a cura di Valter Di Dio

È forse la prima volta che viene pubblicato un programma per la risoluzione di sistemi lineari rettangolari, altri programmi sono stati pubblicati (generalmente in Basic), ma necessitavano di un sistema già «preparato».
Preparare il sistema significa fare la maggior parte del lavoro, soprattutto la parte dove è più facile commettere errori. Questo programma invece accetta un sistema scritto nel modo tradizionale e calcola le infinite alla m-n soluzioni.
La cosa strana è che il lettore si sia fermato qui. Infatti la semplificazione e la ricerca delle soluzioni di un sistema non quadrato non è che il primo passo nella soluzione di un gran numero di problemi che ricadono sotto il nome di Teoria dell'Ottimizzazione.
L'ottimizzazione consente di selezionare, tra le infinite soluzioni, quella che minimizza una determinata funzione che prende il nome di funzione obiettivo. Per fare un esempio dobbiamo scegliere il numero degli inservienti ai piani di un albergo in funzione del numero di stanze, del numero di piani, della stagione ecc. ecc. La funzione obiettivo sarà allora la relazione che lega le variabili (stanze, piani, addetti ecc.) al relativo costo. Ottimizzare la soluzione vuol dire scegliere tra tutte le soluzioni possibili del sistema quella che riduce i costi. Magari prossimamente torneremo a parlare più dettagliatamente e con più rigore di questa interessantissima branca della matematica «computerabile»

v.d.d.

Math Tool S

di Luca Padovan - Imola (BO)

Soluzione di sistemi non quadrati

Chi stesse pensando di trovarsi di fronte all'ennesimo programma per risolvere i sistemi di equazioni lineari, avrebbe solo parzialmente ragione. Math-Tool S in effetti, risolve questi sistemi, ma in maniera particolare rispetto ai suoi numerosi predecessori.

Generalmente questi programmi risolvono solo sistemi «quadrati» (n incognite e n equazioni). Inoltre, non si preoccupano di usare virgole e doppia precisione. Cosicché, un innocuo sistemino come:

7x + 3y = 5
2x − y = 2

dà come soluzioni delle strane cose del tipo:

x = 0.846153846
y = 0.307692307

brutte sia da vedere che da trattare.

Il motivo per cui nessuno abbia mai pensato a scrivere un programmino che dia come soluzione del sistema precedente la coppia di valori:

x = 11/13
y = − 4/13

non è, come si potrebbe pensare che il detto programma sia troppo difficile da mettere a punto; la verità è che nel mondo reale, un programma del genere non servirebbe assolutamente a nessuno.

Infatti, chi ha bisogno di risolvere sistemi lineari, è ormai abituato a trattare e dominare fino alla ventiseiesima cifra decimale e anche più; inoltre, per una peculiare caratteristica dei fenomeni naturali, praticamente tutti i sistemi che servono a risolvere problemi «reali» (per esempio di fisica), sono quadrati. Chi invece ha bisogno di gestire frazioni e sistemi rettangolari (liceali, matricole delle varie facoltà scientifiche), deve risolvere problemi così banali, da non richiedere un computer.

Credo comunque che Math-Tool S possa essere di qualche interesse, se non altro come curioso gadget da mostrare agli amici reduci da un esame di algebra lineare.

Un po' di teoria

Non è senza un moto di disgusto per me stesso che mi accingo a introdurre un po' di teoria. Colgo l'occasione per scusarmi con i matematici veri.

A è combinazione lineare di n termini $v_1, ..v_n$ se è valida la relazione:

$A = k_1v_1 + ... + k_nv_n$

dove i $k_1..k_n$ sono generiche costanti. Una base dell'insieme I è un sottinsieme di I che «genera» tutti gli elementi dell'insieme stesso; per esempio se R^3 è l'insieme delle triplette (x,y,z) di numeri reali, allora una sua base è formata dai tre elementi (1,0,0) (0,1,0) (0,0,1).

Ogni tripletta si ottiene come combinazione lineare di quei tre termini. Per gli scettici:

(15,27,1) = 15*(1,0,0) + 27*(0,1,0) + 1*(0,0,1)

Un sistema di m equazioni in n incognite è un insieme di equazioni del tipo:

$a_{11}x_1 + \quad ..... \quad + a_{1n}x_n = B_1$
. . .
. . .
$a_{m1}x_1 + \quad ..... \quad + a_{mn}x_n = b_m$

dove gli x_1, si dicono incognite, gli a_{ij} si dicono coefficienti e i b_i sono i termini noti.

