている。

この場合最内側のループはロード3本、ストア1本で済む。

コンパイラによる陰アンロールは、最内側のひとつ外側のループをアンロールするに止まるから、今述べた2段同時ガウスは、陽アンロールのプログラムを書いて始めて実行される。それに対し、2列同時ガウスは、実際には陽に書かなくても、陰アンロールによって自然に達成される。実際には陰アンロールは4列同時とされるのが普通である。陽にアンロールされた2段同時ガウス：GLU2、それをさらに陽に2列同時化したGLU4の性能を表3に示す。

表3左を見ると、汎用機の場合にも、GLU1に対して、GLU2、GLU4と進むにつれて速くなっていることが読み取れる。

表3, 列カウス GLU1, GLU2, GLU4の性能比較: (sec)

M260H (関情大)				S810/20 (東大) 86.10 陰なしのコンパイラ				S810/20 (東大) 87.10 陰ありのコンパイラ			S820/80 (東大) 88.9 陰ありのコンパイラ		
元数	GLU1	GLU2	GLU4	元数	GLU1	GLU2	GLU4	GLU1	GLU2	GLU4	GLU1	GLU2	GLU4
800	151	127	107	1000	6.55	3.59	2.08	2.92	1.55*	1.55	.581	.426	.426

* $wk()$, $wkl()$ を使わぬと1.72 となった。

GLU2 (2段同時ガウス) の大筋

do k = 1, n, 2

k段の部分軸選択, p行をえらぶ.

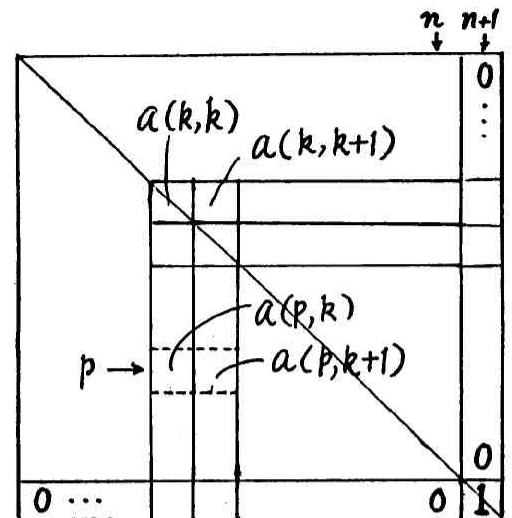
$$a(k, k) \rightleftharpoons a(p, k)$$

$$a(k, k+1) \rightleftharpoons a(p, k+1)$$

do i = k+1, n

$$a(i, k) = -a(i, k)/a(k, k)$$

$$wk(i) = a(i, k)$$



do i = k+1, n

$$a(i, k+1) = a(i, k+1) + wk(i) * a(k, k+1)$$

k+1段の部分軸選択, p1行をえらぶ.

$$a(k+1, k+1) \rightleftharpoons a(p1, k+1)$$

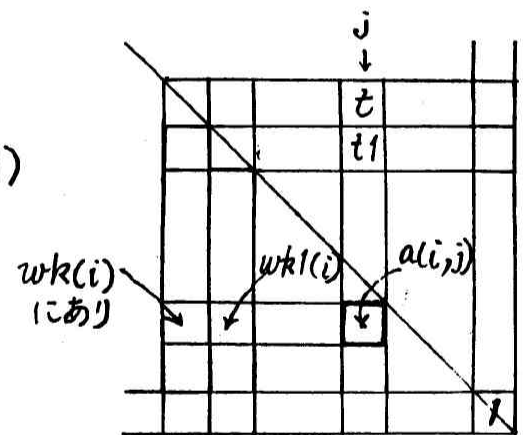
$$a(k+1, k) \rightleftharpoons a(p1, k) \quad \leftarrow \text{これを忘れないで.}$$

