プログラムのページ

担当　森口繁一

6207. ガウス消去法

高田　勝（東大工学部）

係数 a_{ij} が同じである m 組の n 元連立一次方程式

$$\sum_{j=1}^{n} a_{ij}x_{jk}=a_{i,n+k},\ (i=1,\ 2,\ \cdots\cdots,\ n;\quad k=1,\ 2,\ \cdots\cdots,\ m)$$

をガウス消去法で解く．右辺をふくめた行列 $[a_{ij}]$ $(j=1,\ 2,\ \cdots\cdots,\ n+m)$ を与えたとき，もとの係数行列のところにはその逆行列が，右辺ベクトルのあった所にはその解ができる．コンパイルを考えて二つだけ余計な記憶装置を用いるほか，$n\times(n+m)$ だけの記憶装置ですむ．ただし簡単のため計算途中で枢軸要素はすべて0にならないとしている．入出力は省略．

```
procedure COMPACTGAUSS (A, N, M);
  value N, M; array A; integer N, M;
comment This procedure computes the inverse
  of an N×N matrix A[1 : N, 1 : N] and a set
  of solutions of simultaneous linear equations
  Σ(J=1..N) A[I, J]×X[J, K]=A[I, N+K], where I=1,
  2, ……, N and K=1, 2, ……, M. The inverse
  matrix replaces A[1 : N, 1 : N] and the solu-
  tions X[1 : N, 1 : M] replace A [1 : N, N+1 :
  N+M];
begin integer I, J, K; real D, E;
  for K : =1 step 1 until N do
  begin D:=A[K, K]; for J:=1 step 1 until
    N+M do if J≠K then begin E:=A[K, J]
    : =A[K, J]/D; for I:=1 step 1 until N
    do if I≠K then A[I, J]:=A[I, J]−E×A
    [I, K] end;
  D : =A[K, K]: =1/D; for I:=1 step 1 until
    N do if I≠K then A[I, K]:=−A[I, K]×D
end end
```

〔注 1〕 次の数値例で手順を理解されたい．N=3, M=1.

$$A[1:3,\ 1:4]=\begin{pmatrix} 2 & 5 & 4 & 6\\ 3 & 8 & 2 & 1\\ 5 & 10 & 14 & 15\end{pmatrix}.$$

	D				E
(2)	0.5	2.5	2.0	3.0	(2.5)
0.5	−1.5	0.5	−4.0	−8.0	(2.0)
	−2.5	−2.5	4.0	0	3.0

(0.5)	8.0	−5.0	22.0	43.0	(−3.0)
2.0	−3.0	2.0	−8.0	−16.0	(−8.0)
	−10.0	5.0	−16	−40.0	−16.0
(−16.0)	−5.75	1.875	1.375	−12.0	(0.625)
−0.0625	2.0	−0.5	−0.5	4.0	(−0.3125)
	0.625	−0.3125	−0.0625	2.5	2.5

〔注 2〕 この算法は，はじめ枢軸要素をふくむ1列だけをとり出して行なう方法であったが，高橋秀俊教授の提案でこのように改めた．なお，上記プログラム中の D, E も不要かと思うが，コンパイルにはこれを用いた方がよい．

（**編集部注**） 6207 のプログラムは **procedure** 宣言の形をしています．これを使って実際の計算を行なうには，たとえば次のようにすればよろしい（N=10, M=5 の場合の例）．

```
begin<上の procedure 宣言全体をここへ入れる>
begin integer I, J; array A[1 : 10, 1 : 15];
  for J:=1 step 1 until 15 do
  for I:=1 step 1 until 10 do
  READ (A[I, J]);
  COMPACTGAUSS (A, 10, 5);
  for I:=1 step 1 until 10 do
begin CRLF; for J:=1 step 1 until 5 do PRINT
  (A[I, J]);
  CRLF; for J:=6 step 1 until 10 do PRINT
  (A[I, J]);
  CRLF; for J:=11 step 1 until 15 do PRINT
  (A[I, J])
end end end
```