Abbiamo cioè una n-upla $(x_1,..,x_n)$, una m-upla $(b_1,..,b_m)$ e una matrice dei coefficienti (a_{ij}).

Scopo del «gioco» è ricavare degli $x_1,..,x_n$ in modo da verificare le m equazioni dati la matrice (a_{ij}) e il termine noto.

Se il sistema è quadrato, cioè se m=n si dimostra che il sistema ha sempre una e una sola soluzione. Al contrario, se le incognite sono più delle equazioni (n>m) allora un noto teorema ci dice che le soluzioni sono infinite e date dalla relazione:

$z + k_1N_1 + ... + k_rN_r$

dove $Z=(z_1,..,z_n)$ è la Soluzione particolare, $k_1,..,k_r$ sono generiche costanti e $N_1,..,N_r$ sono $r = n - m$ soluzioni del cosiddetto sistema associato:

$a_{11}x_1 + \quad ..... \quad + a_{1n}x_n = 0$
. . .
. . .
$a_{m1}x_1 + \quad ..... \quad + a_{mn}x_nx_n = 0$

che 'generano' tutte le infinite soluzioni (sono cioè una base dello spazio delle soluzioni).

La soluzione particolare è una qualsiasi soluzione del sistema di partenza:

$a_{11}x_1 + \quad ..... \quad + a_{1n}x_n = b_1$
. . .
. . .
$a_{m1}x_1 + \quad ..... \quad + a_{mn}x_n = b_m$

Quindi una equazione a tre incongnite avrà una soluzione generale data da:

S + kB + hC

dove B e C sono due soluzioni generiche del sistema associato. Ma come si trovano queste soluzioni? Il metodo di Gauss-Jordan fa al caso nostro. Data la matrice A se essa è quadrata, si diagonalizza, cioè si annullano tutti i termini tranne quelli sulla diagonale.

È disponibile presso la redazione, il disco con il programma pubblicato in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 249.

Alcuni teoremi ci assicurano che il nuovo sistema ha le medesime soluzioni di quello di partenza. Per diagonalizzare la matrice il metodo è semplice: si moltiplicano e sottraggono le righe del sistema in maniera da annullare alcuni coefficienti.

Per esempio:

$$\begin{cases} x+2y=3 \\ 2x+y=1 \end{cases}$$

riga2−2*riga1

$$\begin{cases} x+2y=3 \\ -3y=-5 \end{cases}$$

2*riga2+3*riga1

$$\begin{cases} -3x=1 \\ -3y=-5 \end{cases}$$

da cui:

$$\begin{cases} x=-1/3 \\ y=5/3 \end{cases}$$

la soluzione ottenuta per il sistema diagonalizzato, vale anche per quello di partenza (provare per credere).

Per un sistema rettangolare, le cose si fanno più complicate. Prima di tutto dobbiamo individuare le cosiddette variabili dipendenti. Una variabile è dipendente se si può assegnarle qualsiasi valore senza modificare la struttura delle soluzioni.

Per esempio nell'equazione:

$3x + 2y = 5$

sia la x che la y sono dipendenti. Si può cioè assegnare un qualsiasi valore alla x e ricavare la y o viceversa. Nel sistema:

$$\begin{aligned} x + 2y + 3z &= 6 \\ 2z &= 3 \end{aligned}$$

la z non può essere dipendente (si dice indipendente). Le variabili dipendenti sono r = n − m (guarda che combinazione).

Il concetto di dipendenza, può associarsi anche alle righe (cioè alle equazioni) del sistema. Le due equazioni:

$$\begin{aligned} x + 3y + 5z &= 10 \\ 2x + 6y + 10z &= 20 \end{aligned}$$

sono chiaramente dipendenti perché una è esprimibile come combinazione lineare dell'altra (riga1=2*riga2).

Si dice rango del sistema, il numero delle sue righe indipendenti. Un altro teorema ci dice che il numero di equazioni indipendenti (il «rango per righe») è sempre uguale al numero di variabili indipendenti (il «rango per colonne»). Si osserva che applicando il metodo di Gauss-Jordan ad una matrice, le righe indipendenti verranno annullate (pensateci un po').