$$wk(k+1) \rightleftharpoons wk(p1)$$

do i = k+2, n

$$a(i, k+1) = -a(i, k+1)/a(k+1, k+1)$$

$$wk1(i) = a(i, k+1)$$



do j = k+2, n

$$a(k, j) \rightleftharpoons a(p, j)$$

$$a(k+1, j) \rightleftharpoons a(p1, j)$$

$$a(k+1, j) = a(k+1, j) + wk(k+1) * a(k, j)$$

$$t = a(k, j)$$

$$t1 = a(k+1, j)$$

do i = k+2, n

$$a(i, j) = a(i, j) + wk(i) * t + wk1(i) * t1$$

次頁に GLU2 のプログラムを示す.

```

*
*               列型ガウス・2段同時化
*
SUBROUTINE GLU2( N, A, IP, EPS, IR, WK1, WK2 )
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(N+1,N+1), IP(N+1), WK1(N), WK2(N)
*               初期値の設定
IR = 0
DO 10 J = 1, N
10  A(N+1,J) = 0.0D0
DO 20 I = 1, N
20  A(I,N+1) = 0.0D0
A(N+1,N+1) = 1.0
*
DO 100 K = 1, N, 2
*               k 段の部分軸選択
K1 = K+1
IPK = K
AMAX = ABS(A(K,K))
DO 110 I = K+1, N
AIK = ABS(A(I,K))
IF( AIK.GT.AMAX ) THEN
IPK = I
AMAX = AIK
END IF
110 CONTINUE
IP(K) = IPK
*               k 段の方程式入れ換え・k 列と k + 1 列
IF( AMAX.GT.EPS ) THEN
IF( IPK.NE.K ) THEN
W = A(IPK,K)
A(IPK,K) = A(K,K)
A(K,K) = W
W = A(IPK,K1)
A(IPK,K1) = A(K,K1)
A(K,K1) = W
END IF
*               α の計算と k + 1 列だけの消去変換
T0 = A(K,K)
T1 = A(K,K1)
DO 120 I = K+1, N
120  A(I,K) = -A(I,K)/T0
A(I,K1) = A(I,K1)+A(I,K)*T1
ELSE
IR = IR+1
IP(K) = K
DO 140 I = K+1, N
140  A(I,K) = 0.0D0
END IF
*               k + 1 段の部分軸選択
AMAX = ABS(A(K1,K1))
IPK1 = K1
DO 210 I = K1+1, N
AIK = ABS(A(I,K1))
IF( AIK.GT.AMAX ) THEN
IPK1 = I
AMAX = AIK
END IF
210 CONTINUE
IP(K1) = IPK1
*               k + 1 段の方程式入れ換え・k 列と k + 1 列
IF( AMAX.GT.EPS ) THEN
IF( IPK1.NE.K1 ) THEN
W = A(IPK1,K)
A(IPK1,K) = A(K1,K)
A(K1,K) = W
W = A(IPK1,K1)
A(IPK1,K1) = A(K1,K1)
A(K1,K1) = W
END IF
*               α の計算
DO 220 I = K1+1, N
220  A(I,K1) = -A(I,K1)/A(K1,K1)
ELSE
IR = IR+1
IP(K1) = K1
DO 230 I = K1+1, N
230  A(I,K1) = 0.0D0
END IF
*
DO 290 I = K1+1, N
WK1(I) = A(I,K)
WK2(I) = A(I,K1)
290
*
DO 300 J = K1+1, N
*               k 段の方程式入れ換え
IF( IPK.NE.K ) THEN
W = A(IPK,J)
A(IPK,J) = A(K,J)
A(K,J) = W
END IF
*               k + 1 段の方程式入れ換え
IF( IPK1.NE.K1 ) THEN
W = A(IPK1,J)
A(IPK1,J) = A(K1,J)
A(K1,J) = W
END IF
*               k + 1 行だけの消去変換
A(K1,J) = A(K1,J)+A(K1,K)*A(K,J)
*               Gaussの消去法・2段同時
T = A(K,J)
T1 = A(K1,J)
DO 310 I = K1+1, N
310  A(I,J) = A(I,J)+WK1(I)*T+WK2(I)*T1
300 CONTINUE
100 CONTINUE
RETURN
END

```

縦ブロックガウス:

これはもと、汎用機のページスワップを減らすために筆者が考えたものだが、スーパー機にとってはI/Oつきガウスの算法の基礎となる。本質的には、列ガウスと同じことを行なっているが、コンパイラが、(ベクトル処理はするが)陰アンロールしないところが出易いので注意がいる。たとえばGLUB1、GLUB2において、 $wk()$ 、 $wk1()$ を使わぬときS810/20では表4左の()内のように、ひどく遅かった。陰アンロールするかしないかは、一般のユーザーにはやってみないとわからない。

