Arbitrarily shaped reinforced concrete members subject to biaxial bending and axial load

Article in Computers & Structures · November 1993

DOI: 10.1016/0045-7949(93)90069-P

2 authors:

Cengiz Dundar
Cukurova University
42 PUBLICATIONS 264 CITATIONS
SEE PROFILE

Besir Sahin
Cukurova University
1 PUBLICATION 33 CITATIONS
SEE PROFILE

Computers & Structures Vol. 49, No. 4, pp. 643–662, 1993
Printed in Great Britain.

0045-7949/93 $6.00 + 0.00
© 1993 Pergamon Press Ltd

ARBITRARILY SHAPED REINFORCED CONCRETE MEMBERS SUBJECT TO BIAXIAL BENDING AND AXIAL LOAD

C. Dundar and B. Sahin
Department of Civil Engineering, Cukurova University, 01330 Adana, Turkey

(*Received* 29 *June* 1992)

Abstract—An approach to the ultimate strength calculation and the dimensioning of arbitrarily shaped reinforced concrete sections, subject to combined biaxial bending and axial compression, is presented. The analysis is performed in accordance with the American Concrete Institute (ACI) code. A computer program is presented for rapid design of arbitrarily shaped reinforced concrete members under biaxial bending and axial load. In the proposed method the equilibrium equations are expressed in terms of the three unknowns, e.g. location of neutral axis and amount of total reinforcement area within the cross-section, which lead to three simultaneous nonlinear algebraic equations which are solved by a procedure based on the Newton–Raphson method. One design problem, available in the literature, is solved by this program to provide possible design procedures. A listing of the computer program is given in the Appendix.

1. INTRODUCTION

Arbitrarily shaped reinforced concrete members subjected to biaxial bending and axial compression are frequently used in multistorey tall buildings and bridge piers. These types of cross-sections are encountered as L-shaped columns at the corners of framed structures, and as channel- and hollow-box sections in staircases and elevator shafts. The design such members subjected to biaxial bending and axial load is more complex than rectangular shaped members with uniformly distributed reinforcement and is, therefore, treated inadequately or ignored by most designers in current practice. These members are usually over-designed which may cause the structure to be stiffer and may result in a loss of ductility when applied to seismic regions.

In recent years some methods have been presented for the ultimate strength analysis of various concrete sections, such as L-, T-, and channel-shaped and polygonal, under combined biaxial bending and axial compression [1–5]. These methods compute the ultimate flexural capacity of section. For design purposes they require trial and error procedures.

Several investigators [6, 7] have suggested a numerical solution for the design of arbitrarily shaped reinforced concrete members subjected to biaxial bending and axial compression and have developed computer programs. In these methods, a section with the reinforcement pattern and the area of total reinforcement have first to be assumed, then a trial adjustment procedure is required to find the inclination and depth of the neutral axis satisfying the equilibrium conditions, and then one has to compute the capacity of this particular section. The reinforcement area must then be successively corrected until the section capacity reaches the strength requirements. For designing irregular shaped sections such as L-, T- and channel-shaped, empirical approximations or design charts to represent the failure surfaces have been suggested [3–5].

The primary objective of this paper is to develop a computer program for the rapid design of arbitrarily shaped reinforced concrete members subjected to combined biaxial bending and axial load. This paper is an expansion of the previous papers [8, 9] covering the same problem in which arbitrarily shaped concrete members were not considered. In the proposed method, the equilibrium equations are expressed in terms of the three unknowns, e.g. location of modified axis and amount of total reinforcement area within the cross-section, which leads to three simultaneous nonlinear algebraic equations which are solved by a procedure based on the Newton–Raphson method.

The proposed design method is based on the following assumptions: (1) sections remain plane after deformation; (2) the reinforcement is subjected to the same variations in strain as the adjacent concrete; (3) the maximum strain in the extreme fibre of concrete in compression is 0.003; (4) the distribution of concrete stress of $0.85f'_c$ and a depth from the compressed edge k_1x_0, where x_0 is the neutral axis depth and the value of k_1 is taken according to recommendations of ACI code; (5) the effect of creep and the tensile strength of concrete and any direct tension stresses due to shrinkage, etc. are ignored; (6) the stress–strain relation for the reinforcing bar may be considered as for cold-work steel and/or mild steel; (7) shear deformation is neglected; and (8) the member does not buckle before ultimate load is attained.

Finally, a computer program based on the assumptions stated above, is described briefly and one design problem, available in the literature, is solved by the computer program presented here to provide possible design procedures.

2. PROBLEM FORMULATION

Material properties

In the analysis two types of steel may be used with different stress–strain relationships.

The stress–strain relation for the mild steel is assumed to be elasto-plastic and is determined as follows:

Calculation of strains for the reinforcing bars

Consider an arbitrarily shaped reinforced concrete member, channel-shaped for instance, subject to a biaxially eccentric load as shown in Fig. 2. The origin of the x–y axis system is chosen to be the most heavily stressed point of the cross-section. This point is determined with respect to the location of the biaxially eccentric load $N_d(x_N, Y_N)$.

The assumption, that plane sections remain plane and there is no bond–slip between the reinforcement and the concrete, implies that the strain distribution is linear across the reinforced concrete section. The strain in the steel may be found by

$$|\epsilon_s| < \epsilon_y, \quad f_s = E_s \epsilon_s \tag{1a}$$

$$|\epsilon_s| \geqslant \epsilon_y, \quad f_s = \frac{\epsilon_s}{|\epsilon_s|} f_y, \tag{1b}$$

where f_s and ϵ_s are the stress and strain values of the reinforcing steel, f_y and E_s are the yield strength and the modulus of elasticity of steel, respectively.

For the case of using cold-work steel, the stress–strain relationship is determined by six points as shown in Fig. 1. First two points and the last two define the linear region and the remaining points define the nonlinear region of the stress–strain curve. Stress values for intermediate strains are computed by using the Lagrange interpolation method.

In the analysis the distribution of concrete stress is assumed to be rectangular with a main stress of $0.85f'_c$ and a depth from the most heavily compressed edge of $k_1 x_0$, where x_0 is the neutral axis depth. The value of k_1 is determined as follows:

$$k_1 = 0.85, \quad f'_c \leqslant 27.6 \text{ MPa} \tag{2a}$$

$$k_1 = 0.85 - 0.0073(f'_c - 27.6), \quad f'_c > 27.6 \text{ MPa} \tag{2b}$$

in which f'_c is the ultimate compression strength of concrete.

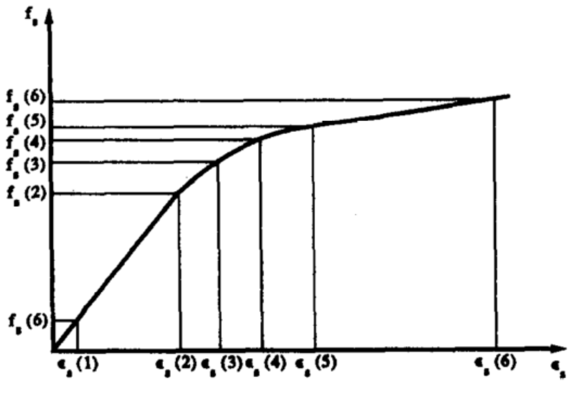

Fig. 1. Stress–strain relation for cold-work steel.

considering the similar triangles of the strain diagram of Fig. 1 as

$$\epsilon_{si} = \epsilon_{cu}\left(\frac{z_i}{x_0} - 1\right). \tag{3}$$

Here $z_i = (y_i + x_i \tan\alpha)\cos\alpha$; $\tan\alpha = c/a$; $\cos\alpha = k_1 x_0/c$, in which a and c are the horizontal and vertical distances between the origin of the x–y axis system and the modified axis (see Fig. 1). The strain, ϵ_{si}, for the ith reinforcing bar can now be determined by using eqn (3) as

$$\epsilon_{si} = \epsilon_{cu}\left[k_1\left(\frac{y_i}{c} + \frac{x_i}{a}\right) - 1\right], \tag{4}$$

where x_i and y_i are the coordinates of the ith reinforcing bar with respect to the x–y axis system which has the origin at the most heavily stressed corner of the cross-section.