Una volta determinate le variabili dipendenti, si procede come segue.

```
(*************************************************************************)
(*                                                                       *)
(*                            MATH-TOOL S                                *)
(*                                                                       *)
(*                    Risoluzione sistemi lineari N x M                  *)
(*                        in calcolo simbolico.                          *)
(*                                                                       *)
(*                     Versione 3.3, 19 Dicembre 1987.                   *)
(*                     Scritto in TURBO PASCAL 3.0 per                   *)
(*                           IBM PC e compatibili                        *)
(*                             sotto Dos 3.10                            *)
(*                                                                       *)
(*                  Copyright(c) 1987 by Luca Padovano,                  *)
(*                    Via Casoni 1/a 40026 IMOLA (BO)                    *)
(*                          Tel:0542-24174.                              *)
(*                                                                       *)
(*************************************************************************)

PROGRAM sistemi (input,output);

CONST maxdim=7;                          (numero massimo di equazioni (modificabile))

TYPE fratto = RECORD
                num:REAL;
                den:REAL
              END;
     matrice = ARRAY [1..maxdim,1..maxdim] OF REAL;
     soluzioni = ARRAY [1..maxdim] OF FRATTO;
     stringa=STRING[30];

VAR mat,mataux:MATRICE;
    equa,dimmat,vars,n,j,l,freevars:INTEGER;
    x:ARRAY [1..maxdim] of STRING[3];
    sol:SOLUZIONI;
    ch:CHAR;
    solpar:BOOLEAN;
    vlb:ARRAY [1..maxdim] OF INTEGER;            (contiene le variabili libere)
    vin:ARRAY [1..maxdim] OF BOOLEAN;             (1=indipendente;0=dipendente)

{-------------------------------------------------------------------------}

PROCEDURE msg(dato:STRINGA);

BEGIN
  TEXTCOLOR(black);TEXTBACKGROUND(white);
  GOTOXY(24,8);WRITE('╔══════════════════════════════╗ ');
  GOTOXY(24,9);WRITE('║      ',dato,'  S      ? ');
  GOTOXY(24,10);WRITE('╚══════════════════════════════╝ ');
  GOTOXY(47,9);ch:='S';
  READ(kbd,ch);WRITE(ch);
  TEXTCOLOR(white);TEXTBACKGROUND(black);
END;

{-------------------------------------------------------------------------}

PROCEDURE inc (VAR dato:INTEGER);BEGIN dato:=SUCC(dato) END;

{-------------------------------------------------------------------------}

PROCEDURE dec (VAR dato:INTEGER);BEGIN dato:=PRED(dato) END;

{-------------------------------------------------------------------------}

PROCEDURE clr_wrk;

BEGIN
  FOR n:=equa+1 TO dimmat DO
    FOR j:=1 TO dimmat DO
      BEGIN
        vin[j]:=FALSE;
        mat[n,j]:=0
      END
END;

{-------------------------------------------------------------------------}

FUNCTION sum (riga,colonna:INTEGER):BOOLEAN;

VAR s:INTEGER;

BEGIN
  s:=0;
  FOR j:=1 TO colonna DO
    IF mat[riga,j]<>0 THEN INC(s);
  sum:=s=0
END;

{-------------------------------------------------------------------------}

FUNCTION rango (mat:MATRICE;eq,vr:INTEGER):INTEGER;

VAR r:INTEGER;

BEGIN
  r:=eq;
  FOR n:=eq DOWNTO 1 DO
    IF sum(n,vr) THEN DEC(r);                    (ci sono righe dipendenti)
  rango:=r
END;

{-------------------------------------------------------------------------}

PROCEDURE get_freevars;

CONST dip=TRUE;
      indip=FALSE;

VAR index,s,cont:INTEGER;
    lettera:CHAR;

{-------------------------------------------------------------------------}

PROCEDURE mem_fv (index:INTEGER);BEGIN INC(cont);vin[index]:=dip END;

{-------------------------------------------------------------------------}

BEGIN
  cont:=0;
  IF RANGO(mat,equa,equa)=equa THEN                        (tutte indipendenti)
    FOR n:=1 TO equa DO mem_fv(n)
      ELSE
        BEGIN
          FOR n:=1 TO equa DO
            IF mat [n,n]<>0 THEN mem_fv(n)     (se il termine sulla diagonale è)
              ELSE                             ({<>0,la variabile è indipendente)
                BEGIN
                  s:=0;
                  FOR j:=n+1 TO vars DO
                    IF mat[n,j]<>0 THEN
                      BEGIN
                        index:=j;
                        INC(s)
                      END;
                  IF s=1 THEN MEM_FV(index)  (se c' è una sola variabile sulla)
                END;                         (riga, allora è indipendente.     )
          index:=equa+1;
          WHILE cont<>equa DO                (se mancano delle variabili dipendenti)
            BEGIN                            (le prende a caso.                    )
              s:=0;
              FOR n:=1 TO equa DO
                IF mat[n,index]<>0 THEN INC(s);
                IF s<>0 THEN mem_fv(index);
                INC(index)
            END
```

```
        END;
    cont:=0;
    IF freevars>1 THEN lettera:='i' ELSE lettera:='e';
    WRITE ('Variabil',lettera,' dipendent',lettera,': ');
    FOR n:=1 TO vars DO
      IF vin[n]=indip THEN
        BEGIN
          INC(cont);vlb[cont]:=n;
          WRITE(x[n]);
          IF cont<freevars THEN WRITE(', ') ELSE WRITE('.')
        END;
    WRITELN;WRITELN
END;
{-------------------------------------------------------------------------------}
PROCEDURE set_freevars (flag:INTEGER);

VAR val,index:INTEGER;

BEGIN
  FOR n:=1 TO freevars DO
    BEGIN
      IF flag=n THEN val:=1 ELSE val:=0;
      IF flag=0 THEN index:=n+equa ELSE index:=vlb[n];
      mat[index,dimmat]:=val;
      mat[index,vlb[n]]:=1
    END
END;
{-------------------------------------------------------------------------------}
PROCEDURE clr_mat (VAR eq,vr:INTEGER);
{-------------------------------------------------------------------------------}
PROCEDURE shift_row (index:INTEGER);

BEGIN
  FOR i:=index TO eq-1 DO
    FOR j:=1 TO vr DO
      mat[i,j]:=mat[i+1,j];
  DEC(eq)
END;
{-------------------------------------------------------------------------------}

BEGIN
  n:=1;
  REPEAT
    IF sum(n,vr) THEN SHIFT_ROW(n)                    {cancella righe nulle}
      ELSE INC(n)
  UNTIL n=eq+1
END;
{-------------------------------------------------------------------------------}
FUNCTION check (value,dato:REAL):BOOLEAN;

BEGIN
  check:=value/dato-INT(value/dato)=0          {i due valori sono divisibili?}
END;
{-------------------------------------------------------------------------------}
PROCEDURE set_sgn (VAR value:fratto);

BEGIN
  WITH value DO
    BEGIN
      IF (num<0) AND (den<0) THEN BEGIN num:=-num;den:=-den END
        ELSE IF den<0 THEN BEGIN num:=-num;den:=-den END
    END
END;
{-------------------------------------------------------------------------------}
PROCEDURE print_sol;

BEGIN
  WRITE ('(');
  FOR n:=1 TO vars DO
    BEGIN
      WRITE(sol[n].num:0:0);
      IF sol[n].den<>1 THEN WRITE('/',sol[n].den:0:0);
      IF n=vars THEN WRITE(')') ELSE WRITE (', ')
    END
END;
{-------------------------------------------------------------------------------}
PROCEDURE semplifica (VAR value:fratto);

VAR dv:INTEGER;

BEGIN
  SET_SGN(value);
  WITH value DO
    BEGIN
      IF (num=0) or (den=0) THEN BEGIN num:=0;den:=1 END
        ELSE IF check(num,den) THEN BEGIN num:=num/den;den:=1 END
          ELSE IF check(den,num) THEN BEGIN den:=den/num;num:=1 END
            ELSE
              BEGIN
                dv:=2;                                  {continua a cercare il mcd}
                WHILE (dv<ABS(num)) and (dv<den) and (dv<24) DO
                  BEGIN
                    IF NOT(check(num,dv) and check(den,dv)) THEN dv:=dv+1
                      ELSE BEGIN num:=num/dv;den:=den/dv END
                  END
              END
    END;

  SET_SGN(value)
END;
{-------------------------------------------------------------------------------}
PROCEDURE add (VAR value,dato:FRATTO);

BEGIN
  IF dato.num<>0 THEN
    BEGIN
      value.num:=value.num*dato.den-value.den*dato.num;
      value.den:=value.den*dato.den;
      SEMPLIFICA(value)
    END
END;
{-------------------------------------------------------------------------------}
PROCEDURE print (mat:MATRICE;msg:STRINGA);

VAR car,cont,pos:INTEGER;
    f:BOOLEAN;

BEGIN
  CLRSCR;WRITELN(msg);
  FOR n:=1 TO LENGTH(msg) DO WRITE('-');WRITELN;WRITELN;
  f:=FALSE;
  WRITELN('┌');WRITELN('|');
  pos:=equa DIV 2+equa MOD 2;
  IF equa MOD 2=0 THEN f:=TRUE;
  FOR n:=1 TO equa DO
    BEGIN
```