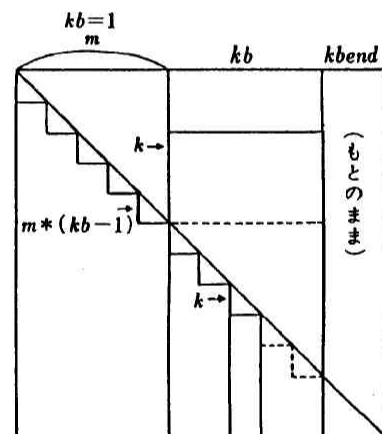
表4 縦ブロックガウスGLUB(μc)

	S810/20 87.10, 陰ありコンパイラ		S820/80 88.9, 陰ありコンパイラ	
元数	GLUB1	GLUB2	GLUB1	GLUB2
1000	2.98	2.11	.573	.442
〃	(5.91)	(3.12)		

縦ブロックガウス GLUB1 の大筋

```

ir=0
do kb=1, kbend
  do k=1, m*(kb-1) | kb=1 のときは 0
    if ipk(k) ≠ 0 then
      第 kb ブロックに対し
      1 段 ~ m*(kb-1) 段
      の消去過程を行う
    do k=m*(kb-1)+1, min(m*kb, n)
      | a(i, k) | の max. look,
      ip と amax を決定
      ipk(k)=ip
      if amax > eps then
        第 kb ブロックに対し
        m*(kb-1)+1 ~ min(m*kb, n) 段
        の消去過程を行う
      else
        ir=ir+1; ipk(k)=0
  
```



2. 2 帯ガウス

‘横方向番地ずけ’ といわれている方法、すなわち

$$a(j-i, i) = a_{ij}$$

の形で行列 (a_{ij}) を配列 $a(-m: 2m+1, n)$ に収容し、行ガウスの算法を適用する。それで、最内側のループ内で左側の添字 j が動くようになる。

帯ガウス BGLU1 の大筋

```

ir=0
do k=1, n
  amax=abs(ar(0, k)); ip(k)=k
  do i=k+1, min(k+m1, n)
    aik=abs(ar(k-i, i))
    if aik>amax then
      amax=aik; ip(k)=i
  if amax>eps then
    if ip(k)≠k then
      do j=k, min(k+2m1, n)
        ar(j-ip(k), ip(k)) ⇔ ar(j-k, k)
      do j=k+1, min(k+2m1, n)
        wk(j)=ar(j-k, k)
      do i=k+1, min(k+m1, n)
        ar(k-i, i)=-ar(k-i, i)/ar(0, k)
        t=ar(k-i, i)
        do j=k+1, min(k+2m1, n)
          ar(j-i, i)=ar(j-i, i)+t*wk(j)
    else
      ir=ir+1

```

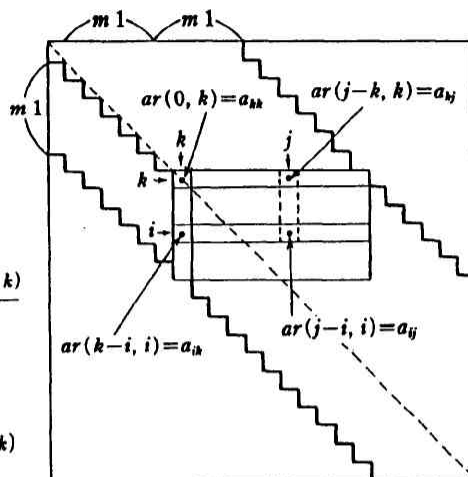


表5 帯ガウス BGLU1, 2, 4 の性能比較例 (sec)

M260H (図情大)				S810/20 (東大)				S810/20 (東大)				S820/80 (東大)			
				86.10 陰なしのコンパイラ				87.10 陰アリのコンパイラ				88.9 陰アリのコンパイラ			
m1/n	BGLU1	BGLU2	BGLU4	m1/n	BGLU1	BGLU2	BGLU4	m1/n	BGLU1	BGLU2	BGLU4	m1/n	BGLU1	BGLU2	BGLU4
50/2550	17.4	12.2	12.1	150/22650	23.7	13.3	8.14	11.6	6.8	6.8		3.12	2.61		

対称帯ガウスと帯コレスキ：対称正値の帯行列に対しては、従来は帯コレスキーが奨励されるのが普通であったが、筆者は対称帯ガウスを推す。部分軸選択なしでよいから2段同時化や2段2行同時化は簡単である。表6参照。ここにコレスキーは、いわゆる平方根なし $U^T D U$ タイプのものに手を加えたものである。にもかかわらず、S820/80