Determination of the geometric properties of the concrete compression zone

In order to determine compressive force and its location in the concrete compression zone, the area and the coordinates of the centroid of the compression zone must be computed. The compression zone will now be described in terms of the distances a and c. Consider an arbitrarily shaped section in the x–y axis system as shown in Fig. 3. The modified axis is located with the distances a and c from the origin of the x–y axis system; the compression zone of the section is indicated above this axis by the shaded area. To determine the area of the compression zone (A_{cc}) and the coordinates of its centroid $(\bar{x}, \bar{y})$, the nodes of the cross-section are numbered in the clockwise or anticlockwise direction from 1 to n. Then, the points of intersection with the modified axis and the sides of the boundary are indicated as $i, i+1$ or $i-1, i$ if the intersection takes place in going from i to $i+1$ or $i-1$ to i, respectively.

With these points of intersection the shaded area

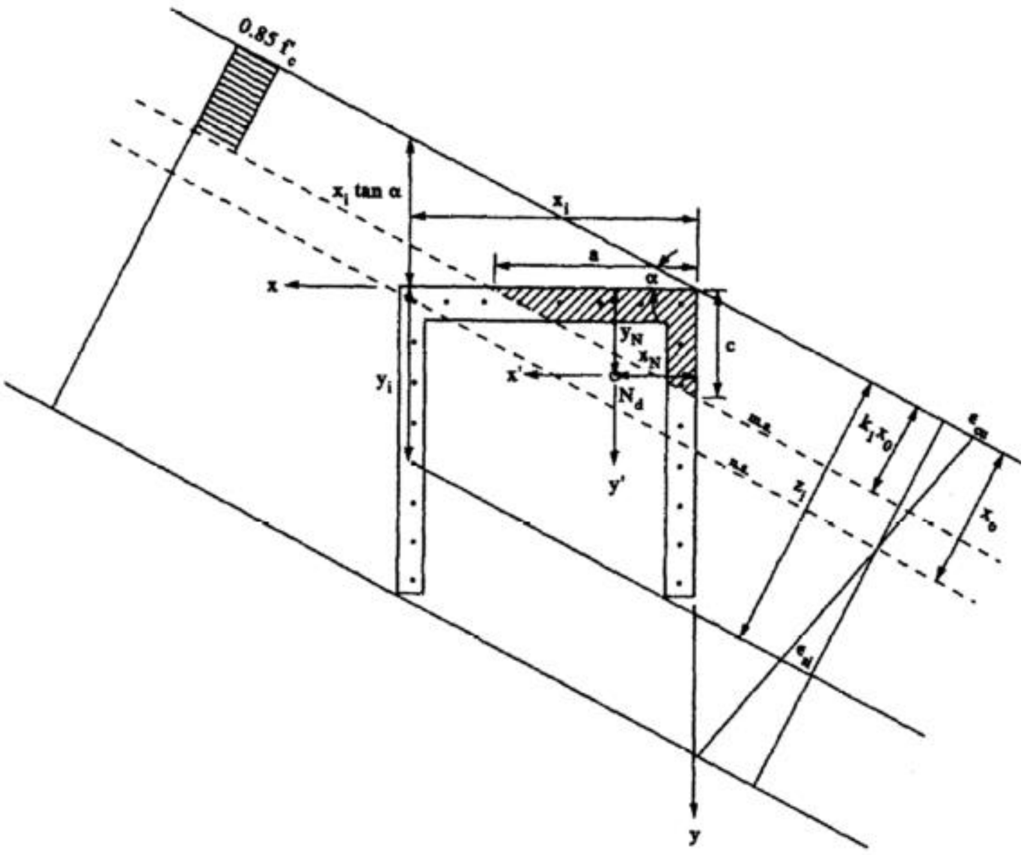

Fig. 2. Stresses and strains in concrete section under biaxial bending and axial compression.

can be described in terms of the triangles. Now the coordinates of the points intersection for i, $i+1$ are

$$x_{i,i+1} = \frac{c - y_i + \delta_{i,i+1} x_i}{c/a + \delta_{i,i+1}} \tag{5a}$$

$$y_{i,i+1} = c\left(1 - \frac{x_{i,i+1}}{a}\right), \tag{5b}$$

where

$$\delta_{i,i+1} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{5c}$$

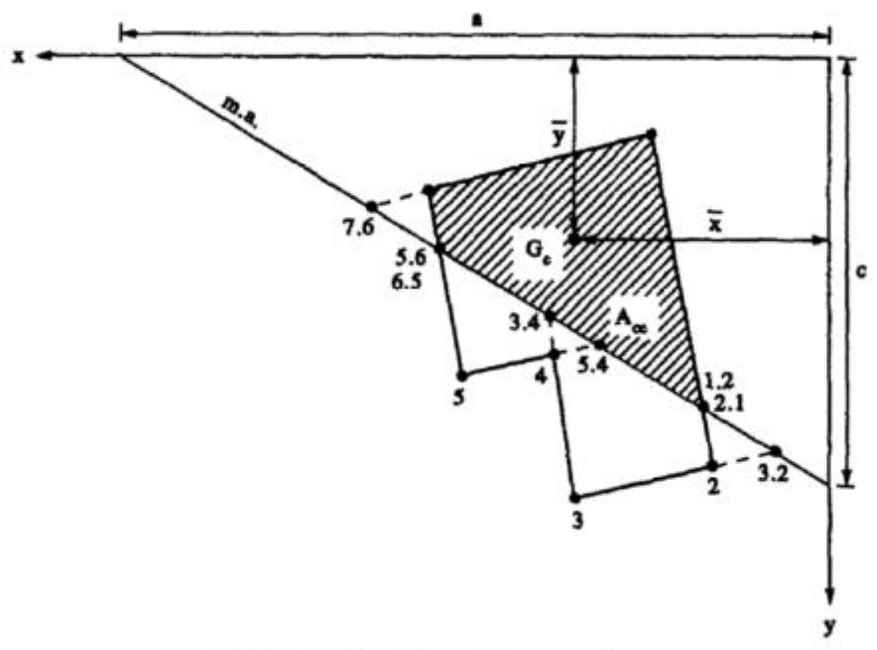

Fig. 3. Determination of concrete compression zone.