6208. 割当問題

高田　勝（東大工学部）

n 人にそれぞれ一個所ずつ n 個所に行かせるのに最も費用が少くてすむようにしたいような問題が割当問題であるが，これは Hitchcock 型の輸送問題で極度に退化したものとして扱われる．この算法は輸送問題における伊理の解法（伊理正夫：輸送問題および割当問題の新しい解き方，経営科学，**3**, 4, (1960), pp. 190～206，その他注2参照）によっている．

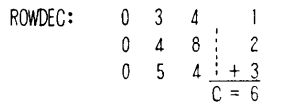
一組の割当に要する費用 d_{ij} $(i=1,\cdots\cdots,\ n;\ j=1,\ \cdots\cdots,\ n)$ の行列を $[d_{ij}]$，割当の行列を $[x_{ij}]$ $(x_{ij}=0$ または1）とするとき，問題は $[x_{ij}]$ の1なる要素に対応する $[d_{ij}]$ の要素の和を最小にするような $[x_{ij}]$ およびそのときの d_{ij} の和を求めることである．d_{ij} は実数であるが，整数としてとり扱ってもよいか

ら，以下のプログラムでは整数としている．数値例については注1を参照のこと．プログラム中 $[d_{ij}]$ は D, n はN, $[x_{ij}]$ は X, 費用はCで表わしている．また INF としては9を12ケタならべたが，これは非常に大きい数の意味で用いた．

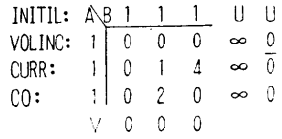
```
procedure ASSIGN (D, N, X, C);
  value N; integer N, C; integer array D, X;
begin integer E, F, I, J, K, P, Q, R, S, W,
  INF, UMIN;
  integer array A, B, L, M, U, V [1 : N];
  INF:=9999 9999 9999; C:=0;
CURDEC: for I:=1 step 1 until N do
  ROWDEC: begin W:=INF;
    for J:=1 step 1 until N do
    begin F:=D[I, J]; if W>F then W:=F
    end;
    if W≠0 then
      begin for J:=1 step 1 until N do
        D[I, J]:=D[I, J]−W; C:=C+W
      end
    end ROWDEC;
  for J:=1 step 1 until N do
COLDEC: begin W:=INF;
  for I:=1 step 1 until N do
    begin F:=D[I, J]; if W>F then
    W:=F end;
  if W≠0 then
    begin for I:=1 step 1 until N do
      D[I, J]:=D[I, J]−W; C:=C+W
    end
  end COLDEC, CURDEC;
for I:=1 step 1 until N do
 INITIL: begin U[I]:=INF; V[I]:=0; A[I]:=
  B[I]:=1;
  for J:=1 step 1 until N do X[I, J]:=0
  end INITIL;
VOLINC:V 0: E:=0;
  for I:=1 step 1 until N do
    begin S:=U[I];
      for J:=1 step 1 until N do
      begin W:=D[I, J]+V[J]; if S>W then
        begin S:=W; E:=E+1 end
      end; U[I]:=S
    end V 0;
    if E=0 then go to CURR;
 V1: E:=0;
  for J:=1 step 1 until N do
    begin S:=V[J];
      for I:=1 step 1 until N do
      begin if X[I, J]=0 then W:=INF else
      W:=U[I]−D[I, J]; if S>W then
        begin S:=W; E:=E+1 end
      end; V[J]:=S
    end V1;
    if E≠0 then go to V 0; comment End
    voltage increasing step;
CURR : C 0 : UMIN:=INF;
  for I:=1 step 1 until N do
    begin if A[I]≠0∧U[I]<UMIN then
      begin UMIN:=U[I]; E:=I end
    end C 0;
 C1 : K:=1; Q:=0;
    KK : L[K]:=E;
    J:=Q;
 JJ: if J=N then J:=1 else J:=J+1;
    if U[E]−V[J]≠D[E, J] then go to JJ;
  Q:=M[K]:=J; if B[Q]≠0 then go toC 2;
  I:=E;
 II: if I=N then I:=1 else I:=I+1;
  if U[I]−V[Q]≠D[I, Q]∨X[I, Q]=0 then
  go to II;
  E:=I; K:=K+1; go to KK;
C 2 : R:=K; P:=L[I];
  C:=C+U[P]; A[P]:=B[Q]:=0;
  for K:=1 step 1 until R do
FLOW: begin integer P1; P:=L[K]; Q:=M[K];
    X[P, Q]:=1;
    if K<R then begin P 1:=L[K+1];
    X[P 1, Q]:=0 end
  end FLOW;
TEST: for I:=1 step 1 until N do
    if A[I]≠0 then go to PŌP; go to EXIT;
PŌP : P:=L[1]; UMIN:=U[P];
  for I:=P step 1 until N do
    begin if U[I]=UMIN∧A[I]≠0 then
      begin E:=I; go to C1 end
      end
  for I:=1 step 1 until N do
```

```
    begin U[I]:=INF; if B[I]=0
      then V[I]:=INF
      else V[I]:=0
    end;
  go to VOLINC;
EXIT:
end
```