(continua a pag. 222)

Sistema iniziale:

$$\begin{cases} 4x + 3y - 4z = 3 \\ 2x + 3y - 3z = 2 \\ x - 2y + z = 2 \end{cases}$$

Sistema triangolarizzato:

$$\begin{cases} 4x + 3y - 4z = 3 \\ 3y - 2z = 1 \\ 2z = 26 \end{cases}$$

Sistema diagonalizzato:

$$\begin{cases} 4x = 28 \\ 3y = 27 \\ 2z = 26 \end{cases}$$

soluzione del sistema. 7, 9, 13

Stampa di tre momenti di lavoro.

1. Si pongono a 0 le variabili dipendenti e si risolve il sistema (che adesso è quadrato!!). In questo modo otteniamo la soluzione particolare.
2. Si considera il sistema omogeneo e si pone a 1 una variabile e a 0 tutte le altre, si risolve e si ripete lo stesso procedimento per le r variabili dipendenti. In questo modo otteniamo la base dello spazio delle soluzioni.

I punti 1 e 2 sono conformi alle premesse teoriche fatte in precedenza. Naturalmente il fatto di assegnare alle variabili dipendenti proprio i valori 1 o 0 è puramente arbitrario. Si potrebbe cioè assegnare alle dipendenti un qualsiasi valore. La soluzione generale sarebbe la stessa.

Il programma

Math-Tool S è scritto in Turbo Pascal 3.0 sotto Dos 2.xx o 3.xx su un M24. L'uso del Pascal in questo caso, è stato fondamentale. Per operare su frazioni, si è semplicemente definito un nuovo tipo (appunto FRATTO) e alcune operazioni su questo nuovo ente. Il blocco principale del programma è dato da tre procedure.

La procedura GAUSS, si occupa di

triangolarizzare la matrice. Applicando una volta la procedura, invertendo la matrice facendo attenzione ad eliminare le righe nulle, riapplicando una seconda volta la procedura e reinvertendo nuovamente, otteniamo la matrice diagonalizzata; inoltre abbiamo tutte le informazioni per riconoscere le variabili dipendenti.

La procedura non presenta particolari novità rispetto ad una qualsiasi delle 847.657 versioni che circolano nelle Università della penisola. Unica distinzione è appunto l'utilizzo degli enti «fratti» al posto dei numeri reali.

Una osservazione: è possibile, aggiungendo a questa routine una manciata di istruzioni, ottenere anche il determinante della matrice. Non ditemi che non ci avete già pensato...

La procedura GET_FREEVARS si occupa di individuare le variabili indipendenti (vedi il listato pubblicato in queste pagine). Supponiamo di avere m equazioni e n incognite e quindi una matrice dei coefficienti di m righe e n colonne.