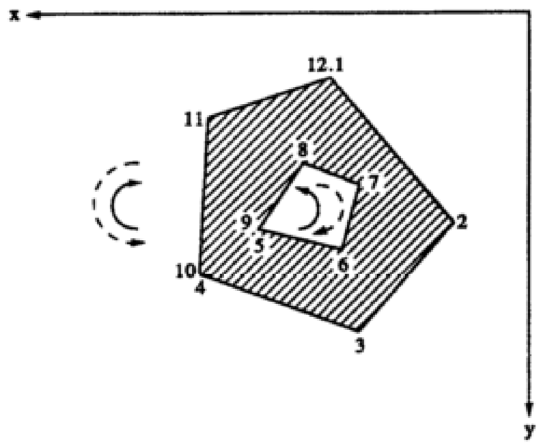

Fig. 4. Concrete section with void.

and for $i-1, i$

$$x_{i-1,i} = \frac{c - y_i + \delta_{i-1,i} x_i}{c/a + \delta_{i-1,i}} \tag{6a}$$

$$y_{i-1,i} = c\left(1 - \frac{x_{i-1,i}}{a}\right), \tag{6b}$$

where

$$\delta_{i-1,i} = \frac{y_{i-1} - y_i}{x_{i-1} - x_i} \tag{6c}$$

may be obtained.

the boundary of the void. Hence, the nodal point numbering must proceed in the opposite direction to that of the external boundary of the cross-section as indicated in Fig. 4.

Equilibrium equations

If the coordinates of the biaxially eccentric load are indicated by x_N and y_N in the x–y axis system, equilibrium equations can be written with respect to the axis system x'–y', through the biaxially eccentric load and parallel to x and y axes, respectively. Therefore, equilibrium equations can be written as follows:

$$f_1 = f_c'' A_{cc} - \frac{A_{st}}{n} \sum_i^n f_{si} - N_d = 0 \tag{9a}$$

$$f_2 = \frac{A_{st}}{n} \sum_i^n [(x_i - x_N) f_{si}] - f_c''(\bar{x} - x_N) A_{cc} = 0 \tag{9b}$$

$$f_3 = \frac{A_{st}}{n} \sum_i^n [(y_i - y_N) f_{si}] - f_c''(\bar{y} - y_N) A_{cc} = 0 \tag{9c}$$

in which $f_c'' = 0.85 f_c'$, A_{st} is the total area of the main reinforcement within the cross-section, and x_i and y_i are the coordinates of the reinforcing bars with respect to the x–y axis system. It is assumed that the reinforcing bars are identical.

In view of eqns (1), (4) and (7), eqn (9) can now be expressed in terms of the parameters a, c, A_{st} as follows:

The area of the concrete compression zone can be computed as follows:

$$A_{cc} = |\sum_i A_i|, \tag{7a}$$

where A_i is the area of triangle $(i-1)$ (i) $(i+1)$

$$A_i = \tfrac{1}{2}[x_i(y_{i-1,i} - y_{i,i+1}) + x_{i,i+1}(y_i - y_{i-1,i}) + x_{i-1,i}(y_{i,i+1} - y_i)]. \tag{7b}$$

The triangles that form below the modified axis are not considered. Finally, the centroid of the shaded area with respect to x–y axis system can be written as follows:

$$\bar{x} = \frac{1}{3\sum A_i}\sum_i [A_i(x_i + x_{i,i+1} + x_{i-1,i})] \tag{8a}$$

$$\bar{y} = \frac{1}{3\sum A_i}\sum_i [A_i(y_i + y_{i,i+1} + y_{i-1,i})]. \tag{8b}$$

Cross-sections with voids can easily be described by nodal coordinates, defining the boundary. If nodal point numbering is done in the clockwise direction for the outer boundary of the section, nodal point numbering must be done in the anticlockwise direction for

$$f_1(a, c, A_{st}) = 0 \tag{10a}$$

$$f_2(a, c, A_{st}) = 0 \tag{10b}$$

$$f_3(a, c, A_{st}) = 0 \tag{10c}$$

or in matrix form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \tag{11}$$

where $\mathbf{x} = [a, c, A_{st}]^t$; $\mathbf{f} = [f_1, f_2, f_3]^t$. Here the superscript t designates transpose. The recursive formula of the Newton–Raphson method for finding the root of eqn (11) is

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \Delta\mathbf{x}_i, \quad i = 0, 1, 2, \ldots, \tag{12}$$

where the increment $\Delta\mathbf{x}_i$ is the solution of the equation

$$\mathbf{A}_i \Delta\mathbf{x}_i = \mathbf{f}_i \tag{13}$$

in which $\mathbf{A}_i = \partial\mathbf{f}(\mathbf{x}_i)/\partial\mathbf{x}$ (Jacobian matrix); $\mathbf{f}_i = \mathbf{f}(\mathbf{x}_i)$. In the determination of the Jacobian matrix, the partial derivatives are expressed in terms of finite differences.

The algorithm presented above requires us to start with an assumed triplet $[a, c, A_{st}]^t$. The third component of the starting triplet can be chosen to be $A_{st} = 0.01A_c$ since the minimum requirement of the steel percentage in the gross cross-sectional area is 1% according to the ACI building code [10].

Initial approximations for a and c are chosen by the user of the computer program. If the initial values of a and c are taken to be the length of contour lines which intersect at the origin of the x–y axis system, it is observed that the sequence generated by the above-mentioned iterative algorithm has converged for many different problems tested in six to seven iterations. If the iterative algorithm diverges then the program gives a warning that the user should choose new initial values for a and c.

Regarding the domain restriction for the solution, if the section selected by the user is an unnecessarily big one, then the iterative algorithm may converge to a negative value for the reinforcement. In this case, the program gives a warning again that the user should either decrease the section dimensions or use the minimum percentage of steel reinforcement required by the building code.

3. DESIGN EXAMPLE

Let us consider a previously studied staircase core section [6] for a compression member subject to biaxial bending to carry the following design loads

$$N_d = 1737.14 \text{ kips}, \quad (7731.1 \text{ kN})$$

$$M_x = 103{,}714.3 \text{ kips-in}, \quad (11{,}725.5 \text{ kN-m})$$

$$M_y = 94{,}971.4 \text{ kips-in}, \quad (10{,}737.0 \text{ kN-m}). \tag{14}$$

The material properties will be taken as

$$f'_c = 4 \text{ ksi}, \quad (27.6 \text{ MPa})$$

$$f_y = 31.9 \text{ ksi}, \quad (220.1 \text{ MPa}). \tag{15}$$

Solution

Step 1: try a staircase core section as shown in Fig. 5.

Step 2: try 84-bar reinforcement as shown in Fig. 5.

Step 3: the computer program requires the input data given below.

The coordinates (x_N, y_N) must first be determined from the design loads as follows:

$$e_x = \frac{94{,}971.4}{1737.14} = 54.67 \text{ in}, \quad (1388.62 \text{ mm})$$

$$x_N = 49.20 - 54.67 = -5.47 \text{ in}, \quad (-138.94 \text{ mm}) \tag{16a}$$

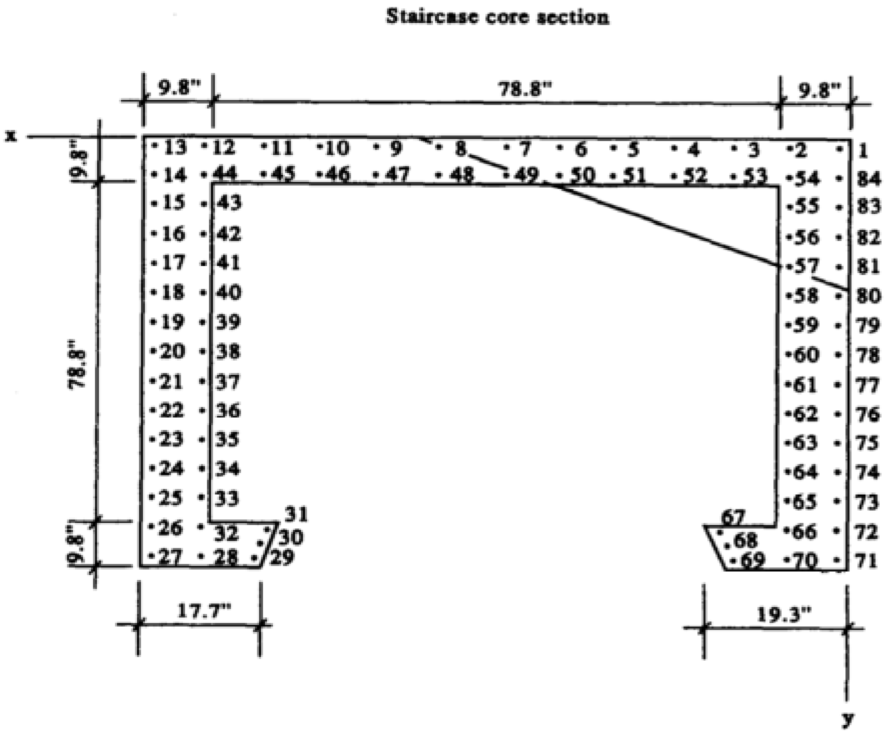

Fig. 5. Graphical output of example problem (1 in = 25.4 mm).

