Description of a program for nonlinear programming

J. Mottl

Geofyzika, Brno, Ječná 29a, Czechoslovakia

The paper describes a program for solving the problem of nonlinear programming and includes the program itself in FORTRAN. Two illustrating examples are given.

Die Arbeit beschreibt Programm für ein nichtlineares Programmierungsproblem und enthält auch das eigene Programm in FORTRAN. Zur Illustration sind zwei Beispiele beigefügt.

L'article décrit le programme pour la solution du probleme de la programmation non-linéaire et contient le programme seul en FORTRAN. Pour l' illustration deux exemples sont présentés.

(Received July 1977)

1. Introduction

Let us consider the problem of nonlinear programming in the form

$$g_e(\mathbf{x}) \leqslant 0 \quad e = 1, 2 \ldots . m \tag{1}$$

$$f(\mathbf{x}) \to \min \tag{2}$$

where $\mathbf{x} = (x_1, x_2 \ldots x_t \ldots x_n)$ and g_e, f are nonlinear functions.

2. Definition of the function determining the feasible domain

The F function will be introduced so that its value at the point $\mathbf{x}_i$ is

$$F(\mathbf{x}_i) = \sum_{e=1}^{m} g_e^+(\mathbf{x}_i) \tag{3}$$

(See **Fig. 1**) where g_e^+ is defined in the following way:

$$g_e(\mathbf{x}_i) > 0 \to g_e^+ = g_e$$

$$g_e(\mathbf{x}_i) \leqslant 0 \to g_e^+ = 0 \tag{4}$$

According to (3) and (4) the F function exhibits the property that in the feasible domain ($g_e(\mathbf{x}) \leqslant 0$, $e = 1, 2, \ldots . m$) its values are smaller (zero) than in any other place of the non-feasible domain.

For the above properties, the F surface should function as a governing surface for searching the feasible domain.

3. The solution of the nonlinear programming by the method of gradual achievements of the feasible region

Let us designate j as the step of achieving the feasible region.

1. $j = 1$ Let us choose arbitrarily the starting point $\mathbf{x}_1$. The lowermost point $\mathbf{x}_2$ ($\mathbf{x}_1 \to \mathbf{x}_2$ in **Fig. 2**) will then be found by the searching program S on the surface

$$F = \sum_{e=1}^{m} g_e^+ \tag{5}$$

   This represents the feasible solution from the point of view of limitations (1) only.

2. In the second step the following member will be added to the F

$$g_{fj=2} = f(\mathbf{x}) + [-f(\mathbf{x}_2) + \bar{\Delta}] \tag{6}$$

   it means that

$$F = \sum_{e=1}^{m,f_j} g_e^+ \tag{7}$$

   where $\bar{\Delta}$ is the chosen constant (see Section 7). By the searching program we will find point $\mathbf{x}_3$ ($\mathbf{x}_2 \to \mathbf{x}_3$) on the surface (7).

3. During the following steps j, then, g_j always changes after achieving the feasible region ($\mathbf{x}_j \to \mathbf{x}_{j+1}$) according to

$$g_{fj} \to g_{fj+1} = g_{fj} + \bar{\Delta} \tag{8}$$

   The realisation of this algorithm is clearly visible in Fig. 2 from which it follows that it can lead to finding the feasible solution with the smallest quantity f.

The necessity for the exceptional steps $j = 1, 2$ (i.e. first achieving arbitrary feasible solution with no regard to $f(\mathbf{x})$, and then the addition of the supplementary limitation g_{fj} formed from $f(\mathbf{x})$) follows from Fig. 2.

At the choice of the special starting point $\mathbf{x}_1 = \mathbf{x}_1'$ according to Fig. 2, though, the immediate application of g_{fj} ($g'_{fj=1}$ in Fig. 2) would mean that the point $\mathbf{x}_1'$ could not achieve the following feasible solution ($\mathbf{x}_2'$), because, according to Fig. 2, the common intersection of the feasible regions does not exist from the point of view of g_e, $e = 1, 2 \ldots m$ and $g_{fj=1}$. During the realisation of the formerly introduced algorithm, though, such a situation could not happen.

4. The description of the searching algorithm S

It follows from the above description, that, during the individual steps j, the lowest point on the F surface according to (7) (where the feasible region will exist) would always be looked