In primo luogo, con la procedura RANGO, si controlla se le prime m variabili sono indipendenti. In questo

Descrizione delle procedure

MSG: stampa dei messaggi utente.
INC, DEC: incrementa/decrementa un valore.
CLR_WRK: inizializzazione degli array.
SUM: controlla se una riga è nulla.
RANGO: controlla le righe indipendenti.
GET_FREEVARS: individua variabili dipendenti.
SET_FREEVARS: prepara la matrice aggiungendo le equazioni delle variabili dipendenti.
CLR_MAT: toglie dalla matrice le righe nulle.
CHECK: controlla se due valori sono divisibili.
SET_SGN: mette il segno al numeratore di un FRATTO.
PRINT_SOL: stampa le soluzioni.
SEMPLIFICA: riduce un FRATTO.
ADD: addizione due FRATTI.
PRINT: stampa la matrice.
INSERT: inserimento della matrice.
GAUSS: triangolarizza la matrice.
TRANSFER: inversione della matrice.
COMPUTE: calcola le soluzioni.
SET_LAYOUT: crea i caratteri delle variabili.
INIT: inizializzazione.
GAUSS_JORDAN: diagonalizza la matrice.
REMOVE: toglie le righe nulle in fondo alla matrice.
ONE_SOL: una sola soluzione
MORE_SOL: più soluzioni.