Table 1. Echo print of input data

```
Staircase core section (Magalhaes)
General information
  k1 = 0.85   Epscu = 0.003   fyd = 31900   fcd = 4000
  Es = 29e+6   Nd = 1737142.9   xN = -5.467   yN = -19.79
  ao = 40   co = 40
:
Polygon coordinates
P=1    X=0       Y=0
P=2    X=98.4
P=3              Y=98.4
P=4    X=80.7
P=5    X=79.1    Y=88.6
P=6    X=88.6
P=7              Y=9.8
P=8    X=9.8
P=9              Y=88.6
P=10   X=19.3
P=11   X=17.7    Y=98.4
P=12             X=0.0
:
Reinforcements
R1=1    R2=13   X1=1      Y1=1      X2=97.4   Y2=1      Milds=1
R1=13   R2=27   X1=97.4   Y1=1      X2=97.4   Y2=97.4
R1=27   R2=29   X1=97.4   Y1=97.4   X2=81.7   Y2=97.4
R1=29   R2=31   X1=81.6   Y1=97.4   X2=80.1   Y2=89.6
R1=32   R2=44   X1=89.6   Y1=89.6   X2=89.6   Y2=8.8
R1=44   R2=54   X1=89.6   Y1=8.8    X2=8.8    Y2=8.8
R1=54   R2=66   X1=8.8    Y1=8.8    X2=8.8    Y2=89.6
R1=67   R2=69   X1=18.3   Y1=89.6   X2=16.7   Y2=97.4
R1=69   R2=71   X1=16.7   Y1=97.4   X2=1      Y2=97.4
R1=71   R2=84   X1=1      Y1=97.4   X2=1      Y2=8.8
:
End of data
```

$$e_y = \frac{103{,}714.3}{1737.14} = 59.70 \text{ in}, \quad (1516.38 \text{ mm})$$

$$y_N = 39.91 - 59.70 = -19.79 \text{ in}, \quad (-502.67 \text{ mm}). \tag{16b}$$

The coordinates of the centroid of the gross cross-section with respect to the x–y axis system (x_c, y_c) are 49.20 and 39.91 in, respectively, which are automatically computed by the program.

The input data of the example problem are given in Table 1. The input data have three data blocks which are: 'general information', 'polygon coordinates' and 'reinforcements'. For the 'reinforcements' block, the program has the facility of data generation. For example, to generate the coordinates of reinforcing bars, first coordinates and the last coordinates may be given. So the program equally spaces the intermediate reinforcing bars in the section. The coordinates of the reinforcing bars may also be given one by one.

To specify the type of steel used for the section, the parameters 'milds' for mild steel, 'coldws' for cold-work steel are used, respectively.

If the program is run with the input data given in Table 1, an output whose echo print is given in Table 2 is produced. The program also has a graphical output which indicates the section geometry, location of the reinforcing bars with corresponding numbers and the position of the modified axis with respect to x–y axis system.

The numerical procedure converges to the solution $a = 59.69$ in (1516.12 mm), $c = 34.23$ in (869.44 mm), $A_{st} = 49.02 \text{ in}^2$ (31,625.74 mm^2) in five iterations. $\|\Delta \mathbf{x}_i\| \leqslant \epsilon$ is used in the program as the convergence criterion, where ϵ is the convergence factor which is chosen to be 10^{-3}.

Step 4: choose 84 # 7 bars, $\rho_g = A_{st}/A_c = (84 \times 0.6)/2871.4 = 0.0175$, say 1.75% which is between 1% and 8% of the gross area as permitted by the ACI building code.

Step 5: design the ties as required by the ACI building code.

This example problem has also been solved by Magalhaes [6]. In this study, following the determination of the position of the neutral axis by iterative numerical algorithm, the ultimate capacity of the section is computed. The reinforcement area is then successively corrected until the section capacity reaches the strength requirement. The area of the total reinforcement was found to be 48 in^2 (30,967.68 mm^2) by Magalhaes.

The computer program, presented here, computes the location of neutral axis and the total area of the reinforcement in one iterative procedure.

A computer program which is developed for the ultimate strength design of arbitrarily shaped

Table 2. Echo print of program output

```
Staircase core section (Magalhaes)
General information
k1=0.85   εcu=0.003   fcd=4000   fyd=31900
Es=29,000,000
Nd=1,737,142.9   yn=-19.79   xn=-5.467
ao=40   co=40

Polygon coordinates
P=1     X=0.000   Y=0.000
P=12    X=0.000   Y=98.400

Reinforcements
Ry=1     X=1.000   Y=1.000
Ry=84    X=1.000   Y=8.800

Iteration
   a=40.000   c=40.000   Ast=28.713999
1, a=64.081   c=28.256   Ast=45.022427
2, a=58.795   c=34.319   Ast=48.362925
3, a=59.693   c=34.223   Ast=49.021301
4, a=59.690   c=34.228   Ast=49.025357
5, a=59.690   c=34.228   Ast=49.025358

Reinforcement stresses
R=1    σs=-31,900.000    R=43   σs=31,900.000
R=42   σs=31,900.000     R=84   σs=-31,900.000

Section
x=49.200   Ac=2871.400
y=39.908

Compression zone
a=59.690   x=19.535   Acc=713.081
c=34.228   y=9.393

Total reinforcement area=49.025358
```

reinforced concrete members under biaxial bending and axial compression is prepared in QuickBASIC. The full listing of the computer program is provided in the Appendix.

REFERENCES

1. T. Brondum-Nielsen, Ultimate flexural capacity of cracked polygonal concrete sections under biaxial bending. *ACI Jnl* **82**, 863–869 (1985).
2. T. Brondum-Nielsen, Ultimate flexural capacity of fully prestressed, partially prestressed, arbitrary concrete sections under symmetric bending. *ACI Jnl* **83**, 29–35 (1986).
3. T. C. Hsu, Biaxially loaded L-shaped reinforced concrete columns. *J. Struct. Engng, ASCE* **111**, 2576–2598 (1985).
4. T. C. Hsu, Channel-shaped reinforced concrete compression members under biaxial bending. *ACI Jnl* **84**, 201–211 (1987).
5. T. C. Hsu, T-shaped reinforced concrete members under biaxial bending and axial compression. *ACI Jnl* **86**, 460–468 (1989).
6. M. P. Magalhaes, Biaxially loaded concrete sections. *J. Struct. Engng, ASCE* **105**, 2639–2656 (1979).
7. K. H. Kwan and T. C. Liauw, Computer aided design of reinforced concrete members subjected to axial compression and biaxial bending. *The Structural Engineer* **63B**, 34–40 (1985).
8. C. Dundar, Computer aided ultimate strength design of columns subject to biaxial bending. *Les Annales de l'Enit* **2**, 57–69 (1988).
9. C. Dundar, Concrete box sections under biaxial bending and axial load. *J. Struct. Engng, ASCE* **116**, 860–865 (1990).
10. Building Code Requirements for Reinforced Concrete, Report No. 318-71, ACI, Detroit, MI (1974).