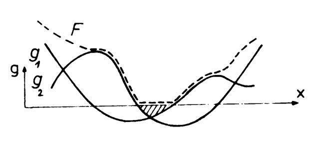

Fig. 1

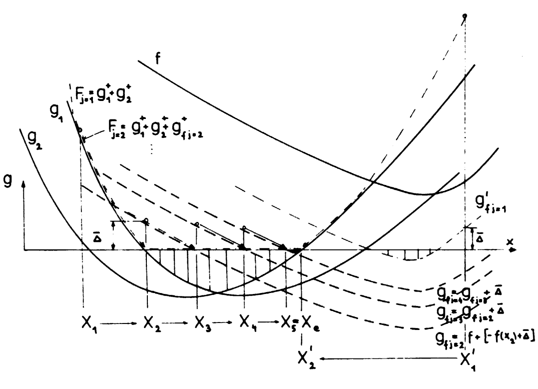

Fig. 2

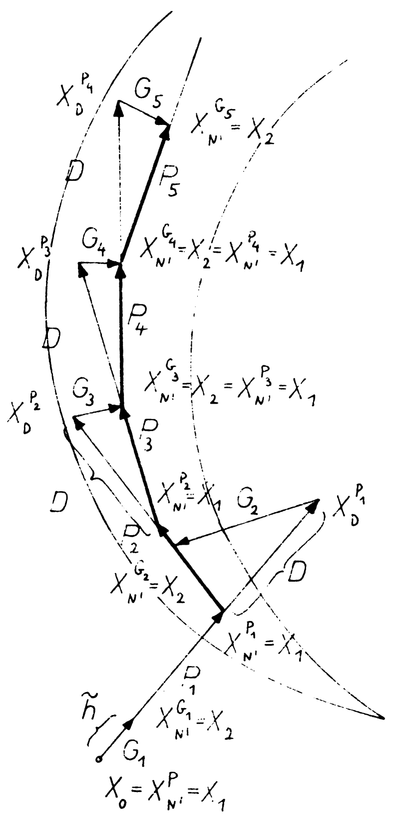

Fig. 3

for. The searching itself will thus be implemented in the direction of the F surface decrease.

The method chosen is explained in **Fig. 3**, which represents surface F in the form of curved valley (two-dimensional problem)—we want to move in the direction of its decline.

At the starting point $\mathbf{x}_0$ the gradient to the F surface is found numerically. In the negative direction of this gradient (thus in the direction of the decline of F ordinates) we move forward along straight line G_1 by distance $\tilde{h}$ into point $\mathbf{x}_{N'}^{G_1}$, which represents the lowest F ordinate on straight line G_1.

Point $\mathbf{x}_0$ will be designated as $\mathbf{x}_1$ point $\mathbf{x}_{N'}^{G_1}$ as $\mathbf{x}_2$ (this designation follows from the definition $F(\mathbf{x}_1) > F(\mathbf{x}_2)$) and through these points $\mathbf{x}_1$, $\mathbf{x}_2$ straight line P_1 is fitted. Along this line we move ahead in the direction $\mathbf{x}_1 \to \mathbf{x}_2$ (thus in the direction of decline of F) into the point with the lowest F ordinate designated as $\mathbf{x}_{N'}^{P_1}$. From this point, then, we still move ahead in the direction $\mathbf{x}_1 \to \mathbf{x}_2$ along straight line P_1 by distance D into point $\mathbf{x}_D^{P_1}$.

In point $\mathbf{x}_D^{P_1}$ the gradient to F surface is again found and in the negative direction of this gradient we move ahead along straight line G_2 into its lowermost point $\mathbf{x}_{N'}^{G_2}$.

Points $\mathbf{x}_{N'}^{P_1}$ (on the previous straight line P_1) and $\mathbf{x}_{N'}^{G_2}$, (on straight line G_2) are again designated as $\mathbf{x}_1$, $\mathbf{x}_2$ (according to the rule $F(\mathbf{x}_1) > F(\mathbf{x}_2)$). Through these points $\mathbf{x}_1$, $\mathbf{x}_2$ straight line P_2 is fitted and we move ahead in the direction $\mathbf{x}_1 \to \mathbf{x}_2$ (the direction of decline of F surface) to point $\mathbf{x}_{N'}^{P_2}$ (the point with the lowest ordinates F). In addition to it we move on from $\mathbf{x}_{N'}^{P_2}$ by distance D along straight line P_2, up to point $\mathbf{x}_D^{P_2}$.

In point $\mathbf{x}_D^{P_2}$, the gradient to the F surface is found and we move in its negative direction along straight line G_3 into their lowest point $\mathbf{x}_{N'}^{G_3}$. Points $\mathbf{x}_{N'}^{P_2}$, $\mathbf{x}_{N'}^{G_3}$ will be designated as $\mathbf{x}_1$, $\mathbf{x}_2$ $(F(\mathbf{x}_1) > F(\mathbf{x}_2))$ and through them a straight line P_3 will be fitted etc.

5. The termination of the process

According to the method described in Section 4 we move ahead along the valley for so long until we find the feasible domain (that is the step j in the sense of Section 3). After $g_{fj} \to g_{fj+1}$ acc. (8) the whole procedure is repeated. The last from the sequence of these feasible domain is then the searched extreme. At a certain step j, it can happen, though (for generally undulated F surface) that there can be more depressions. From these depressions, then, only some are with the feasible domain ($F = 0$), while others exhibit ($F > 0$) in the lowest place, where the point may be stuck (see example 2 in Section 8).

In order to solve this eventuality at least partly the following operation is introduced. If we are not able to find a point exhibiting $F = 0$ in step j within a chosen number of steps sending straight lines G respectively $P(z = z_M)$ (during the search of the $\mathscr{H}$ valley), we will choose a new starting searching point $\mathbf{x}_0 = \mathbf{x}_G$ outside of this valley.