(segue da pag. 221)

```
        IF (n=pos) AND (NOT f) THEN WRITE('< ') ELSE WRITE('| ');
        cont:=0;
        FOR j:=1 TO vars DO
          IF mat[n,j]<>0 THEN
            BEGIN
              INC(cont);
              IF cont=1 THEN car:=32 ELSE car:=43;
              IF j>1 THEN
                BEGIN
                  IF (mat[n,j]<0) THEN WRITE(' - ')
                    ELSE WRITE(' ',CHR(car),' ')
                END
                  ELSE
                    IF mat[n,1]<0 THEN WRITE(' - ') ELSE WRITE('   ');
              IF ABS(mat[n,j])<>1 THEN WRITE(ABS(mat[n,j]):0:0);
              WRITE(x[j])
            END;
        WRITELN(' = ',mat[n,dimmat]:0:0);
        IF (n=pos) AND (f) THEN WRITELN('<  ') ELSE WRITELN('| ')
      END;
    WRITELN('L');WRITELN;
    TEXTCOLOR(black);TEXTBACKGROUND(white);
    GOTOXY(1,25);WRITE('  Premere un tasto per continuare.');CLREOL;
    REPEAT UNTIL KEYPRESSED;
    TEXTCOLOR(white);TEXTBACKGROUND(black);
    GOTOXY(1,25);CLREOL;GOTOXY(1,2*equa+9)
  END;

{-------------------------------------------------------------------------------}


  PROCEDURE insert;

  BEGIN
    solpar:=TRUE;CLRSCR;
    FOR n:=1 TO equa DO
      BEGIN
        WRITELN;WRITELN('Equazione numero ',n,':');WRITELN;
        FOR j:=1 TO vars DO
          BEGIN
            WRITE('Coefficiente variabile ',x[j],' = ');
            READLN(mat[n,j])
          END;
        WRITE('Termine noto = ');readln(mat[n,dimmat]);
        solpar:=(mat[n,dimmat]=0) AND (solpar)       {controlla se c'è sol. part.}
      END
  END;

{-------------------------------------------------------------------------------}

  PROCEDURE GAUSS (VAR m:MATRICE;equa,vars:INTEGER);

  VAR n,j,w,v:INTEGER;
      t,k,h,p,q:REAL;
      fr:FRATTO;
      found:BOOLEAN;

  BEGIN
    w:=1;v:=2;
    REPEAT
      IF m[v-1,w]=0 THEN          {se non c' è bisogno di annullare, passa oltre}
        BEGIN                     {e scambia due righe                           }
          found:=FALSE;n:=v;
          WHILE (NOT found) AND (n<=equa) DO
            BEGIN
              IF m[n,w]<>0 THEN
                BEGIN
                  FOR j:=w TO vars DO
                    BEGIN
                      t:=m[v-1,j];
                      m[v-1,j]:=m[n,j];
                      m[n,j]:=t
                    END;
                  found:=TRUE
                END;
              INC(n)
            END
        END;
      FOR n:=v TO equa DO
      IF m[n,w]<>0 THEN
        BEGIN
          k:=m[n,w];h:=m[v-1,w];
          fr.num:=ABS(k);fr.den:=ABS(h);
          SEMPLIFICA(fr);
          p:=fr.num;q:=fr.den;
          FOR j:=w TO vars DO
            IF h/k>0 THEN m[n,j]:=m[n,j]*q-p*m[v-1,j]
              ELSE m[n,j]:=m[n,j]*q+p*m[v-1,j]                  {elide i termini}
        END;
      INC(v);INC(w)
    UNTIL v>equa
  END;

{-------------------------------------------------------------------------------}

  PROCEDURE transfer (eq,vr:INTEGER);

  VAR mataux:MATRICE;

  BEGIN
    mataux:=mat;
    FOR n:=1 TO eq DO
      FOR j:=eq+1 TO vr DO mataux[n,j]:=mat[eq-n+1,j];

    FOR n:=1 TO eq DO
      FOR j:=1 TO eq DO mataux[n,j]:=mat[eq-n+1,eq-j+1];
    mat:=mataux
  END;

{-------------------------------------------------------------------------------}

  PROCEDURE compute;

  VAR value,dato:FRATTO;
      eq:INTEGER;

  BEGIN
    eq:=vars;
    sol[eq].num:=mat[eq,dimmat];
    sol[eq].den:=mat[eq,eq];
    SEMPLIFICA(sol[eq]);
    FOR n:=eq-1 DOWNTO 1 DO
      BEGIN
        value.num:=mat[n,dimmat];
        value.den:=1;
        FOR j:=1 TO eq-n DO
          BEGIN
            dato:=sol[dimmat-j];
            dato.num:=dato.num*mat[n,dimmat-j];
            SEMPLIFICA(dato);
            ADD(value,dato)
          END;
        sol[n]:=value;
        sol[n].den:=sol[n].den*mat[n,n];
        SEMPLIFICA(sol[n])
      END
  END;

{-------------------------------------------------------------------------------}
```

```