APPENDIX 1

APPENDIX : LISTING OF THE COMPUTER PROGRAM

```
DEFDBL A-Z
DEFINT I-N
DEFSTR U
DECLARE SUB Reads (U, Uoku, okun, Ls)
DECLARE SUB Plot (Ugn, UDat)
DECLARE SUB Area ()
DECLARE SUB Prep (f1, f2, f3, dk1, dNp)
DECLARE SUB Gnrt (U, Itype, n, Ls)
COMMON SHARED Epscu, fyd, fcd, Es, Ac, x1, y1, yn, xn, Eps(), z()
COMMON SHARED x(), y(), Sigma(), Ast, np, nd, a, c, Itype()
CONST npara = 6000, FALSE = 0, TRUE = NOT FALSE
DIM x(npara), y(npara), Sigma(npara / 2), Itype(npara / 2), Eps(6),z(6)
KEY 15, CHR$(&H0) + CHR$(&H1)     'ESC key
ON KEY(10) GOSUB Ciz
ON KEY(15) GOSUB Son
PRINT
IF COMMAND$ = "" THEN INPUT "Input File = ", UDat ELSE UDat = COMMAND$
OPEN UDat FOR INPUT AS 1
FOR m = 1 TO LEN(UDat)
  U = MID$(UDat, m, 1)
  IF U = "." THEN EXIT FOR
  UDos = UDos + U
NEXT m
```

```
UDat = UDos + MID$(UDat, m, 4)
UDos = UCASE$(UDos) + ".OUT"
KEY(10) ON
KEY(15) ON
CLS
LINE INPUT #1, Ugn
COLOR 15
PRINT Ugn
LOCATE 25, 1
PRINT "Esc"; : COLOR 7: PRINT " Exit, ";
COLOR 15: PRINT "  F10"; : COLOR 7: PRINT " Plot";
COLOR 15: PRINT TAB(52); "Output File = "; UDos;
COLOR 7
VIEW PRINT 3 TO 23
GOSUB Format
DO
  LINE INPUT #1, Ug: Ug = UCASE$(Ug)
  IF Ug = "END OF DATA" THEN EXIT DO
  PRINT
  SELECT CASE Ug
    CASE "GENERAL INFORMATION"
      PRINT Ug
      LOCATE CSRLIN, 1: PRINT "Reading..."; : LOCATE CSRLIN, 1
```

```
DO
  LINE INPUT #1, U: IF U = ":" THEN EXIT DO
  CALL Reads(U, "K1", okun, Ls): IF Ls THEN dk1 = okun
  CALL Reads(U, "EPSCU", okun, Ls): IF Ls THEN Epscu = okun
  CALL Reads(U, "FYD", okun, Ls): IF Ls THEN fyd = okun
  CALL Reads(U, "FCD", okun, Ls): IF Ls THEN fcd = okun
```

```
    CALL Reads(U, "ES", okun, Ls): IF Ls THEN Es = okun
    CALL Reads(U, "ND", okun, Ls): IF Ls THEN dNp = okun
    CALL Reads(U, "YN", okun, Ls): IF Ls THEN yn = okun
    CALL Reads(U, "XN", okun, Ls): IF Ls THEN xn = okun
    CALL Reads(U, "AO", okun, Ls): IF Ls THEN ao = okun
    CALL Reads(U, "CO", okun, Ls): IF Ls THEN co = okun
    FOR j = 1 TO 6
      CALL Reads(U, "EPS(" + RIGHT$(STR$(j), 1) + ")", okun, Ls)
      IF Ls THEN Eps(j) = okun
      CALL Reads(U, "Z(" + RIGHT$(STR$(j), 1) + ")", okun, Ls)
      IF Ls THEN z(j) = okun
    NEXT j
  LOOP
  PRINT "k1 ="; dk1; TAB(17); "εcu ="; Epscu; TAB(36); "fcd =";fcd;
  PRINT TAB(55); "fyd ="; fyd
  PRINT "Es ="; Es
  PRINT "Np ="; dNp; TAB(17); "yn ="; yn; TAB(36); "xn ="; xn
  PRINT "ao ="; ao; TAB(17); "co ="; co
  FOR j = 1 TO 6
    PRINT "Eps(" + RIGHT$(STR$(j), 1) + ") ="; Eps(j);
    PRINT TAB(36); "z(" + RIGHT$(STR$(j), 1) + ") ="; z(j)
  NEXT j

CASE "POLYGON COORDINATES"
  PRINT Ug
  DO
    LINE INPUT #1, U: IF U = ":" THEN EXIT DO
    CALL Reads(U, "P", okun, Ls): np = okun
    IF NOT Ls THEN
      PRINT : PRINT "There is no information about 'P'...": GOTO Er
    END IF
    IF np <> i + 1 THEN
      PRINT : PRINT "The order of P is not correct...": GOTO Er
    ELSE
      i = np
    END IF
    CALL Reads(U, "X", okun, Ls): x(np) = okun
    IF NOT Ls THEN
      IF np = 1 THEN
        PRINT :PRINT "There is no information about 'X'...":GOTO Er
      ELSE
        x(np) = x(np - 1)
      END IF
    END IF
    CALL Reads(U, "Y", okun, Ls): y(np) = okun
    IF NOT Ls THEN
      IF np = 1 THEN
        PRINT :PRINT "There is no information about 'Y'...":GOTO Er
      ELSE
      y(np) = y(np - 1)
     END IF
   END IF
   PRINT "P ="; np;
   PRINT TAB(11); "X ="; USING U43; x(np);
   PRINT TAB(28); "Y ="; USING U43; y(np)
 LOOP

CASE "REINFORCEMENTS"
  PRINT Ug
  LOCATE CSRLIN, 1: PRINT "Reading..."; : LOCATE CSRLIN, 1
  IF np = 0 THEN
    PRINT
    PRINT "Please, input polygon coordinates firstly...": GOTO Er
  END IF
  Itype = 1
  DO
    LINE INPUT #1, U: IF U = ":" THEN EXIT DO
    CALL Reads(U, "R", okun, Ls)
    IF Ls THEN n = okun ELSE n = 0
    CALL Reads(U, "MILDS", okun, Ls)
    IF Ls THEN
      Itype(n) = TRUE
    ELSE
      CALL Reads(U, "COLDWS", okun, Ls)
      IF Ls THEN Itype(n) = FALSE ELSE Itype(n) = Itype
    END IF
    Itype = Itype(n)
    IF Itype = 1 THEN
      PRINT
      PRINT "There is no information about 'COLDWS' or 'MILDS' ..."
      GOTO Er
    END IF
```

```
CALL Reads(U, "R1", okun, Ls)
IF Ls THEN
  CALL Gnrt(U, Itype, n, Ls)
  IF NOT Ls THEN PRINT:PRINT "R1 is greater than R2...":GOTO Er
ELSE
  CALL Reads(U, "X", okun, Ls): x(np + n) = okun
  IF NOT Ls THEN x(np + n) = xtemp
  xtemp = x(np + n)
  CALL Reads(U, "Y", okun, Ls): y(np + n) = okun
  IF NOT Ls THEN y(np + n) = ytemp
  ytemp = y(np + n)
END IF
    IF nd < n THEN nd = n
  LOOP
  FOR l = 1 TO nd
    PRINT "R";