Let us assume that through a scale reduction the assumption of the existence of a feasible point can be achieved in the area of the magnitude H^n about point $\mathbf{x}^L$ (where n is the number of variables, H is the dimension chosen). Let us then define $\mathbf{x}_0 = \mathbf{x}_G$ as the centre of gravity of this area of searching, weakened by 'cavities' in the vicinity of points $\mathbf{x}_{\mathscr{H}}$ exhibiting magnitudes $H_0^n(H_0 < H)$. Points $\mathbf{x}_{\mathscr{H}}$ are these points, where the lowest places of 'unsuccessful depressions' were found during the course of the process. If the designation $\omega = \frac{H_0^n}{H^n}$ in introduced (ω will be chosen, for instance, within the interval $\langle 0, 1 - 0, 25 \rangle$ then

$$x_{G,t} = \frac{x_t^L - \omega \sum_{\mathscr{H}=1}^{\mathscr{H}} x_{\mathscr{H},t}}{1 - \mathscr{H}\omega}, \quad t = 1, 2 \ldots . n \qquad (9)$$

If the number of points $\mathbf{x}_{\mathscr{H}}$, in the step j considered reaches the limiting value $\mathscr{H} = \mathscr{H}_M$, then the process terminates. It means that we have searched through the step j the neighbourhood of $\mathscr{H}_M$ depressions on the surface, without reaching the feasible domain. Further searching is already less hopeful. Thus the extreme is the point in the last found feasible domain, i.e. in the preceding step j.

6. To determine the direction of the straight line

The gradient to the F surface for determining the direction of straight line G will be derived numerically from F values in the points at the distance h from the central point in the direction of individual co-ordinate axes.

For the searching the lowermost point on the straight line G (or P) the simplest algorithm was chosen, where it proceeds along the straight line in steps of the length $\tilde{h}$ (K_M steps), that leads to the point $\mathbf{x}_{GN}$. The neighbourhood of this point will be searched by means of the same algorithm (but now being of a smaller step $\tilde{h}' < \tilde{h}$) and in the k'_M number of steps to both sides from $\mathbf{x}_{GN}$ on the straight line G. This way, the final point $\mathbf{x}_{GN'}$ will be found. This simple algorithm is chosen because $\mathbf{x}_{N'}^G$ and $\mathbf{x}_{N'}^P$ do not need to be defined precisely. Straight line P will be found by fitting points $\mathbf{x}_1$, $\mathbf{x}_2$ (according to Section 4). Finding the minimum point on it ($\mathbf{x}_{N'}^P$) is then analogous to the case of straight line G.

When the situation $\mathbf{x}_1 \equiv \mathbf{x}_2$ occurs the program chooses $x_{1,t} - x_{2,t} = 1$ for $t = 1, 2 \ldots \ldots . n$; thus we can leave the point where $\mathbf{x}$ was stuck and the process can continue.

7. The complete algorithm

Precision of the solution will be influenced by the choice of constant $\bar{\Delta}$ (**Fig. 4**). If the process is to be terminated after r steps the following should be chosen

