数理解析研究所講究録
第613巻 1987年 91-108

連立一次方程式の直接解法とスーパーコンピュータ

図書館情報大学 長谷川秀彦(Hidehiko Hasegawa)
村田 健郎(Kenro Murata)

1. はじめに

線形計算の分野では，連立一次方程式 $Ax=b$ を解くことが基本的であり，固有値計算における逆反復法などでもひんぱんに利用される。ここではガウスの消去法に基いた直接解法について，最近広く使われるようになったスーパーコンピュータ向けの解法とはどういうものかを述べる。そこで注意するのは次の3点である。

(1) 精度を落としてスーパーコンピュータに合わせるようなことはしない

(2) スーパーコンピュータだけにとって良いプログラムではなく，汎用計算機にとっても効率的となるようにする

(3) 特定の(メーカーの)スーパーコンピュータだけに効率的なプログラムは除く

2. ガウスの消去法

連立一次方程式 $Ax=b$ を解くためには部分軸選択のガウスの消去法が用いられる。この算法では k-1 段までの消去変換が行なわれた方程式 $A^{(k-1)}x=b^{(k-1)}$ に対して，第 k 段での方程式交換の行列 P_k とガウスの消去変換の行列 G_k を作用させ，

$$G_k P_k A^{(k-1)} x = G_k P_k b^{(k-1)}$$

$$A^{(k)} x = b^{(k)}, \quad A^{(k)} = G_k P_k A^{(k-1)}, \quad b^{(k)} = G_k P_k b^{(k-1)}$$

行列 A の k 列の対角を除いた下三角部分を消去する。この操作を $k=1,\cdots n-1$ までくり返して上三角行列 U を作る。すなわち

$$\underbrace{G_{n-1} P_{n-1} \cdots G_1 P_1 A}_{U} x = \underbrace{G_{n-1} P_{n-1} \cdots G_1 P_1 b}_{b'} \quad (U:\text{上三角行列})$$

となる。方程式交換とガウスの消去変換の全体を表わす行列を G とおけば，$G = G_{n-1} P_{n-1} \cdots G_1 P_1$ となり以下のように書ける。

$$GAx = Gb, \quad GA = U \quad (U:\text{上三角行列}) \qquad (1)$$

(1)は $Ux=Gb$ とも書けるので，$G^{-1}=L$ として LU分解 $A=LU$ ($L=G^{-1}$)と言うこともある。ここでの L に下三角行列という意味はない。

部分軸選択のガウスの消去法のアルゴリズムを図１に示す。

実際には G を作るのではなく，k 段の方程式交換 P_k の情報を $ip(k)$，k 段の消去変換 G_k の情報を a の対角を除いた下三角部分に格納する。

図１のアルゴリズムで，行列 A についての P_k と G_k ($k=1,\cdots n-1$)を

いったん決めておけば，$\mathbb{b}$が与えられたときに $P_1, G_1, \cdots P_{n-1}, G_{n-1}$を順に作用させて$G\mathbb{b}$を作ることができる。この，$G\mathbb{b}$を作る操作を前進消去という。$Ux = G\mathbb{b}$ の Uは上三角行列なので，あとは上三角方程式を後退代入で解けばxが求まる。

$\mathbb{b}$に対するアルゴリズムを図2.に示す。

＊で示した計算の主部 $a'_{ij} = a_{ij} + a_{ik} a_{kj}$ に注目する。図1では最も内側のループで i 行に対する消去変換が行なわれているので「行型ガウス」と呼ぶ。行型ガウスでは，最も内側のループで配列の第二添字が動くため，メモリ参照は連続的でない。

i のループと j のループを交換して，最も内側のループで j 列に対する消去変換を行なう「列型ガウス」を作ることができる。列型ガウスでは，最も内側のループで

```
ir=0
do k=1, n
  amax=|a(k,k)| ; ip(k)=k
  do i=k+1, n
    if |a(i,k)| > amax then
      amax=|a(i,k)|; ip(k)=i
  if amax > ε then
    if ip(k) ≠ k then
      do j=k, n
        a(k,j) ⇄ a(ip(k),j)
    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)
  else
    ir=ir+1 ; ip(k)=k
```

```
do k=1, n
  if ip(k) ≠ k then
    b(k) ⇄ b(ip(k))
  do i=k+1, n
    b(i) = b(i) + a(i,k)*b(k)
do k=n, 1, -1
  b(k) =(b(k) - Σ[j=k+1..n] a(k,j)*b(j))/a(k,k)
```