PROCEDURE set_layout;

VAR buf:STRING[3];

BEGIN
  IF vars<6 THEN
    BEGIN
      CASE vars OF
        2:BEGIN x[1]:='x';x[2]:='y' END;
        3:BEGIN x[1]:='x';x[2]:='y';x[3]:='z' END;
        4:BEGIN x[1]:='x';x[2]:='y';x[3]:='z';x[4]:='t' END;
        5:BEGIN x[1]:='x';x[2]:='y';x[3]:='z';x[4]:='t';x[5]:='u' END
      END
    END
      ELSE
        BEGIN
          FOR n:=1 to vars DO
            BEGIN
              STR(n,buf);
              x[n]:=CONCAT('x',buf)
            END
        END
END;

{-------------------------------------------------------------------------------}

PROCEDURE init;


BEGIN
  REPEAT
    REPEAT
      CLRSCR;
      WRITE('Numero equazioni : ');READLN(equa)
    UNTIL equa<maxdim;
    REPEAT
      WRITE('Numero incognite : ');readln(vars)
    UNTIL vars<maxdim;
    SET_LAYOUT;
    dimmat:=vars+1;
    INSERT;
    MSG('Tutto giusto?')
  UNTIL ch IN ['s','S',#13];
  CLRSCR;PRINT(mat,'Sistema iniziale:')
END;

{-------------------------------------------------------------------------------}

PROCEDURE gauss_jordan (VAR mat:MATRICE;VAR eq,vr:INTEGER);

VAR zero:BOOLEAN;
    fr:FRATTO;

{-------------------------------------------------------------------------------}

PROCEDURE remove;

BEGIN
  REPEAT
    zero:=sum(eq,vr);
    IF zero THEN DEC(eq)                    {cancella righe in fondo alla matrice}
  UNTIL NOT zero
END;

{-------------------------------------------------------------------------------}

BEGIN
  GAUSS(mat,eq,vr);                                              {triangolarizza}
  REMOVE;                                                  {toglie righe nulle}
  mataux:=mat;
  IF eq>=vr THEN CLR_MAT(eq,vr)
    ELSE
      BEGIN
        TRANSFER(eq,vr);                                                {inverte}
        GAUSS(mat,eq,vr);                                       {triangolarizza}
        REMOVE;                                                 {toglie righe nulle}
        TRANSFER(eq,vr);                                                {reinverte}
        CLR_MAT(eq,vr);                                         {pulisce...}
        GAUSS(mat,eq,vr)                                        {... e ricalcola}
      END
END;

{-------------------------------------------------------------------------------}

PROCEDURE one_sol;

BEGIN
  WRITE('Soluzione del sistema: ');
  COMPUTE;
  PRINT_SOL
END;

{-------------------------------------------------------------------------------}

PROCEDURE more_sol;

BEGIN
  CLR_WRK;
  freevars:=vars-equa;
  GET_FREEVARS;
  SET_FREEVARS(0);
  GAUSS(mat,vars,dimmat);                          {calcola soluzione particolare}
  COMPUTE;
  WRITELN('Soluzione generale del sistema:');WRITELN;
  IF NOT solpar THEN
    BEGIN
      PRINT_SOL;
      WRITE(' + ')
    END;
  FOR n:=1 TO dimmat DO mat[n,dimmat]:=0;                {sistema omogeneo (b=0)}
  FOR i:=1 TO freevars DO
    BEGIN
      SET_FREEVARS(i);                                  {da i valori alle var. dip.}
      GAUSS(mat,vars,dimmat);
      COMPUTE;
      WRITE(CHR(i+103));
      PRINT_SOL;
      IF i<>freevars THEN WRITE(' + ')
    END
END;

{-------------------------------------------------------------------------------}
{-------------------------->PROGRAMMA  PRINCIPALE<------------------------------}
{-------------------------------------------------------------------------------}

BEGIN
  REPEAT
    INIT;
    GAUSS_JORDAN(mat,equa,dimmat);
    IF RANGO(mat,equa,vars)<>RANGO(mat,equa,dimmat) THEN
      WRITELN(#13,'Il sistema non ha soluzione.')                {Teorema di .....}
        ELSE
          BEGIN
            PRINT(mataux,'Sistema triangolarizzato:');
            PRINT(mat,'Sistema diagonalizzato:');
            IF equa=vars THEN one_sol ELSE more_sol
          END;

    MSG(' Ci riprovi ?')
  UNTIL ch in ['n','N'];
  CLRSCR;WRITELN('Bye, bye....')
END.

{-------------------------------------------------------------------------------}
```