$$\bar{\Delta} = \frac{1}{r}[f(\mathbf{x}_0) - f(\mathbf{x}_e)]$$

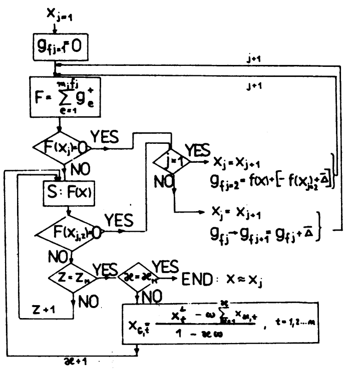

Fig. 4

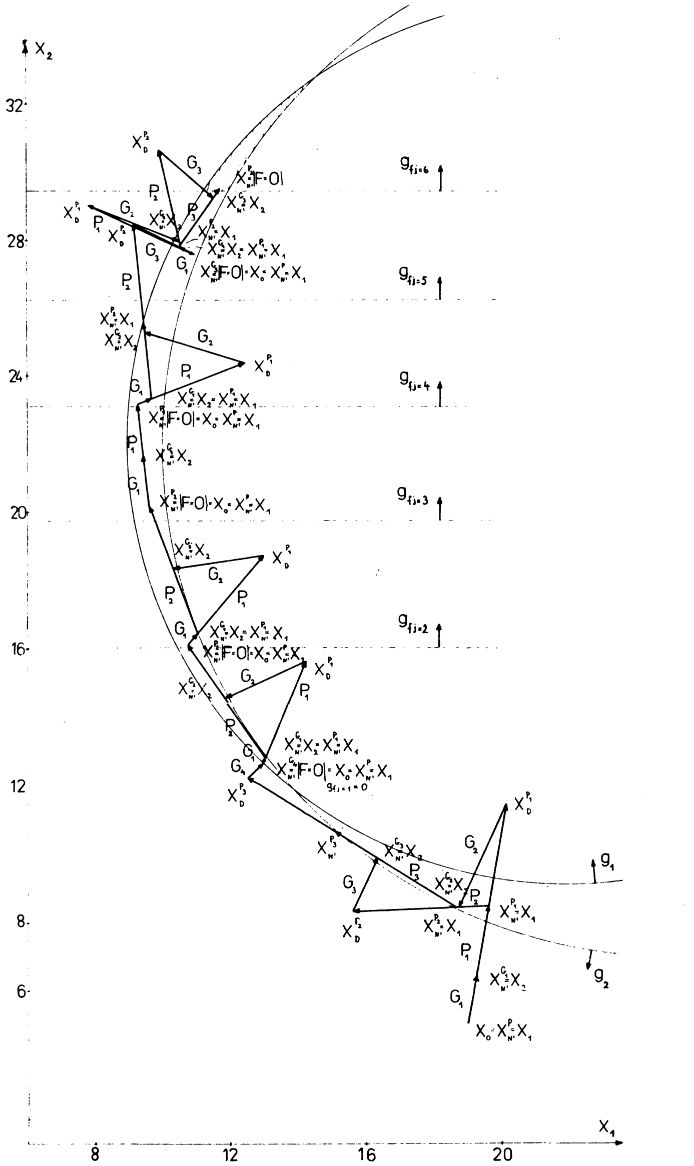

Fig. 5

where $\mathbf{x}_e$ designates an estimate of the solution to be found. If, further, the found solution is to be made more precise, it is enough to define the found $\mathbf{x}_e$ as a starting point of the new solution, where chosen $\overline{\Delta}$ is smaller (and thus the searching interval for $\tilde{h}$ and $\tilde{h}'$ is adequately smaller).

8. Illustrative examples

Example 1

$$g_1 = (x_1 - 22)^2 + (x_2 - 22)^2 - 168 \leqslant 0$$
$$g_2 = -(x_1 - 25)^2 - (x_2 - 22)^2 + 224 \leqslant 0$$

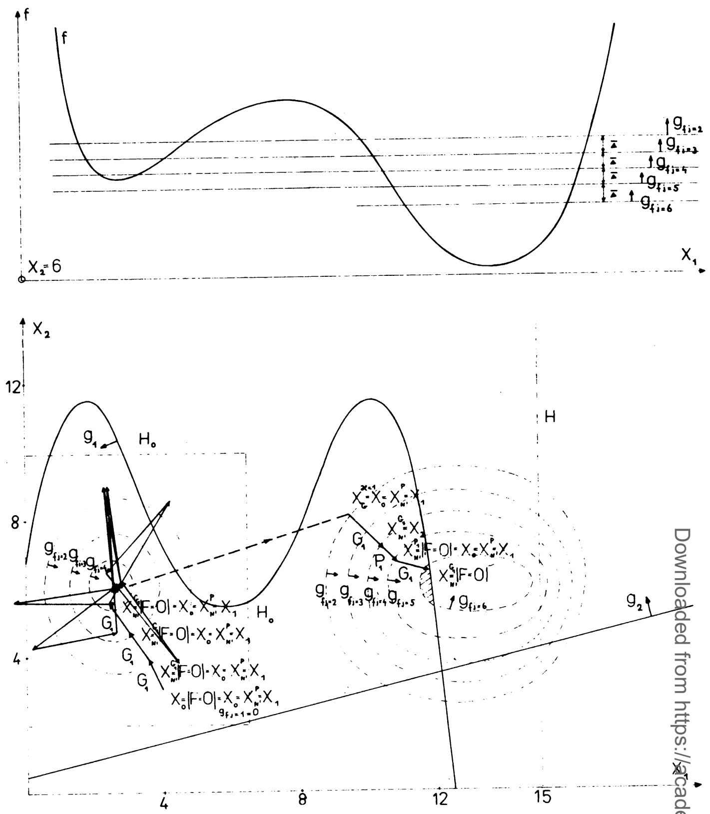

Fig. 6

Table 1

```
   N       M      KM      KMC     ZM    KAPM
   2       2       3       3      12      2

      H          HVL         HVLC          D          DELTP        OMEGA
  .5000E+00   .1000E+01   .1500E+00   .3000E+01   .1000E+02   .2000E+00

  XL

  .1000E+02   .1000E+02

 F0= .130000E+03           INITIAL VECTOR X0
   1  -  2   19.0000   5.0000

 FG= .820407E+02
   1  -  2   19.2520   6.4279

 FP= .327015E+02
   1  -  2   19.6109   6.4960

 FD= .890083E+02
   1  -  2   20.1363  11.4503

 FG= .275428E+02
   1  -  2   18.7571   8.4533

 FP= .271377E+02
   1  -  2   18.6182   6.4464

 FD= .604385E+02
   1  -  2   15.6219   8.2975

 FG= .130355E+02
   1  -  2   16.3393   9.8367

 FP= .966103E+01
   1  -  2   15.0768  10.6101

 FD= .184773E+02
   1  -  2   12.5169  12.1742

 F0= .100003E+02           FEASIBLE DOMAIN
   1  -  2   13.0030  12.6778

 FG= .958375E+01
   1  -  2   13.0599  12.8166

 FP= .958375E+01
   1  -  2   13.0599  12.8166

 FD= .675429E+02
   1  -  2   14.1996  13.5917

 FG= .448761E+01
   1  -  2   11.6879  14.5153

 F0= .970562E+01           FEASIBLE DOMAIN
   1  -  2   10.7833  16.1090

 FG= .868346E+01
   1  -  2   11.0819  16.4500
```