図1. 部分軸選択のガウスの消去法　図2. $\mathbb{b}$に対するアルゴリズム

配列の第1添字が動くため，メモリ参照が連続的になる。
　仮想メモリ方式の計算機では，ページスワップを減少させて実行時間を短かくするために　列型ガウスを用いる必要がある。行型と列型をそのままプログラムにして，スーパーコンピュータS810/20と汎用計算機M260H上で実行した結果を図3に示す。この例では，仮想メモリ方式の汎用計算機M260H上で，行型のプログラムは$N=900$のとき3倍のCPUタイムが必要となっている。スーパーコンピュータ上でも，列型よりも確実に遅く，ときにはひどく遅くなることがわかる。結局，メモリ参照は連続的でないとよくないことがわかる。（スーパーコンピュータ上で遅くなる理由はバンクコンフリクトのためと考えられる。VP200上では$N=600$で3.9倍になった。）
　これから，列型ガウスをもとにしたスーパーコンピュータ向けの解法を作ることにする。ガウスの消去法の変形にクラウト法があり，その主部は

$$s = s + l_{ik} l_{kj}$$

という内積演算になる。内積はメモリ上のデータ

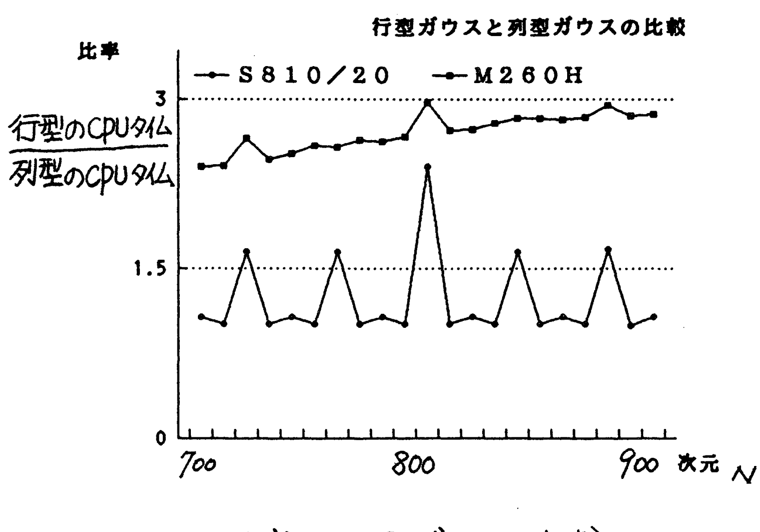

図3.　行型ガウスと列型ガウスの比較

が単精度であっても，レジスタ上の演算結果は倍精度となるため，機械語によってプログラムが書かれた時代には精度が良いとされてよく用いられた。今は混合演算を用いた時以外はメリットがないことに加え，大規模な計算にも向いていない。[2)]　理由は，軸選択がむづかしく，消去の進行とともにメモリ参照が減るのではなく，常にWorking Set が一定だからである。そのため，行型ガウスと同様にメモリ参照が多くなる。特に帯行列用の算法を考えると，帯行列の帯巾が広がってしまう，あるいは間接アドレス使用したステートメント（リストベクトル使用）になる。　表1，2にその結果を示す。LU22W[3),4)]はスーパーコンピュータ向けの対策がとられているので速いがGLU1もそのような対策をとれば，M260Hで約75%，S810/20で約30%のCPUタイムですむ。そうすれば，クラウト法よりも速くなる。また，倍精度計算のため精度についてはどちらも問題ないといえる。データ行列はいずれもフランク行列である。