caso le rimanenti variabili sono dipendenti.

In caso contrario, si controlla se i termini sulla diagonale di questa matrice m×m sono nulli. Grazie al modo di operare del metodo di Gauss-Jordan, possiamo dire che la variabile corrispondente alla i-esima colonna è sicuramente indipendente se il termine a_{ii} è non nullo. Se invece esso è nullo, si deve ancora controllare se sulla i-esima riga ci sono altri coefficienti non nulli. Se ci sono la i-esima variabile è dipendente.

Controllate le prime m variabili, nel caso in cui manchino ancora delle variabili indipendenti, le si prende a caso fra le rimanenti variabili. Il vettore vin[] mi dice se la i-esima variabile è indipendente, il vettore vlb[] conserva le n - m variabili dipendenti. Fatto questo, basterà aggiungere in coda alle equazioni date le equazioni ottenute ponendo alternativamente a zero e a uno le variabili dipendenti. In questo modo il sistema diventa quadrato e può essere facilmente calcolato.

Facciamo un esempio; Math-Tool S risolve l'equazione x + 2y + 3z = 2 risolvendo i tre sistemi quadrati:

$$\begin{aligned} x + 2y + 3z &= 2 \\ y &= 0 \\ z &= 0 \end{aligned}$$

$$\begin{aligned} x + 2y + 3z &= 0 \\ y &= 1 \\ z &= 0 \end{aligned}$$

$$\begin{aligned} x + 2y + 3z &= 0 \\ y &= 0 \\ z &= 1 \end{aligned}$$

Il calcolo vero e proprio viene affidato alla procedura COMPUTE. Anche in questo caso, valgono le considerazioni fatte per la procedura GAUSS. MC

Principali variabili del programma

maxdim: numero massimo di equazioni (modificabile).
fratto: lo zoccolo duro del programma.
mat[]: matrice dei coefficienti.
sol[]: soluzioni del sistema.
equa: numero di equazioni.
vars: numero dei variabili.
dimmat: reale dimensione della matrice (vars+1).
freevars: numero di variabili dipendenti.
x[]: stringhe contenenti il formato delle variabili.
vlb[],vln[]: conservano le variabili indipendenti e dipendenti.