Table 1 continued

```
FP= .868345E+01
 1  -  2   11.0619    16.4500

FD= .720632E+02
 1  -  2   13.0390    16.7237

FG= .305228E+01
 1  -  2   10.3683    18.3271

FD= .740665E+01              FEASIBLE DOMAIN
 1  -  2    9.6608    20.1890

FG= .315057E+01
 1  -  2    9.4798    21.6276

FD= .853683E+01              FEASIBLE DOMAIN
 1  -  2    9.2806    23.1056

FG= .806035E+01
 1  -  2    9.7076    23.3244

FP= .806035E+01
 1  -  2    9.7076    23.3244

FD= .673178E+02
 1  -  2   12.5146    24.3032

FG= .207732E+01
 1  -  2    9.5007    25.3187

FP= .208094E+01
 1  -  2    9.4921    25.4636

FD= .376442E+02
 1  -  2    9.1913    26.4464

FD= .538022E+01              FEASIBLE DOMAIN
 1  -  2   10.9777    27.5491

FG= .461051E+01
 1  -  2   10.5707    27.7410

FP= .461051E+01
 1  -  2   10.5707    27.7410

FD= .822775E+02
 1  -  2    7.6572    29.0204

FG= .420893E+01
 1  -  2   10.4951    27.9415

FP= .438945E+01
 1  -  2   10.5178    27.8614

FD= .647602E+02
 1  -  2    9.4532    30.6864

FG= .324963E+00
 1  -  2   11.5533    29.2362

FD= .962573E+01              FEASIBLE DOMAIN
 1  -  2   11.7337    29.4692
```

$$f = -3x_2 \rightarrow \min \rightarrow g_{fj} = -3x_2 + \text{DELTJ}$$
$$\mathbf{x}_0 = (19\,;\,5)$$

Example 2

$$g_1 = 2x_2 + 0{,}03708x_1^4 - 0{,}8898x_1^3 + 6{,}674x_1^2 - 16{,}0202x_1 - 11{,}002 \leqslant 0$$
$$g_2 = -2x_2 + 0{,}5x_1 + 1 \leqslant 0$$
$$g_{fj} = 0{,}02377561716x_1^4 - 0{,}734996505x_1^3 + 7{,}12039086x_1^2 - 23{,}78989107x_1 + 1{,}35x_2^2 - 16{,}2x_2 + \text{DELTJ}$$
$$\mathbf{x}_0 = (4\,;\,3)$$

9. The FORTRAN program

The description of input data of the program OPT.

The input data for each problem form two control cards, cards with values of the starting vector of variables and the subroutine GFUN for a calculation of functional values set up by the program user.

(*a*) The first control card (6I5)

columns	*variables*	*input*
1– 5	$N(n)$	the total number of variables ($N \leqslant 100$)
6–10	$M(m)$	the total number of limitations ($M \leqslant 99$)
11–15	$KM(k_M)$	the number of rough steps on the straight line P and G
16–20	$KMC(k_M')$	the number of fine steps on the straight line P and G making solutions more accurate
21–25	$ZM(z_M)$	the limiting number of generated straight lines P and G in one step KAPA
26–30	$KAPM(\mathscr{H}_M)$	the limiting number of steps KAPA

Table 2

```
     N       M      KM      KMC      ZM    KAPM
     2       2       8       3       10      2

       H           HVL          HVLC           D          DELTP        OMEGA
 .5000E+00    .1000E+01    .1500E+00    .3000E+01    .4500E+01    .2844E+00

 XL

 .7500E+01    .7500E+01

FD= .000000E+00           INITIAL VECTOR XO
 1  -  2    4.0000        3.0000

FD= .450000E+01           FEASIBLE DOMAIN
 1  -  2    4.0000        3.0000

FD= .121117E+01           FEASIBLE DOMAIN
 1  -  2    3.5670        3.9014

FD= .530099E+00           FEASIBLE DOMAIN
 1  -  2    2.9938        4.7208

FD= .274882E+01           FEASIBLE DOMAIN
 1  -  2    2.4832        5.5836

FG= .251807E+01
 1  -  2    2.7178        5.9706

FP= .251807E+01
 1  -  2    2.7178        5.9706

FD= .180013E+02
 1  -  2    4.2485        6.5507

FG= .286335E+01
 1  -  2    2.3498        6.4252

FP= .249512E+01
 1  -  2    2.6329        6.6754

FD= .149043E+02
 1  -  2    4.5203        3.7435

          .
          .
          .
          .
          .

FG= .250422E+01
 1  -  2    2.6822        5.9463

FP= .250009E+01
 1  -  2    2.5323        5.9426

FD= .504191E+02
 1  -  2    -.4605        5.8262

FG= .258439E+01
 1  -  2    2.5333        5.7432

FD= .144395E+02           NEW VECTOR XO=XG
 1  -  2    9.4391        8.0966

FG= .317003E+01
 1  -  2   10.4708        7.0658

FD= .365421E+01           FEASIBLE DOMAIN
 1  -  2   10.8704        6.7017

FD= .111842E+01           FEASIBLE DOMAIN
 1  -  2   11.8384        6.4507
```