表1.　M260HでのCPUタイム(s)

次元	行型 GLUROW	列型 GLU1	クラウト法 LU22W
100	0.292	0.2941	0.236
200	2.384	2.361	1.856
300	8.083	7.928	6.081
400	25.35	18.72	14.3
500	41.95	36.71	27.84
600	108.8	63.23	50.66
700	234.9	99.75	85.59
800	450.8	150.7	126
900	615.8	213.7	186.9

表2.　S810/20でのCPUタイム(s)

次元	行型 GLUROW	列型 GLU1	クラウト法 LU22W
100	0.01854	0.01812	0.006249
200	0.1197	0.08895	0.02749
300	0.2608	0.2477	0.08062
400	0.801	0.5281	0.1497
500	1.012	0.9506	0.2929
600	2.509	1.575	0.4947
700	2.566	2.41	0.7541
800	8.379	3.49	0.9595
900	5.221	4.877	1.819
1,000	10.94	6.55	2.115

3. スーパーコンピュータのために

列型ガウスの主部は以下のようになる。[5][6]

```
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)
```

ここではアンダーラインをつけた3つの変数が最も内側のループで変化する。これをパイプライン制御のスーパーコンピュータでは，ロードパイプライン2本，ストアパイプライン1本，乗算・加算パイプライン1本で処理を行なう。一般にはパイプラインを遊ばせないように最も内側のループ内に演算をつめこめば良いとされている。そこで，アンローリングを2重に行なうと

```
do  j = k+1, n, 2
    do  i = k+1, n
        a(i,j)   = a(i,j)   + a(i,k) * a(k,j)
        a(i,j+1) = a(i,j+1) + a(i,k) * a(k,j+1)
```

。

このプログラムでは，ロードパイプ3~4本，ストアパイプ2本，乗算・加算パイプ2本は必要である。そのため，S810/10ではロードパイプが不足するため，2倍にはなり得ない。S810/20，VP200ならばこれだけの対策で2倍になると，理論上は

いえる。また，この程度のアンローリングならばコンパイラの能力によっては自動的に行なわれる場合もあるので，パイプの不足が起こらない機種ならばそれでも良い。しかし，ロードパイプ，演算パイプは余っているのに，ストアパイプの不足のために性能が上がらないのは不満であろう。

そこでアルゴリズムから考え直す。ストアパイプと一度に消去変換を行なう列の数は対応しているので，右辺の演算量だけをふやすようにしてやればよい。そのために変えられるのは k だけで，２つ外側のループに対するアンローリングと考えることもできる，２段同時の列ガウスに到達する。２段同時に対応させて，j に関するアンローリングを２列同時と呼ぶ。

２段同時の列ガウスの主部は

```
do j = k+1, n
  do i = k+1, n
    a(i,j) = a(i,j) + a(i,k) * a(k,j) + a(i,k+1) * a(k+1,j)
```

。

このプログラムでは，ロードパイプ３本，ストアパイプ１本，乗算·加算パイプ２本となってS810/10でも２倍の速さが期待できる。２段同時のアルゴリズムを図４に示す。この対策はアルゴリズムを考えなければ不可能であり（たとえば，ランク落ちならば無変換になど），コンパイラのベクトル化能力

に頼っていたのでは無理であろう。同様に3段，4段を考えても良いがプログラムを作成するのは人間であることを思えば，これらも非現実的だといえよう。

do $k=1, n, 2$

① k段の部分軸選択 : ipk

② $a_{kk} \rightleftarrows a_{ipk,k}$

$a_{k,k+1} \rightleftarrows a_{ipk,k+1}$ } k列と$k+1$列に方程式の入れ換え

③ k段の$\alpha = -a_{ik}/a_{kk}$を作る. a_{ik}に入れる。

$k+1$列の消去変換

→ ランク落ちならば無変換に

①' $k+1$段の部分軸選択 : $ipk1$

②' $a_{k+1,k+1} \rightleftarrows a_{ipk1,k+1}$

$a_{k+1,k} \rightleftarrows a_{ipk1,k}$ } (k列と)$k+1$列に方程式の入れ換え

* k列には③のαが入っている

③' $k+1$段の$\alpha' = -a_{i,k+1}/a_{k+1,k+1}$を作る。$a_{i,k+1}$に入れる.