にてベクトル化はしたが、陰アンロールしなかった。

表6： 対称帯ガウスと帯コレスキの比較

M260H (図情大)					S820/80 (東大, 88, 9)				
m1/n	CHDCMP コレスキ	BSLU1	BSLU2	BSLU4	m1/n	CHDCMP	BSLU1	BSLU2	BSLU4
50/2550	4.54	4.70	3.24	3.24	150/22650	4.58	3.65	2.12	1.93

帯ガウスの2段同時となると、ちょっと素人には難しい。BGLU4と、対応するBGSLV4を文献(1)から転載しておくから利用されたい。

```

*
*      帯ガウス・2段2行同時
*
SUBROUTINE BGLU4( N, M1, A, IP, EPS, IR, WK, WK1 )
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(-M1-1:2*M1+1,N+1), IP(N+1), WK(N), WK1(N)
*      初期値の設定
IR = 0
DO 10 J = -M1-1, 2*M1+1
10  A(J,N+1) = 0.0D0
DO 20 I = 1, N+1
  A(-M1-1,I) = 0.0D0
20  A(2*M1+1,I) = 0.0D0
  A(0,N+1) = 1.0D0
*
DO 100 K = 1, N, 2
*
  K1 = K+1
  IMAX = MIN(K+M1,N)
  JMAX = MIN(K+2*M1,N)
  IMAX1 = MIN(K1+M1,N)
  JMAX1 = MIN(K1+2*M1,N)
*      k 段の部分軸選択
  AMAX = ABS(A(0,K))
  IPK = K
  DO 110 I = K+1, IMAX
    AIK = ABS(A(K-I,I))
    IF( AIK.GT.AMAX ) THEN
      IPK = I
      AMAX = AIK
  110  CONTINUE
  IP(K) = IPK
*      k 段の方程式入れ換え
  IF( AMAX.GT.EPS ) THEN
    IF( IPK.NE.K ) THEN
      DO 120 J = K, JMAX
        W = A(J-IPK,IPK)
        A(J-IPK,IPK) = A(J-K,K)
        A(J-K,K) = W
  120  CONTINUE
      END IF
*      α の計算と k+1 列だけの消去変換
      T = A(0,K)
      DO 130 I = K+1, IMAX1
        A(K-I,I) = -A(K-I,I)/T
      130  T = A(1,K)
      DO 140 I = K+1, IMAX
        A(K1-I,I) = A(K1-I,I)+A(K-I,I)*T
      140  ELSE
        IR = IR+1
        IP(K) = K
        DO 150 I = K+1, IMAX1
          A(K-I,I) = 0.0D0
        150  END IF
*      k+1 段の部分軸選択
        AMAX1 = ABS(A(0,K1))
        IPK1 = K1
        DO 200 I = K1+1, IMAX1
          AIK = ABS(A(K1-I,I))
          IF( AIK.GT.AMAX1 ) THEN
            IPK1 = I
            AMAX1 = AIK
        200  CONTINUE
        IP(K1) = IPK1
*      k+1 段の方程式入れ換え
        IF( AMAX1.GT.EPS ) THEN
          IF( IPK1.NE.K1 ) THEN
            DO 210 J = K, JMAX1
              W = A(J-IPK1,IPK1)
              A(J-IPK1,IPK1) = A(J-K1,K1)
              A(J-K1,K1) = W
        210  CONTINUE
          END IF