(*b*) The second control card (8E10.3)

columns	*variables*	*input*
1–10	$H(h)$	the length of step during the numerical calculation of the gradient of the F surface
11–20	$HVL(\tilde{h})$	the length of step on straight line G (P)
21–30	$HVLC(\tilde{h}')$	the length of a refined step on straight line G, P
31–40	D	the distance between $\mathbf{x}_{N'}^P$, and $\mathbf{x}_D^P$ on straight line P
41–50	$DELTP(\Delta)$	the decline of the value of the section
51–60	$OMEGA(\omega)$	the ratio of the 'cavity' about point $\mathbf{x}_{\mathscr{H}}$ (i.e. the centre of the

unsuccessful depression) and the total volume of the searching space

(*c*) The third control card (as well as the following ones) contain the vector $\mathbf{x}^L$

61–70 $\quad \mathbf{x}^L \quad$ the centre of the searching area

(*d*) Cards with starting values of vector $\mathbf{x}_0$ of variables (8E10.3). *N* starting values of the vector of variables is given on these cards in an increasing sequence. For instance for $N = 13$, the first computer card contains the starting values $x_{0,1} \div x_{0,8}$ and the second card contains values $x_{0,9} \div x_{0,13}$

(*e*) Subroutine GFUN is in the form

SUBROUTINE GFUN (**x**, *DELTJ*, N)
DIMENSION *x*(1) , *G*(1)
COMMON/*GGG*/*G*(100)

$$\begin{array}{l} : \\ : \\ :\ldots\ldots\ldots \\ : \\ : \\ : \end{array}\left\{\begin{array}{l}\text{statements for a calculation of functional}\\ \text{values of individual limitations}\\ G(1)\,,\,G(2)\,,\,\ldots.\,G(M)\\ \text{calculation of the functional value of the}\\ \text{preference function } G(M+1) \text{ in the form}\\ G(M+1) = f(\mathbf{x}) + DELTJ\end{array}\right.$$

RETURN
END

Note:
In the program general integer variables are used for the input and output statements to designate the input and output arrangement, i.e. *NI* and *NO* respectively. These numbers have to be defined by assignment statements at the beginning of the main program (for instance $NI = 2$ and $NO = 5$).

Note:
The part of the program marked * serves to print the course of the process as a whole. For solving practical problems, it is possible to leave this part of the program out, the program then executes more economically and it prints the achieved feasible regions only.

As illustration of the subroutine GFUN, the input values of problem 2 are in the program given.