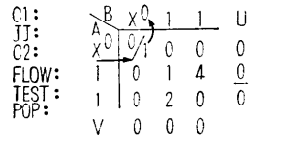
〔注 1〕 例: N=3, D= $\begin{pmatrix} 1 & 4 & 5 \\ 2 & 6 & 10 \\ 3 & 8 & 7 \end{pmatrix}$ について はつぎのとおり.



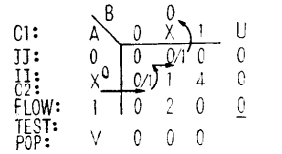
COLDEC: 0 0 0
0 1 4
0 2 0
3+4=7
C=6+7=13.



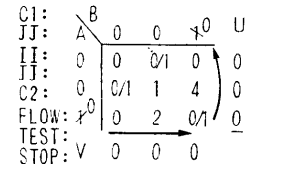
UMIN=0, I=1,
E=1(CURR),
C=13.



P=L[1]=1, K=1,
E=2(POP),
C=13.



L[1]=2, M[1]=1,
P=L[2]=1, Q=M[2]=2,
K=2,
E=3(POP),
C=13.



P=L[1]=3, Q=M[1]=3,
K=1,
C=13.

〔注 2〕 文献

M. Iri, J. Operations Research Soc. of Japan, **3** (1960), pp. 27～87.

高田, オペレーションズリサーチ誌 **3**, 1 (1960), pp. 3～9.

高田, 情報処理, **1**, 4 (1960), pp. 191～198.

おことわり

前号の本欄のプログラム中に誤りがありましたので, 誤植と共にここで訂正します.

6201. (誤) データ A, B から読み取り, ……
(正) データ A, B をテープから読み取り, ……

6202. (誤) MIN:=1,000; (正) MIN:=1000

6203. (誤) **for** I=1　　(正) **for** I:=1

6204. PREP: READ(N); の次に **begin** を置き, そのあとへ **array** A[0 : N]; を入れ, PREP: の前の **array** A [0 : N]; を削る. 最後の **end end** のところへ, もう一つ **end** を追加する. なお注 2 は **begin** と **end** が 1 対追加されたのに応じて修正する必要がある.

6205. 上の 6204 の訂正と同じ.

6204 と 6205 の訂正の理由は, 第 1 に **array** A[0 : N] という宣言が行なわれるときには N の値がきまっていなければならないので, これは READ(N) のあとでなければならないこと, 第 2 に, 宣言は必ずブロックの頭に **begin** を先頭にしてならべねばならないので, この **array** 宣言を READ(N) という実行命令にすぐ続けるわけにいかず, 新たに **begin** を入れて新しいブロックを起こさねばならないこと, そしてこの **begin** に応じて最後の **end** を 1 個ふやさねばならないという次第です.

(森口 繁一)