```

```

*      α の計算
      T = A(0,K1)
      DO 220 I = K1+1, IMAX1
        A(K1-I,I) = -A(K1-I,I)/T
      220  ELSE
        IR = IR+1
        IP(K1) = K1
        DO 230 I = K1+1, IMAX1
          A(K1-I,I) = 0.0D0
        230  END IF
*      k+1 行だけの消去変換
      T = A(-1,K1)
      DO 240 J = K1+1, JMAX1
        WK(J) = A(J-K,K)
        A(J-K1,K1) = A(J-K1,K1)+T*WK(J)
      240  WK1(J) = A(J-K1,K1)
*      Gauss の消去法・2段2行同時
      DO 300 I = K1+1, IMAX1, 2
        T = A(K-I,I)
        T1 = A(K1-I,I)
        U = A(K-I-1,I+1)
        U1 = A(K1-I-1,I+1)
        DO 310 J = K1+1, JMAX1
          A(J-I,I) = A(J-I,I) + T*WK(J)+T1*WK1(J)
          A(J-I-1,I+1) = A(J-I-1,I+1)+U*WK(J)+U1*WK1(J)
        310  CONTINUE
      300  CONTINUE
      100 CONTINUE
      RETURN
      END
*
*      帯行列の右辺の計算・2段同時
*
SUBROUTINE BGSLV4( N, M1, A, B, IP )
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(-M1-1:2*M1+1,N+1), B(N+1), IP(N+1)
*      前進消去
DO 100 K = 1, N, 2
*      k 段の方程式入れ換え
  IPK = IP(K)
  IF( IPK.NE.K ) THEN
    W = B(IPK)
    B(IPK) = B(K)
    B(K) = W
  END IF
*      k+1 段の方程式入れ換え
  K1 = K+1
  IPK1 = IP(K1)
  IF( IPK1.NE.K1 ) THEN
    W = B(IPK1)
    B(IPK1) = B(K1)
    B(K1) = W
  END IF
*      k+1 行だけの消去変換
  B(K1) = B(K1)+A(-1,K1)*B(K)
*      Gauss の消去法・2段同時
  T = B(K)
  T1 = B(K1)
  DO 110 I = K1+1, MIN(K1+M1,N)
    B(I) = B(I)+A(K-I,I)*T+A(K1-I,I)*T1
  110  CONTINUE
*      後退代入・2段同時
  IF( MOD(N,2).EQ.0 ) THEN
    NEND = N
  ELSE
    NEND = N+1
    B(N+1) = 0.0D0
    A(0,N+1) = 1.0D0
  END IF
C*      DO 200 K = NEND, 1, -2
    S1 = -B(K)
    S0 = -B(K-1)
    DO 210 J = K+1, MIN(K+2*M1,N)
      S1 = S1+A(J-K,K)*B(J)
      S0 = S0+A(J-K+1,K-1)*B(J)
    210  B(K) = -S1/A(0,K)
      B(K-1) = (-S0-A(1,K-1)*B(K))/A(0,K-1)
    200 CONTINUE
  RETURN
  END

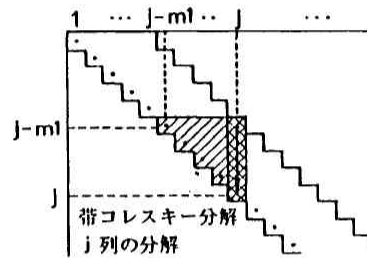
```

```

*
*      帯コレスキー分解
*
SUBROUTINE CHDCMP( N, M1, A, W )
IMPLICIT REAL*8 ( A-H,O-Z )
DIMENSION A(0:M1,0:N), W(N)
*      帯修正コレスキー分解
DO 10 J = 1, N
  DO 20 I = MAX(1,J-M1), J-1
    T = A(I-J,J)
    DO 30 K = MAX(1,J-M1), I-1
30    T = T-W(K)*A(K-I,I)
    W(I) = T
    A(I-J,J) = W(I)*A(0,I)
20  CONTINUE
*
  T = A(0,J)
  DO 40 I = MAX(1,J-M1), J-1
40    T = T-W(I)*A(I-J,J)
    A(0,J) = 1.0/T
10  CONTINUE
  RETURN
END

```

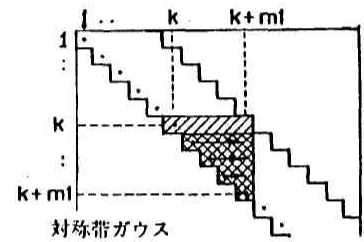
注： ランク落ちの検出を省略
右辺の計算を省略



```

*
*      対称帯ガウス
*
SUBROUTINE BSLU1( N, M1, A, EPS, WK )
IMPLICIT REAL*8 ( A-H,O-Z )
DIMENSION A(0:M1,N), WK(N)
*
DO 110 J = K+1, MIN(K+M1,N)
110  WK(J) = A(J-K,K)
*      Gaussの消去法
DO 120 I = K+1, MIN(K+M1,N)
  T = -A(I-K,K)/A(0,K)
  DO 130 J = I, MIN(K+M1,N)
130    A(J-I,I) = A(J-I,I) + T*WK(J)
120  CONTINUE
100  CONTINUE
  RETURN
END