→ ランク落ちならば無変換に

do $j = k+2, n$

k段の方程式の入れ換え

$k+1$段の方程式の入れ換え

$k+1$行にk段の消去変換を ②'が役に立つ.

do $i = k+2, n$

$a_{ij} = a_{ij} + \alpha_i a_{kj} + \alpha'_i a_{k+1,j}$

図4. 2段同時の列ガウスのアルゴリズム

2段同時にしただけではパイプが余っている高級なスーパーコンピュータでは，さらに j に関するアンローリングを行なって2段2列同時にする。そのときの主部は，

```
do j = k+1, n, 2
  do i = k+1, n
    a(i,j)   = a(i,j)   + a(i,k)*a(k,j)   + a(i,k+1)*a(k+1,j)
    a(i,j+1) = a(i,j+1) + a(i,k)*a(k,j+1) + a(i,k+1)*a(k+1,j+1)
```

このプログラムだと，ロードパイプ4~6本，ストアパイプ2本，乗算・加算パイプ4本となってS810/20では4倍の速さが期待できる。

表3. アルゴリズムの特徴とCPUタイム

	ロードパイプ	ストアパイプ	演算パイプ	S810/10	S810/20	VP200	M260H
列ガウス	2	1	1	1	1	1	1
2列同時	3~4	2	2	$1+\alpha$倍	2倍	2倍	$1+\beta$倍
2段同時	3	1	2	2倍	2倍	2倍	$1+\beta'$倍
2段2列同時	4~6	2	4	$2+\alpha'$倍	4倍	4倍	$1+\beta''$倍

表4~7に各機種での実測値を示す。S810/10は気象研究所，S810/20は東大大型計算機センター，VP200は京大大型計算機センター，M260Hは図書館情報大学で各々実測した。NUMPACは，LEQLUWを用いた。文献7)8)によれば，LEQLUWはVP200上では2重，S810上では8重のアンローリングを施したクラウト法である。

表4.　S810/10での各種プログラムの実測値(s)

次元	列ガウス GLU1	2段同時 GLU2	2段2列 GLU4	クラウト法 NUMPAC
100	0.01833	0.01312	0.01145	0.012
200	0.08937	0.05999	0.05354	0.052
300	0.2491	0.1579	0.1443	0.142
400	0.5318	0.3264	0.2999	0.307
500	0.9535	0.5741	0.5297	0.556
600	1.572	0.9306	0.8597	0.92
700	2.4	1.397	1.294	1.417
800	3.471	1.998	1.854	2.066
900	4.836	2.752	2.559	2.857
1,000	6.497	3.666	3.414	3.884

表5.　S810/20での各種プログラムの実測値(s)

次元	列ガウス GLU1	2段同時 GLU2	2段2列 GLU4	クラウト法 NUMPAC	LU22W
100	0.01812	0.01312	0.009583	0.01	0.006249
200	0.08895	0.05958	0.04083	0.041	0.02749
300	0.2477	0.1533	0.1008	0.102	0.08062
400	0.5281	0.3162	0.2014	0.204	0.1497
500	0.9506	0.5558	0.3443	0.349	0.2929
600	1.575	0.9006	0.5564	0.552	0.4947
700	2.41	1.355	0.8135	0.814	0.7541
800	3.49	1.944	1.149	1.154	0.9595
900	4.877	2.713	1.591	1.572	1.819
1,000	6.55	3.592	2.087	2.091	2.115

表6.　VP200での各種プログラムの実測値(s)