    IF Itype(l) THEN PRINT "y ="; l;  ELSE PRINT "s ="; l;
    PRINT TAB(11); "X ="; USING U43; x(np + l);
    PRINT TAB(28); "Y ="; USING U43; y(np + l)
  NEXT l

   CASE ELSE
     PRINT :PRINT "No need the data block named '";Ug; "'...": GOTO Er
 END SELECT
LOOP
CLOSE 1
IF nd = 0 THEN PRINT : PRINT "Please, input reinforcements...": GOTO Er
PRINT Ug: PRINT
OPEN UDos FOR OUTPUT AS 2
PRINT #2, UDos
PRINT #2,
PRINT #2, Ugn
PRINT #2,
PRINT #2, "GENERAL INFORMATION :"
PRINT #2, "k1 ="; dk1; TAB(17); "εcu ="; Epscu; TAB(36); "fcd ="; fcd;
PRINT #2, TAB(55); "fyd ="; fyd
PRINT #2, "Es ="; Es
PRINT #2, "Np ="; dNp; TAB(17); "yn ="; yn; TAB(36); "xn ="; xn
PRINT #2, "ao ="; ao; TAB(17); "co ="; co
FOR j = 1 TO 6
  PRINT #2, "Eps(" + RIGHT$(STR$(j), 1) + ") ="; Eps(j);
  PRINT #2, TAB(36); "z(" + RIGHT$(STR$(j), 1) + ") ="; z(j)
NEXT j
PRINT #2,
fcd = .85 * fcd
zmaxx = x(1)
zmaxy = y(1)
zminx = zmaxx
zminy = zmaxy
FOR i = 1 TO np
  IF zmaxx < x(i) THEN zmaxx = x(i)
  IF zmaxy < y(i) THEN zmaxy = y(i)
  IF zminx > x(i) THEN zminx = x(i)
  IF zminy > y(i) THEN zminy = y(i)
NEXT i

FOR i = 1 TO np + nd
  x(i) = x(i) - zminx
  y(i) = y(i) - zminy
NEXT i
zmaxx = zmaxx - zminx
zmaxy = zmaxy - zminy
x(0) = x(np): y(0) = y(np)
a = zmaxx + zmaxy: a = 2 * a
c = a
CALL Area
PRINT "Center of Gravity : x ="; USING U43 + " "; xl
PRINT "                     y ="; USING U43; yl
PRINT
IF xn > xl THEN   'x -Axis Conversion
  FOR i = 1 TO np + nd
    x(i) = zmaxx - x(i)
  NEXT i
  xn = zmaxx - xn
  xl = zmaxx - xl
  x(0) = x(np)
  PRINT "x-coord. transformed w.r.t. most heavily stressed point..."
  PRINT
  PRINT #2,"x-coord. transformed w.r.t. most heavily stressed point..."
  PRINT #2,
END IF
```

```
IF yn > yl THEN     'y -Axis Conversion
  FOR i = 1 TO np + nd
    y(i) = zmaxy - y(i)
  NEXT i
  yn = zmaxy - yn
  dMy = -dMy


   yl = zmaxy - yl
   y(0) = y(np)
   PRINT "y-coord. transformed w.r.t. most heavily stressed point..."
   PRINT
   PRINT #2,"x-coord. transformed w.r.t. most heavily stressed point..."
   PRINT #2,
 END IF
 PRINT #2, "POLYGON COORDINATES :"
 FOR i = 1 TO np
   PRINT #2, "P ="; i;
   PRINT #2, TAB(11); "X ="; USING U43; x(i);
   PRINT #2, TAB(28); "Y ="; USING U43; y(i)
 NEXT i
 PRINT #2,
 PRINT #2, "REINFORCEMENTS :"
 FOR i = 1 TO nd
   PRINT #2, "R";
   IF Itype(i) THEN PRINT #2, "y ="; i;  ELSE PRINT #2, "s ="; i;
   PRINT #2, TAB(11); "X ="; USING U43; x(np + i);
   PRINT #2, TAB(28); "Y ="; USING U43; y(np + i)
 NEXT i
 PRINT #2,
 PRINT "Press any key to continue..."
 WHILE INKEY$ = ""
 WEND
```

```
LOCATE CSRLIN - 1, 1
Actemp = Ac
xltemp = xl
yltemp = yl
'
' Newton-Raphson
' Iteration begins...
'
a = ao: c = co: Ast = Ac * .01
PRINT TAB(12); "a ="; USING U43; a;
PRINT TAB(31); "c ="; USING U43; c;
PRINT TAB(49); "Ast ="; USING U46; Ast
PRINT #2, "ITERATION :"
PRINT #2, TAB(12); "a ="; USING U43; a;
PRINT #2, TAB(31); "c ="; USING U43; c;
PRINT #2, TAB(49); "Ast ="; USING U46; Ast
dela = .001: delc = .001: delAst = .000001
atemp = a
ctemp = c
Asttemp = Ast
CALL Prep(f1, f2, f3, dkl, dNp)
rf1 = f1: rf2 = f2: rf3 = f3
atmp = a + dela
ctmp = c + delc
Asttmp = Ast + delAst
DO

 Iter = Iter + 1
 PRINT Iter; ",";
 PRINT #2, Iter; ",";
 c = ctemp: a = atmp: Ast = Asttemp
```

```
  CALL Prep(f1, f2, f3, dk1, dNp)
  df1a = (f1 - rf1) / dela
  df2a = (f2 - rf2) / dela
  df3a = (f3 - rf3) / dela
  c = ctmp: a = atemp: Ast = Asttemp
  CALL Prep(f1, f2, f3, dk1, dNp)
  df1c = (f1 - rf1) / delc
  df2c = (f2 - rf2) / delc
  df3c = (f3 - rf3) / delc
  c = ctemp: a = atemp: Ast = Asttmp
  CALL Prep(f1, f2, f3, dk1, dNp)
  df1Ast = (f1 - rf1) / delAst
  df2Ast = (f2 - rf2) / delAst
  df3Ast = (f3 - rf3) / delAst
  Di = df1a * (df2c * df3Ast - df2Ast * df3c) - df2a * (df1c * df3Ast -
df1Ast * df3c) + df3a * (df1c * df2Ast - df1Ast * df2c)
  Deltaa = (df1c * df2Ast * rf3 - df1c * df3Ast * rf2 - df2c * df1Ast *
rf3 + df2c * df3Ast * rf1 + df3c * df1Ast * rf2 - df3c * df2Ast*rf1)/Di
  Deltac = -(df1Ast * rf2 * df3a - df1Ast * rf3 * df2a -df2Ast * rf1 *
df3a + df2Ast * rf3 * df1a + df3Ast * rf1 * df2a -df3Ast * rf2*df1a)/Di
  DeltaAst = (df1c * rf2 * df3a - df1c * rf3 * df2a - df2c * rf1 * df3a
+ df2c * rf3 * df1a + df3c * rf1 * df2a -df3c * rf2 * df1a) / Di
  a = a - Deltaa
  c = c - Deltac
  Ast = Ast - DeltaAst
  PRINT TAB(12); "a ="; USING U43; a;
  PRINT TAB(31); "c ="; USING U43; c;
```

```
   PRINT TAB(49); "Ast ="; USING U46; Ast
   PRINT #2, TAB(12); "a ="; USING U43; a;
   PRINT #2, TAB(31); "c ="; USING U43; c;