```



```

*
*      対称帯ガウス・2段2行同時
*
SUBROUTINE BSLU4( N, M1, A, EPS, WK, WK1 )
IMPLICIT REAL*8 ( A-H,O-Z )
DIMENSION A(0:M1+2,N+1), WK(N), WK1(N)
*      初期値の設定
DO 10 I = 1, N+1
  A(M1+1,I) = 0.000
10  A(M1+2,I) = 0.000
  A(0,N+1) = 1.000
  DO 20 J = 1, M1+2
20    A(J,N+1) = 0.000
*
DO 100 K = 1, N, 2
  K1 = K+1
*      k+1行だけの消去変換
  T = -A(1,K)/A(0,K)
  DO 110 J = K+1, MIN(K1+M1,N)
    WK(J) = A(J-K,K)
    A(J-K1,K1) = A(J-K1,K1) + T*WK(J)
    WK1(J) = A(J-K1,K1)
110  CONTINUE
*      Gaussの消去法・2段2行同時
DO 120 I = K1+1, MIN(K1+M1,N), 2
  T = -A(I-K,K)/A(0,K)
  T1 = -A(I-K1,K1)/A(0,K1)
  U = -A(I+1-K,K)/A(0,K)
  U1 = -A(I+1-K1,K1)/A(0,K1)
  A(0,I) = A(0,I) + T*WK(I) + T1*WK1(I)
  DO 130 J = I+1, MIN(K1+M1,N)
    A(J-I,I) = A(J-I,I) + T*WK(J) + T1*WK1(J)
    A(J-I-1,I+1) = A(J-I-1,I+1) + U*WK(J) + U1*WK1(J)
130  CONTINUE
120  CONTINUE
100  CONTINUE
  RETURN
END

```

```

*
*      右辺の計算, 2段同時
*
SUBROUTINE BSSLV4( N, M1, A, B )
IMPLICIT REAL*8 ( A-H,O-Z )
DIMENSION A(0:M1+2,N+1), B(N+1)
*      前進消去
DO 100 K = 1, N, 2
  K1 = K+1
*      k+1行だけの消去変換
  T = -A(1,K)/A(0,K)
  B(K1) = B(K1) + T*B(K)
*      Gaussの消去法・2段同時
  BK = B(K)/A(0,K)
  BK1 = B(K1)/A(0,K1)
  DO 110 I = K1+1, MIN(K1+M1,N)
110    B(I) = B(I) - A(I-K,K)*BK - A(I-K1,K1)*BK1
100  CONTINUE
*      後退代入・2段同時
IF( MOD(N,2).EQ.0 ) THEN
  NEND = N
ELSE
  NEND = N+1
  B(N+1) = 0.000
  A(0,N+1) = 1.000
END IF
DO 200 K = NEND, 1, -2
  S1 = -B(K)
  S0 = -B(K-1)
  DO 210 J = K+1, MIN(K+M1,N)
    S1 = S1 + A(J-K,K)*B(J)
210    S0 = S0 + A(J-K+1,K-1)*B(J)
    B(K) = -S1/A(0,K)
    B(K-1) = (-S0 - A(1,K-1)*B(K))/A(0,K-1)
200  CONTINUE
  RETURN
END

```

非対称帯行列の場合にも、対角優位行列のように部分軸選択不要と分かっているときは、それ専用のプログラムを使うとよい。CPUタイム、メモリ容量共に大幅に節約される。プログラムは上記対称帯よりさらに簡単だからのせない。

他にMartin・Wilkinsonの特殊ガウスがある。線形の時間依存問題に適用するとBLU系より速く、ページスワップも少ない。しかし非線形問題に対しては、かえって遅くなる。速いとか遅いとか言っても差はわずかだから、連立一次方程式の解法用としては必ずしも必要ない。帯行列固有値解法のスツルムカウントには、もうひとつのMartin・Wilkinsonの特殊ガウスが是非必要になるが今回は割愛する。

おわりにひとこと：

コンパイラによる陰アンロールは、密ガウス、普通の帯ガウスに対しては、まず問題なくかかるようである。ただしwk（）を使っておくことが必要である。縦ブロックガウス、対称帯ガウスとなると、ベクトル化処理はするが陰アンロールしないことがあるから注意がいる。Martin・Wilkinsonも同様である。

結局、普通には、2段同時の陽アンロールまでは書いて、あとは陰アンロールにまかせる、ということであろうか。

この章の参考文献：

- | | | |
|--------------|--------------------------|-----|
| (1) 村田・小国ほか | ： 工学における数値シミュレーション(1987) | 丸 善 |
| (2) 村田・小国・唐木 | ： スーパーコンピュータ付録(1985) | 丸 善 |
| (3) 村田・名取・唐木 | ： 大型数値シミュレーション(1990) | 岩 波 |