次元	列ガウス GLU1	2段同時 GLU2	2段2列 GLU4	クラウト法 NUMPAC
100	0.01289	0.01403	0.00888	0.009
200	0.05523	0.06088	0.03786	0.032
300	0.1421	0.1467	0.09216	0.073
400	0.2916	0.2883	0.1857	0.147
500	0.5192	0.4979	0.327	0.264
600	0.8471	0.7939	0.545	0.42

表4を見れば，S810/10では2段同時にすると約2倍，2段2列同時にしても変わらないことがわかる。また8重のアンローリングを施したクラウト法と2段同時が同じ速度であることもわかる。このことから，スーパーコンピュータ向けの

改良で大事なのは，アンローリングの多重度ではなく，理屈に基いたアルゴリズム上での改良であることがわかる。

表5を見れば，S810/20の場合には2段同時にすると約2倍，2段2列同時にするとさらに約2倍となり，元々の列ガウスの約4倍まで改良することができる。クラウト法LU22wは，行列が小さい時には良いが，次元が800をこえると遅くなる。

表6を見れば，VP200ではS810/10，S810/20とだいぶ違っていることがわかる。列ガウスGLU1と2段同時GLU2がほとんど同じであり，コンパイラがアンローリングを行なってGLU1が2列同時のプログラムとして測定されたものと考えられる。図5のAを測定するつもりが，コンパイラによってアンローリングを受けBを測定しまったという訳である。VP200ではバンクコンフリクトの影響が大きいのにもかかわらずクラウト法が速いのは，制限3項演算 $a_i = a_i + tb_i$ が内積演算 $s=s+a_ib_i$

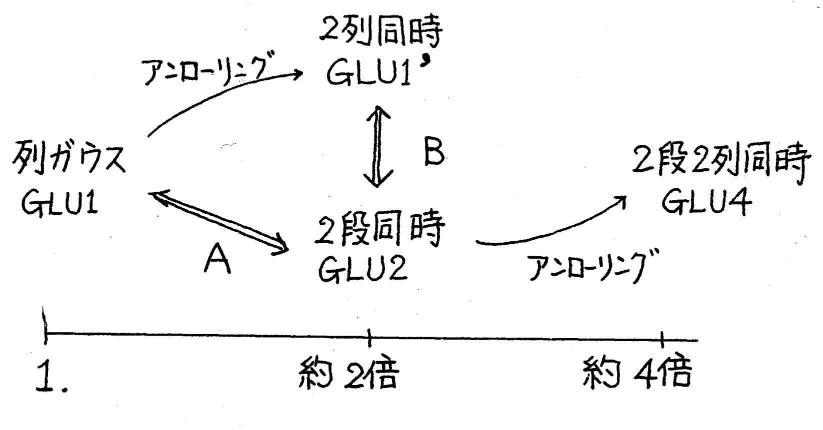

図5. アルゴリズム相互の関係

よりも遅いためかと思われるが，詳細はわからない。

表7を見れば，M260Hでも2段同時にすると約1.2倍，2段2列同時にすると約1.1倍ほど速くなる。汎用計算機上ではループの判定にも多くのCPUタイムを必要とするため，ループの判定を減らせばそれだけ速くなる。汎用計算機の場合にはループの判定の回数による速度向上なので，2段同時でも2列同時でも向上の程度は同じである。汎用計算機上でのクラウト法はループ内の演算を多くしてもメモリ参照が多いため遅い。

これらのことから

(1) 2段2列同時はどの機種にとってもよい

(2) クラウト法は汎用計算機や大規模行列にとっては遅いことがわかる。2段2列同時のプログラムを付録につけるが，この程度の複雑さで汎用に使えるのであれば，改良を行なう価値はじゅうぶんにあるだろう。

表7.　M260Hでの各種プログラムの実測値(s)