Acknowledgement
The author wishes to express his appreciation to Mr K. Rohovsky for his help and effort in formulating the algorithm into FORTRAN.

```
      DIMENSION X0(100),XP(100),XK(100),GG(100),          XKAP(100),
     +          X1(100),XH(100),XZ(100),XL(100)
      COMMON /GFJ/ JX1
      COMMON /GGG/ G(100)

      NI=9
      NO=11
      READ(NI,100) N,M,KMV,KMC,IZM,KAPM
  100 FORMAT(6I5)
      WRITE(NO,101)N,M,KMV,KMC,IZM,KAPM
  101 FORMAT(1H1,5X,1HN,6X,1HM,5X,2HKM,4X,3HKMC,5X,2HZM,3X,4HKAPM/
     +6I7//)
      READ(NI,102) H,HVL,HVLC,D,DELTP,OMEGA
  102 FORMAT(8E10.3)
      WRITE(NO,103)H,HVL,HVLC,D,DELTP,OMEGA
  103 FORMAT(1H0,7X,1HH,9X,3HHVL,8X,4HHVLC,11X,1HD,
     +7X,5HDELTP,8X,5HOMEGA/6E12.4///4X,2HXL/)
      READ(NI,102)(X0(I),I=1,N)
      READ (NI,102) (XL(I),I=1,N)
      WRITE(NO,104) (XL(I),I=1,N)
  104 FORMAT(5E12.4)
      CALL ROMO(N,M,KMV,KMC,IZM,KAPM,NO,
     +          H,HVL,HVLC,D,DELTP,OMEGA,XL,
     +          X0,XP,XK,GG,XKAP,X1,XH,XZ)
      END

      SUBROUTINE ROMO(N,M,KMV,KMC,IZM,KAPM,NO,
     +          H,HVL,HVLC,D,DELTP,OMEGA,XL,
     +          X0,XP,XK,GG,XKAP,X1,XH,XZ)
      DIMENSION X0(N),XP(N),XK(N),GG(N),XKAP(N),X1(N),XH(N),XZ(N),
     +          XL(N)
      COMMON /GFJ/ JX1
      COMMON /GGG/ G(100)
      JX1=1
      DELTJ=0.
      CALL FBAR(M,X0,DELTJ,N,F0)
      WRITE(NO,104)F0
  104 FORMAT(1H0,3HF0=,E12.6,10X,*INITIAL VECTOR X0*)
      CALL TIVE(N,NO,X0)
      DO 20 I=1,N
   20 XKAP(I)=0.
    1 KAP=0
      CALL PRESUN(N,X0,XH)
      FH=F0
   11 IZ=0
      DO 21 I=1,N
   21 XZ(I)=0.
      KM=1
      CALL PRESUN(N,X0,X1)
      F1=F0
      IF(F0.EQ.0.) GOTO 7
    3 CALL GRAD(N,X1,XP,XK,GG,KM,KMC,M,DELTJ,F1,HVL,HVLC,H)
      IF(F1.EQ.0.) GOTO 12
      KM=KMV
      WRITE(NO,105)F1
  105 FORMAT(1H0,3HFG=,E12.6)
      CALL TIVE(N,NO,X1)
      IF(F0.GE.F1) GOTO 5
      DO 4 I=1,N
      X=X0(I)
      X0(I)=X1(I)
    4 X1(I)=X
      X=F0
      F0=F1
      F1=X
    5 IF(F1.GE.FH) GOTO 15
      CALL PRESUN(N,X1,XH)
      FH=F1
   15 DO 6 I=1,N
    6 XZ(I)=XZ(I)+X1(I)
      IZ=IZ+1
      IF(IZ.EQ.IZM) GOTO 8
      CALL PRIM(N,X0,X1,XK,GG,KM,KMC,M,DELTJ,F0,HVL,HVLC,D,F1)
      IF(F1.EQ.0.) GOTO 12
      IF(F0.EQ.0.) GOTO 7
      WRITE(NO,110)F0
  110 FORMAT(1H0,3HFP=,E12.6)
      CALL TIVE(N,NO,X0)

      WRITE(NO,106)F1
  106 FORMAT(1H0,3HF0=,E12.6)
      CALL TIVE(N,NO,X1)
      GOTO 3
   12 CALL PRESUN(N,X1,X0)
    7 IF(JX1.NE.1.) GOTO 30
      JX1=0
      CALL GFUN(X0,DELTJ,N)
      DELTJ=-G(M+1)
   30 DELTJ=DELTJ+DELTP
      CALL FBAR(M,X0,DELTJ,N,F0)
      WRITE(NO,107)F0
  107 FORMAT(1H0,3HF0=,E12.6,10X,*FEASIBLE DOMAIN*)
      CALL TIVE(N,NO,X0)
      GOTO 1
    8 IF(KAP.EQ.KAPM) GOTO 10
      KAP=KAP+1
      DO 9 I=1,N
      XKAP(I)=XKAP(I)+XZ(I)/FLOAT(IZ)
    9 X0(I)=(XL(I)-OMEGA*XKAP(I))/(1.-FLOAT(KAP)*OMEGA)
      CALL FBAR(M,X0,DELTJ,N,F0)
      WRITE(NO,108)F0
  108 FORMAT(1H0,3HF0=,E12.6,10X,*NEW VECTOR X0=X0*)
      CALL TIVE(N,NO,X0)
      GOTO 11
   10 CONTINUE
      RETURN
      END

      SUBROUTINE GRAD(N,X0,XP,XK,GG,KM,KMC,M,DELTJ,F0,HVL,HVLC,H)
      DIMENSION X0(N),XP(N),XK(N),GG(N)
      DO 1 I=1,N
      X0(I)=X0(I)+H
      CALL FBAR(M,X0,DELTJ,N,F1)
      X0(I)=X0(I)-H-H
      CALL FBAR(M,X0,DELTJ,N,F2)
      X0(I)=X0(I)+H
    1 GG(I)=(F1-F2)/(H+H)
      CALL XVYSI(N,X0,XP,XK,GG,KM,KMC,HVL,HVLC,C,M,DELTJ,F0)
      RETURN
      END

      SUBROUTINE PRIM(N,X0,X1,XK,GG,KM,KMC,M,DELTJ,F0,HVL,HVLC,D,F1)
      DIMENSION X0(N),X1(N),XK(N),GG(N)
      DO 1 I=1,N
    1 GG(I)=X0(I)-X1(I)
      CALL XVYSI(N,X0,X1,XK,GG,KM,KMC,HVL,HVLC,C,M,DELTJ,F0)
      ALFD =    D/C
      DO 4 I=1,N
    4 X1(I)=X0(I)-GG(I)*ALFD
      CALL FBAR(M,X1,DELTJ,N,F1)
      RETURN
      END

      SUBROUTINE XVYST(N,X0,XP,XK,GG,KM,KMC,HV,HVC,C,M,DELTJ,F0)
      DIMENSION X0(N),XP(N),GG(N),XK(N)
      C=0.
      DO 12 I=1,N
   12 C=C+GG(I)*GG(I)
      IF(C.NE.0.)GOTO 14
      DO 13 I=1,N
   13 GG(I)=1.
      C=FLOAT(N)
   14 C=SQRT(C)
      IPREP=0
    1 IPREP=IPREP+1
      CALL PRESUN(N,X0,XP)
      GOTO(2,3,7),IPREP
```

```
2 KP=1
  KK=KM
  ALFV=HV/C
  GOTO 4
3 KP=-KMC
  KK=KMC
  ALFV=HVC/C
4 DO 5 I=1,N
5 XK(I)=XP(I)-GG(I)*ALFV*FLOAT(KP)
  CALL FBAR(M,XK,DELTJ,N,FK)
  IF(FK.GE.F0) GOTO 6
  CALL PRESUN(N,XK,X0)
  F0=FK
  IF(F0.EQ.0.) GOTO 7
6 IF(KK.EQ.KP) GOTO 1
  KP=KP+1
  GOTO 4
7 RETURN
  END
```

```
  SUBROUTINE FBAR(M,X,DELTJ,N,F)
  DIMENSION X(N)
  COMMON /GFJ/ JX1
  COMMON /GGG/ G(100)
  CALL GFUN(X,DELTJ,N)
  F=0.
  M1=M+1
  IF(JX1.EQ.1) M1=M
  DO 1 I=1,M1
1 IF(G(I).GT.0.) F=F+G(I)
  RETURN
  END
```

```
  SUBROUTINE PRESUN(N,X1,X2)
  DIMENSION X1(N),X2(N)
  DO 1 I=1,N
1 X2(I)=X1(I)
  RETURN
  END
```

```
    SUBROUTINE TIVE(N,NO,X)
    DIMENSION X(N)
    NT=5
    IP=1
  1 IK=IP+NT-1
    IF(IK.GT.N)IK=N
    WRITE(NO,100)IP,IK,(X(I),I=IP,IK)
100 FORMAT(I5,3H  -,I4,10F10.4)
    IP=IP+NT
    IF(IP.LE.N) GOTO 1
    RETURN
    END
```

```
  SUBROUTINE GFUN(X,DELTJ,N)
  DIMENSION X(N)
  COMMON /GGG/ G(100)
  G(1)=2.*X(2)+0.03708*X(1)**4-0.8898*X(1)**3+6.674*X(1)**2
 +      -16.0202*X(1)-11.002
  G(2)=-2.*X(2)+0.5*X(1)+1.
  G(3)=0.0237756171 6*X(1)**4-0.734996505*X(1)**3+7.12039086*X(1)**2
 +      -23.78989107*X(1)+1.35*X(2)**2-10.2*X(2)+DELTJ
  RETURN
  END
```