   PRINT #2, TAB(49); "Ast ="; USING U46; Ast
   ctemp = c: atemp = a: Asttemp = Ast
   ctmp = c + delc: atmp = a + dela: Asttmp = Ast + delAst
   CALL Prep(f1, f2, f3, dk1, dNp)
   rf1 = f1: rf2 = f2: rf3 = f3
LOOP UNTIL ABS(Deltaa)<.001 AND ABS(Deltac)<.001 AND ABS(DeltaAst)<.001
PRINT #2,
PRINT #2, "REINFORCEMENT STRESSES :"
ii = INT(nd / 2 + .5)
FOR i = 1 TO ii
   PRINT #2, "R ="; i; TAB(13); "σs ="; USING U73; Sigma(i);
   IF i + ii > nd THEN
      PRINT #2,
   ELSE
      PRINT #2, TAB(36); "R ="; i + ii; TAB(55); "σs =";
      PRINT #2, USING U73; Sigma(i + ii)
   END IF
NEXT i
PRINT
PRINT "Section : x ="; USING U43 + "         "; xltemp;
PRINT "Ac ="; USING U73; Actemp
PRINT "            y ="; USING U43; yltemp
PRINT

PRINT "Compression Zone : a ="; USING U43 + "         "; a;
PRINT "x ="; USING U43 + "         "; xl;
PRINT "Acc ="; USING U73; Ac
PRINT "                   c ="; USING U43 + "        "; c;
PRINT "y ="; USING U43; yl
PRINT
PRINT "Total Reinforcement Area ="; USING U46; Ast
PRINT #2,
PRINT #2, "Section : x ="; USING U43 + "        "; xltemp;
PRINT #2, "Ac ="; USING U73; Actemp
PRINT #2, "          y ="; USING U43; yltemp
PRINT #2,
PRINT #2, "Compression Zone : a ="; USING U43 + "        "; a;
PRINT #2, "x ="; USING U43 + "        "; xl;
PRINT #2, "Acc ="; USING U73; Ac
PRINT #2, "                   c ="; USING U43 + "        "; c;
PRINT #2, "y ="; USING U43; yl
PRINT #2,
PRINT #2, "Total Reinforcement Area ="; USING U46; Ast
PRINT
PRINT "Press any key..."
WHILE INKEY$ = ""
WEND
GOSUB Son
Er:
BEEP: PRINT : PRINT "ERROR IN DATA !"
Son:
CLOSE
END
```

```
Ciz:
Lin = CSRLIN
CALL Plot(Ugn, UDat)
VIEW PRINT 3 TO 23
LOCATE Lin, 1
RETURN
Format:
U43 = "####.###"
U16 = " #.######"
U46 = "####.######"
U06 = " .######"
U73 = "#######.###"
RETURN

SUB Area
Begin:
  Ac = 0: x1 = 0: y1 = 0
  FOR k = 1 TO np
    IF k = np THEN
      xt = x(1)
      yt = y(1)
    ELSE
      xt = x(k + 1)
      yt = y(k + 1)
    END IF

 IF xt = x(k) THEN
   xp = x(k)
 ELSE
```

```
      dLamdap = (yt - y(k)) / (xt - x(k))
      IF c / a + dLamdap = 0 THEN c = c + c / 100: GOTO Begin
      xp = (c - y(k) + dLamdap * x(k)) / (c / a + dLamdap)
    END IF
    yp = c * (1 - xp / a)
    IF x(k - 1) = x(k) THEN
      xm = x(k)
    ELSE
      dLamdam = (y(k - 1) - y(k)) / (x(k - 1) - x(k))
      IF c / a + dLamdam = 0 THEN a = a + a / 100: GOTO Begin
      xm = (c - y(k) + dLamdam * x(k)) / (c / a + dLamdam)
    END IF
    ym = c * (1 - xm / a)
    sA = 0
    IF y(k) <= c * (a - x(k)) / a THEN
      sA = (x(k) * (ym - yp) + xp * (y(k) - ym) + xm * (yp - y(k))) / 2
    END IF
    Ac = Ac + sA
    x1 = x1 + sA * (x(k) + xp + xm)
    y1 = y1 + sA * (y(k) + yp + ym)
  NEXT k
  IF Ac <> 0 THEN
    x1 = x1 / 3 / Ac
    y1 = y1 / 3 / Ac
  END IF
  Ac = ABS(Ac)
```

```
END SUB

SUB Gnrt (U, Itype, n, Ls)
  CALL Reads(U, "R1", okun, Ls)
  nd1 = okun
  CALL Reads(U, "R2", okun, Ls): IF NOT Ls THEN EXIT SUB
  nd2 = okun: IF nd2 < nd1 THEN Ls = FALSE: EXIT SUB
  CALL Reads(U, "X1", okun, Ls): IF NOT Ls THEN EXIT SUB
  x(np + nd1) = okun
  CALL Reads(U, "Y1", okun, Ls): IF NOT Ls THEN EXIT SUB
  y(np + nd1) = okun
  CALL Reads(U, "X2", okun, Ls): IF NOT Ls THEN EXIT SUB
  x(np + nd2) = okun
  CALL Reads(U, "Y2", okun, Ls): IF NOT Ls THEN EXIT SUB
  y(np + nd2) = okun
  xa = (x(np + nd2) - x(np + nd1)) / (nd2 - nd1)
  ya = (y(np + nd2) - y(np + nd1)) / (nd2 - nd1)
  Itype(nd1) = Itype
  FOR l = nd1 + 1 TO nd2
    x(np + l) = x(np + l - 1) + xa
    y(np + l) = y(np + l - 1) + ya
    Itype(l) = Itype
  NEXT l
  n = nd2
END SUB
SUB Plot (Ugn, UDat)
```

```
STATIC default
DIM Image(1975)
k = 0
```

```
FOR i = 1 TO 25
  IF i = 25 THEN Imcolor = SCREEN(i, 1, 1)
  FOR j = 1 TO 79
    k = k + 1
    Image(k) = SCREEN(i, j)
  NEXT j
NEXT i
CLS
IF default = 0 THEN default = 1
PRINT "Scale Factor <"; default; "> ="; 
INPUT " ", sf
IF sf = 0 THEN sf = default
default = sf
SCREEN 2
PRINT TAB(INT((79 - LEN(Ugn)) / 2)); Ugn
LOCATE 25, 68: PRINT UDat;
zmaxx = x(1)
zmaxy = y(1)
zminx = zmaxx
zminy = zmaxy
FOR i = 1 TO np
  IF zmaxx < x(i) THEN zmaxx = x(i)
  IF zmaxy < y(i) THEN zmaxy = y(i)
  IF zminx > x(i) THEN zminx = x(i)
  IF zminy > y(i) THEN zminy = y(i)
NEXT i
```

```
  zmx = ABS(zminx) + ABS(zmaxx)
  zmy = ABS(zminy) + ABS(zmaxy)
  ratx = sf * 300 / zmx
  raty = ratx / 3
  IF raty * zmy > 160 THEN
    raty = sf * 160 / zmy
    ratx = raty * 3
  END IF
'
' Axes
'
  x = 600 - (600 - zmx * ratx) / 2 - ABS(zminx) * ratx
  y = (200 - zmy * raty) / 2 + ABS(zminy) * raty
  LINE (x, 10)-(x, 190), , , &HFF00