次元	列ガウス GLU1	2段同時 GLU2	2段2列 GLU4	クラウト法 LU22W
100	0.2941	0.2535	0.2164	0.236
200	2.361	1.89	1.723	1.856
300	7.928	6.384	5.697	6.081
400	18.72	14.87	13.42	14.3
500	36.71	29.39	26.99	27.84
600	63.23	50.69	45.31	50.66
700	99.75	79.66	72.35	85.59
800	150.7	127.4	106.6	126
900	213.7	171	154.5	186.9

4. 上階段化ガウス，縦ブロックガウス，帯ガウス

列ガウスはランク落ちがなければ $Ax=b$ を解くことができる。上階段化ガウスはランク落ちを正しく決定し，Aのランク落ちに対応した $b_i=0$ とした特解を求めることができる。[1] この上階段化ガウスに対しても２段同時，２段２列同時は有効である。そして，そのプログラムはS810/10，S810/20では列ガウスと同程度に，M260Hでは列ガウスより若干遅い程度で動く。

縦ブロックガウスは，仮想メモリ方式の計算機でメモリ参照を効率的に行なうために作られた算法である。[2)5)6)] スーパーコンピュータ上で超大型の行列を拡張記憶（半導体ディスク）を用いて処理しようとするときに使用すると，拡張記憶とのデータ交換が少なくて良い。ここでは拡張記憶は使わずに，列ガウスと上階段化ガウスを縦ブロックガウスにして測定した。ブロックの大きさは40列に固定したため，S810/20上ではほぼ同程度，２段２列同時でブロックのときに10%程度遅くなることがあった。S810/10，VP200では全く同程度であった。M260Hでは次元が200~300のとき，ブロックガウスがわづかに速かった。これは，測定するためにプログラムに割り当てられた実メモリ量とブロック２つ分のWorking Setが一致したためと考えられる。

帯ガウスは，行型ガウスを $a(j-i,i)' = a(i,j)$ という変換で移した a(-M1:2*M1,N)' という長方形領域で行なうものである。(変換先のメモリで考えれば列ガウス)結果を表8に示す。このときも2段同時，2段2列同時が有効である。

帯行列に対する上階段化ガウスも，2段同時，2段2列同時が可能であるが，まだできていない。

5. まとめ

スーパーコンピュータ向けと称して，特殊なアルゴリズムとやみくもなアンローリングが使われている。アルゴリズムを少し考えることによって，精度の保証されたガウスの消去法でじゅうぶんな性能がひきだせた。そして，このプログラムは汎用計算機上でもじゅうぶんに速い。数値計算ライブラリとしては，汎用的で精度の保証されたプログラムである必要があるが，このプログラムはその条件を満足していよう。

右辺に対する処理についても同じようなことは可能で，実際に効果をあげるが，オーダーが違うのでここでは左辺のLU

表8. 帯ガウスの実測結果(ms)

		S810/20			M260H		
		列ガウス	2段同時	2段2列	列ガウス	2段同時	2段2列
M1	N	BGLU1	BGLU2	BGLU4	BGLU1	BGLU2	BGLU4
10	230	10.4	7.4	6.4	75.4	58.5	58.9
20	860	73.9	45.6	36.2	1,000	723	725
30	1,890	250.2	145.4	110.2	4,799	3,368	3,389
40	3,320	610.8	344.9	252.7	14,712	10,728	10,436
50	5,150	1,262	696	497.9	約35秒	約25秒	
60	7,380	2,291	1,241	861.4	約73秒	約51秒	

分解だけをとりあげた。

6. 謝辞

NUMPACの実測値は中京大学秦野甯世さんから頂きました。深く感謝致します。

7. 文献

1) 村田健郎．線形代数と線形計算法序説．東京，サイエンス社，1986，vii，225p．

2) 村田健郎．物理学特別講義 昭和55年夏学期－輸送現象における有限要素解析－．
〔東京，東京大学理学部，1980〕22，40，27，34p．（講義用テキスト）