References

FLETCHER, R. (1969). *Optimization*, Academic Press, London.

HIMMELBLAU, D. (1972). *Applied nonlinear programming*, McGraw-Hill.

Book reviews

Proceedings of the 1978 UKSC Conference on Computer Simulation, Chester, April 1978; 556 pages. (*IPC Science and Technology Press*, £26·00)

The conference was jointly sponsored by the UK Simulation Council and the (US) Society for Computer Simulation. As will be referred to later, there were some notable European contributions particularly from the Dutch school. Fifty-three papers in all are contained, all in their original form with no further editing or record of discussion. Whilst this has some drawbacks, it has made for remarkably rapid publication and the proceedings were available the same month as the conference.

The conference programme had an agreeably catholic range and spread. In many ways the general papers of the three open sessions look to have the most permanent value. One would mention here the seminal work of Dekker (the Dutch school) discussing the use of the new generation of parallel processors for simulation, the paper by Elzas (Holland again) entitled 'Whither simulation?'—and giving some answers to this apparently rhetorical question—as well as a session on the portability and universality of simulation languages.

Turning to techniques, there was further evidence—as if this was needed—of the role played by the digital computer in simulation, the cuckoo that may yet evict the original analogue occupant of the nest. Dickie and Ricketts for example had a paper detailing a comparison between 13 numerical integration routines: Runge-Kutta rides again. There were also sessions on simulation methodology and hardware aspects that touch on the fundamental question of whether simulation is or will ever be a discipline.

Other sessions were of the more conventional form: illustrative applications. It was interesting to see the use of the Van der Pol oscillator in haematology studies; perhaps we shall all have an artificial radar set inside us yet. The final session offered some workmanlike applications to control applications.

Your eagle-eyed reviewer spotted two minor typing errors in the papers (reproduced photographically of course): one gave us 'reactor' for 'vector'; another 'limped model' for 'lumped'—how often have my models indeed turned out to have a limp.

The overview of the conference proceedings suggests two disparate elements whose mixture, one hopes, was intellectually explosive; crude mechanics turning pots and handles with metaphysicians relating the model of the model to a model or, in Dekker's notation, 'systems' to 'sistems'. Have we simulated the dynamics of a conference?

J. LEWINS (London)

Syntactic Pattern Recognition: An Introduction, by R. C. Gonsalez and M. G. Thomason, 1978; 283 pages. (*Addison-Wesley*, $29.50, $17.50 paper)

The series editor's foreword describes this book as 'the first textbook written at an introductory level with emphasis on fundamentals of formal language and automata theory as they apply to pattern recognition and machine learning'. Syntactic pattern recognition has its origins in the crippling inability of decision theoretic pattern recognition systems to construct or use structural descriptions of the important relationships in a scene. The essential idea is to exploit the analogy between a class of images and well formed sentences generated by an appropriate grammar; then the desired structural description of the image corresponds to the parse structure of the corresponding sentence. After a brief introductory chapter, the authors present a clear account of formal language theory and its extensions to tree, web, and shape grammars. Most of the examples relate to some area of picture processing. Ledley's early grammar for chromosomes is well documented. A number of parsing algorithms for context-free grammars are given, as are their extensions to tree grammars. There is a chapter on stochastic automata motivated by the non-uniformity of the distribution of patterns and deviation from such patterns. The final chapter shows how a simple grammar might be inferred from a set of example sentences. Within its terms of reference, the book is clear, well written, with lucid examples.

There is however a significant lack of examples of working systems. Only the Moayer-Fu fingerprint system is described, with no data about its performance. As a field, syntactic pattern recognition seems too intent on developing its formalist framework, and too little concerned with computing the basics of perception, such as surface orientation, texture or movement. Given its roots it is surprising and sad to see that it equates perception with structured classification rather than the computation of useful structural descriptions.

J. M. BRADY (Colchester)