  LINE (0, y)-(600, y), , , &HFF00
  LINE (0, y - 8)-(4, y - 4)
  LINE (0, y - 4)-(4, y - 8)
  LINE (x + 8, 184)-(x + 10, 186)
  LINE (x + 8, 188)-(x + 12, 184)
'
' Polygons
'
  PSET (x - x(1) * ratx, y(1) * raty + y)
  FOR i = 1 TO np
    LINE -(x - x(i) * ratx, y(i) * raty + y)
  NEXT i

  LINE -(x - x(1) * ratx, y(1) * raty + y)
'
' Reinforcements
'
```

```
  FOR i = np + 1 TO np + nd
    j = i - np
    xx = x - x(i) * ratx
    yy = y(i) * raty + y
    CIRCLE (xx, yy), 1
    FOR k = 2 TO LEN(STR$(j))
      xx = xx + 6
      SELECT CASE MID$(STR$(j), k, 1)
        CASE "1": GOSUB 1
        CASE "2": GOSUB 2
        CASE "3": GOSUB 3
        CASE "4": GOSUB 4
        CASE "5": GOSUB 5
        CASE "6": GOSUB 6
        CASE "7": GOSUB 7
        CASE "8": GOSUB 8
        CASE "9": GOSUB 9
        CASE "0": GOSUB 10
        CASE ELSE
      END SELECT
    NEXT k
  NEXT i
'
' Modified Axis
'
  LINE (x - a * ratx, y)-(x, c * raty + y)
  LINE (x - a * ratx, y)-(x, c * raty + y)
  WHILE INKEY$ = ""
  WEND
  SCREEN 0
  k = 0
  FOR i = 1 TO 25
    FOR j = 1 TO 79
      IF i=1 OR (i=25 AND (j<4 OR j>51)) OR (i=25 AND j>12 AND j<16)
THEN
        COLOR Imcolor
      ELSE
        COLOR 7
      END IF
      k = k + 1
      LOCATE i, j
      PRINT CHR$(Image(k));
    NEXT j
  NEXT i
  COLOR 7
  EXIT SUB
1 :
  LINE (xx, yy + 2)-(xx + 3, yy)
  LINE -(xx + 3, yy + 4)
  RETURN

2 :
  CIRCLE (xx + 2, yy + 1), 2, , 0, 3.14
  LINE (xx + 4, yy + 1)-(xx, yy + 4)
  LINE -(xx + 4, yy + 4)
  RETURN
```

```
3 :
  CIRCLE (xx + 2, yy + 1), 2, , 5, 3.14
  CIRCLE (xx + 2, yy + 3), 2, , 4, 0
  RETURN
4 :
  LINE (xx + 4, yy + 3)-(xx, yy + 3)
  LINE -(xx + 3, yy)
  LINE -(xx + 3, yy + 4)
  RETURN
5 :
  LINE (xx + 4, yy)-(xx, yy)
  LINE -(xx, yy + 2)
  LINE -(xx + 3, yy + 2)
  LINE -(xx + 4, yy + 3)
  LINE -(xx + 3, yy + 4)
  LINE -(xx, yy + 4)
  RETURN
6 :
  CIRCLE (xx + 2, yy + 3), 2
  LINE (xx, yy + 2)-(xx, yy + 1)
  LINE -(xx + 1, yy)
  LINE -(xx + 3, yy)
  RETURN
  7 :
    LINE (xx, yy)-(xx + 4, yy)
    LINE -(xx + 1, yy + 4)
    RETURN
  8 :
    CIRCLE (xx + 2, yy + 1), 2
    CIRCLE (xx + 2, yy + 3), 2
    RETURN
  9 :
    CIRCLE (xx + 2, yy + 1), 2
    LINE (xx + 4, yy + 2)-(xx + 4, yy + 3)
    LINE -(xx + 3, yy + 4)
    LINE -(xx + 1, yy + 4)
    RETURN
  10 :
    CIRCLE (xx + 2, yy + 2), 2, , , , 1
    RETURN
  END SUB

  SUB Prep (f1, f2, f3, dk1, dNp)
    f1 = 0: f2 = 0: f3 = 0
    zmnx = x(1)
    zmny = y(1)
    FOR i = 1 TO np
      IF zmnx < x(i) THEN zmnx = x(i)
      IF zmny < y(i) THEN zmny = y(i)

FOR i = 1 TO np
  IF y(i) = 0 THEN IF zmnx > x(i) THEN zmnx = x(i)
  IF x(i) = 0 THEN IF zmny > y(i) THEN zmny = y(i)
NEXT i
z = a * c / SQR(a ^ 2 + c ^ 2)
```

```
z11 = (a - zmnx) * c / SQR(a ^ 2 + c ^ 2)
z12 = (c - zmny) * a / SQR(a ^ 2 + c ^ 2)
IF z11 > z12 THEN z1 = z11 ELSE z1 = z12
Epsc = Epscu * z / z1
FOR i = 1 TO nd
  Epss = Epsc * (dk1 * (x(np + i) / a + y(np + i) / c) - 1)
  IF Itype(i) THEN      'MILD STEEL
    IF ABS(Epss) < fyd / Es THEN
      Sigmas = Es * Epss
    ELSE
      Sigmas = Epss * fyd / ABS(Epss)
    END IF
  ELSE                       'COLD-WORK STEEL
    i1 = 0: i2 = 0
    IF ABS(Epss) <= Eps(6) THEN i1 = 5: i2 = 6
    IF ABS(Epss) <= Eps(5) THEN i1 = 2: i2 = 5
    IF ABS(Epss) <= Eps(2) THEN i1 = 1: i2 = 2
    Sigmas = 0
    FOR j = i1 TO i2
      p = 1
      FOR k = i1 TO i2
        IF j<>k THEN p = p * (ABS(Epss) - Eps(k)) / (Eps(j) -Eps(k))
      NEXT k
      Sigmas = Sigmas + p * z(j)
    NEXT j
```

```
      Sigmas = Sigmas * Epss / ABS(Epss)
    END IF
    Sigma(i) = Sigmas
    f1 = f1 + Sigmas
    f2 = f2 + (x(np + i) - xn) * Sigmas
    f3 = f3 + (y(np + i) - yn) * Sigmas
  NEXT i
  CALL Area
  f1 = fcd * Ac - f1 * Ast / nd - dNp
  f2 = f2 * Ast / nd - fcd * (x1 - xn) * Ac
  f3 = f3 * Ast / nd - fcd * (y1 - yn) * Ac
END SUB

SUB Reads (U, Uoku, okun, Ls)
  U = " " + UCASE$(U) + " "
  Uoku = " " + Uoku
  Ls = FALSE
  UB = ""
  FOR i = 1 TO LEN(U)
    IF RIGHT$(MID$(U, 1, i), LEN(Uoku)) = Uoku THEN Ls = TRUE: EXIT FOR
  NEXT i
  IF NOT Ls THEN EXIT SUB
  FOR i = i TO LEN(U)
    IF RIGHT$(MID$(U, 1, i + 1), 1) = "=" THEN
      Ls = TRUE: EXIT FOR
    ELSE

    ELSE
      IF RIGHT$(MID$(U, 1, i + 1), 1) <> " " THEN Ls = FALSE: EXIT FOR
    END IF
  NEXT i
  IF NOT Ls THEN EXIT SUB


   FOR i = i + 1 TO LEN(U)
     IF RIGHT$(MID$(U, 1, i + 1), 1) <> " " THEN EXIT FOR
   NEXT i
   FOR j = 1 TO LEN(U)
     UB = UB + MID$(U, i + j, 1)
     IF RIGHT$(UB, 1) = " " THEN EXIT FOR
   NEXT j
   okun = VAL(UB)
 END SUB
```