3) 村田健郎，小国力，唐木幸比古．スーパーコンピュータ．東京，丸善，1985，vi，304p．

4) 呉永化．VP向き二列同時消去LU分解法．第2回ベクトル計算機応用シンポジウム論文集．
京都，1986-3，京都大学大型計算機センター．p.72-77．

5) 村田健郎．スーパーコンピュータと線形計算．1985年秋季総合分科会 応用数学分科会講演
アブストラクト．富山，1985-9/10，富山大学．〔東京，日本数学会，1985〕p.113-126．

6) 村田健郎．"スーパーコンピュータと線形計算"．コンピュータと数学5：コンピュータから生まれた新しい
数学．野崎昭弘，廣瀬健編．東京，日本評論社，1986，p.193-208（別冊数学セミナー）

7) 秦野甯世，二宮市三．"数学ライブラリNUMPACのスーパー・コンピュータ版"．数値解析研究会資料
No.18．東京，情報処理学会，1986，86-NA-18，p.1-8．（情報処理学会研究報告 Vol.86，No.68

8) 二宮市三．スーパコンピュータと数学ライブラリ．情報処理．Vol.27，No.11，p.1235-1241（1986）

8. 付録

```
*        連立一次方程式の直接解法とスーパーコンピュータ
*              図書館情報大学　長谷川秀彦・村田健郎
*
*                   密行列のＬＵ分解・２段２列同時
      SUBROUTINE GLU4( 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+1), WK2(N+1)
```

```
*                       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
        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
*                       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
           DIVA = 1.0D0/A(K,K)
           DO 120 I = K+1, N
  120        A(I,K) = -A(I,K)*DIVA
           DO 130 I = K+1, N
  130        A(I,K1) = A(I,K1)+A(I,K)*A(K,K1)
        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
         DIVB = 1.0D0/A(K1,K1)
         DO 220 I = K1+1, N
  220      A(I,K1) = -A(I,K1)*DIVB
      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)
  290   WK2(I) = A(I,K1)
*
      DO 300 J = K1+1, N, 2
*                    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
           W = A(IPK,J+1)
           A(IPK,J+1) = A(K,J+1)
           A(K,J+1) = 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
           W = A(IPK1,J+1)
           A(IPK1,J+1) = A(K1,J+1)
           A(K1,J+1) = 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)
        A(K1,J+1) = A(K1,J+1)+A(K1,K)*A(K,J+1)
*                    Gaussの消去法・2段2列同時
        T = A(K,J)
        T1 = A(K1,J)
        U = A(K,J+1)
        U1 = A(K1,J+1)
        DO 310 I = K1+1, N
          A(I,J) = A(I,J)+WK1(I)*T+WK2(I)*T1
  310     A(I,J+1) = A(I,J+1)+WK1(I)*U+WK2(I)*U1
  300   CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
*
*                         右辺の計算・前進２段，後退２段
*
      SUBROUTINE GSLV4( N, A, B, IP, IR )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N+1,N+1), B(N+1), IP(N+1)
*
      IF( IR.EQ.0 )  THEN
*                         初期値の設定
         B(N+1) = 0.0D0
*
         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(K1,K)*B(K)
*                         Gaussの消去法・２段同時
           T = B(K)
           T1 =B(K1)
           DO 110 I = K1+1, N
  110        B(I) = B(I)+A(I,K)*T+A(I,K1)*T1
  100    CONTINUE
*                         後退代入・２段同時
         IF( MOD(N,2).EQ.0 )  THEN
            NEND = N
            B(N) = B(N)/A(N,N)
         ELSE
            NEND = N+1
            B(NEND) = 0.0D0
         END IF
         B(NEND-1) = (B(NEND-1)-A(NEND-1,NEND)*B(NEND))/A(NEND-1,NEND-1)
         DO 200 K = NEND-2, 1, -2
           T1 = B(K+1)
           T2 = B(K+2)
           DO 210 I = 1, K
  210        B(I) = B(I)-A(I,K+1)*T1-A(I,K+2)*T2
           B(K) = B(K)/A(K,K)
           B(K-1) = (B(K-1)-A(K-1,K)*B(K))/A(K-1,K-1)
  200    CONTINUE
      END IF
      RETURN
      END
```